

PDM Workbench

PDM Workbench Release 20.0 for Aras Innovator

Installation & Administration Manual

Version 2

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

https://plm.t-systems.net/en-DE/pdm-workbench

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

Email: *cmi_support@t-systems.com*

Manual History

Version	Date	Version	Date	Version	Date
1.0	Apr 2005	3.6	Apr 2014	11.0	Nov 2019
2.0	Nov 2006	3.7	Oct 2014	12.0	May 2020
2.1	Nov 2007	3.8	Apr 2015	13.0	Nov 2020
2.2	Sep 2008	3.9	Oct 2015	14.0	May 2021
2.5	Sep 2010	4.0	Apr 2016	15.0	Jan 2022
3.0	Oct 2011	5.0	Oct 2016	16.0	Nov 2022
3.1	Feb 2012	6.0	Apr 2017	17.0	Jun 2023
3.2	Mar 2012	7.0	Oct 2017	18.0	May 2024
3.3	Oct 2012	8.0	May 2018	19.0	Mar 2025
3.4	Apr 2013	9.0	Dec 2018	20.0	Dec 2025
3.5	Oct 2013	10.0	May 2019		

This edition 20.0 of the manual 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

Email: *cmi_support@t-systems.com*

Preface

About this Manual

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

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

Related Documents

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

Manual Title	Version
PDM Workbench Installation & Administration Manual	*20.0*
PDM Workbench User Manual	*20.0*

Trademarks

CATIA is a registered trademark of Dassault Systèmes.

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

CHAPTER 1 1

OVERVIEW 1

SYSTEM HARDWARE AND SOFTWARE REQUIREMENTS 1
Aras Innovator Server 1
CATIA V5 Client 1
INSTALLATION STEPS 2

CHAPTER 2 3

ADAPTING CATIA V5 3

LOADING PWBCATV5 SOFTWARE FROM CD-ROM 3
PWBCATV5 INSTALLATION 3
Configuring the Installation 3
SILENT INSTALLATION 12
Parameters 12
Usage 14
SILENT UN-INSTALLATION 14
Parameters 14
Usage 14
TROUBLESHOOTING R18 (64 BIT) 15
TESTING THE INSTALLATION 16
Windows 16
CATIA V5 CONFIGURATION 16
SWITCHING TO THE PWBSCHEMA FILE SUITED FOR BOM PART STRUCTURES 18
PWBSCHEMA MODIFICATION 18
SETTING OF ENVIRONMENT VARIABLES 19
ADMINISTRATIVE LOCK FOR PDM WORKBENCH PREFERENCES 19
INSTALLATION WITHOUT PWB SPECIFIC ENVIRONMENT VARIABLES 20
SUPPORT USER SPECIFIC EXCHANGE DIRECTORY WHEN USING "TO CATIA" 21

CHAPTER 3 23

PDM WORKBENCH DATA MODEL 23

INSTALLATION 23
Standard Package 24
BOM Part Structure Configuration 24
Open in CATIA 24
Synchronize CAD Structure to BOM 24

CHAPTER 4 25

PDM WORKBENCH SERVER DLL 25

COPYING THE DLL 25
MODIFYING THE SERVER CONFIGURATION FILE 25

CHAPTER 5 27

CLIENT CUSTOMIZATION 27

DISPLAY NAMES 27
ICONS 27
DATA MODEL DEFINITION 28

CHAPTER 6 29

SERVER CONFIGURATION 29

CONFIGURATION VARIABLES 29
PWB Logging on Server 29

CONFIGURATION ITEMS 30
REPLACEMENT OF CLASS "ARASUTIL" WITH CLASS "PWBSERVERADDIN.PWBSERVERAPI" IN CUSTOM SERVER METHODS 31
MAKING SURE CAD/NATIVE_FILE DEFINITION IS "FIXED" 32
AUTOMATICALLY UPDATING THE "PART BOM" QUANTITY IF INSTANCES ARE DELETED IN THE ARAS INNOVATOR WEB CLIENT 32

CHAPTER 7 33

CONFIGURATIONS FOR SPECIFIC FUNCTIONALITIES 33

STANDARD CONFIGURATION 33
Exchange Map 33
SOAP target URL 33
SOAP Client Call Path Environment Variable 34
File Client Call Path Environment Variable 34
Maximum Expansion Level 34
Read only 34
Reload after Update 34
Allow loading different File Versions 35
Show out-of-date Link Info 35
Update after Create new CATIA file 35
Upload changed Files at Sync Failure 35
Asynchronous File Upload 35
Rollback on File Upload Error 36
Required Attributes 36
Maximum Description Attribute Length 36
Show Data Source NLS Values 36
Rename existing Files in Exchange Map 37
Auto Login on Session expired 37
Session Settings (contains Setting for Windows Single-Sign-On) 37
Use Custom Login Token Provider 38
Language Settings 39
Date Format 39
Key Attribute 39
Class Attribute 39
Relation Attribute 39
Relationship Attribute 39
Left Relationship Attribute 39
Right Relationship Attribute 40
Left Relation Class Attribute 40
Right Relation Class Attribute 40
Extended Relation Class Attribute 40
Last Modification Date Attribute 40
Mark superseded Nodes 40
Part Classes 40
Select Type of additional Parts in CAD Document Structure Data Model 41
TOOLBAR CONFIGURATION 41
CUSTOMER-SPECIFIC ENVIRONMENT 41
Use Case "Query" (loadOthersReadOnly="true") 43
Use Case "Update" 43
Use Case "Load" (loadOthersReadOnly="false") 43
Use Case "Load" (loadOthersReadOnly="true") 43
DATA MODEL CONFIGURATION 44
BOM Part Structure Data Model 44
Support Part BOM Classification 44
CAD Document Structure Data Model 45
Client/Server "UseBomPartStructure" Setting Compatibility Check 46
BOM PART STRUCTURE CONFIGURATION 47
DIFFERENT LIFE CYCLE STATES FOR CAD AND PART WITH "PROMOTE" 47
QUERY CONFIGURATION 48
Configuring the Size of the "PDM Query" Dialog 48
QueryOrderByAttribute 48

MaxQueryResults *48*
Default Sort Criteria for Query Results *49*
Display Custom Query Buttons *49*
Modal Dialog *49*
Check if no Query Attribute filled *49*
Display String in Query List View *49*
Automatically loading CATDrawings or linked CATParts *50*
Keep Filter Attribute Values when changing the Type *51*
"Select Date" Enhancement in the "PDM Query" Dialog *51*
Special Context Menu for Multi-Select in Query Window *54*
UPDATE CONFIGURATION 55
Optionally only showing the "PDM Update" Dialog when new Files exist *55*
"PDM Create" Dialogs *55*
Show "PDM Create" Dialogs with Multi Column *55*
"Create new file versions at update?" Button *56*
Default create Version Value *56*
"Unclaim after Save?" Button *57*
Show create Parts *57*
Default create Parts Value *58*
Validate Structure before Update *58*
Let Aras Innovator delete orphaned Files created at Update *59*
Add newly created and updated Part or CAD Document Items to existing Items *59*
Fetch Part Number allow manual Input *62*
Clear pre-filled Attributes from "PDM Create" Dialog *62*
Optional "Clear" Button in "Register" Dialogs of "CAD" Item Definitions *63*
Deny create of CAD at top level Structure in BOM Part Structure Data Model *63*
Custom Row in "Update to PDM" Dialog *65*
Optional Refresh before Update *66*
Recursive Save of linked Documents *66*
Always save the "Path to Root" CATProducts of modified CATParts *67*
REVISE CONFIGURATION 67
Context Action "Revise" *67*
Modifiable Attributes *67*
Expand Structure *68*
Revising as a different User ("ReviseAs") *68*
DUPLICATE CONFIGURATION 68
Duplicate *68*
Immediately start Update *69*
DUPLICATE STRUCTURE CONFIGURATION 69
Context Action "Duplicate Structure" *69*
Variant A (only available in CAD Document Structure Data Model) *70*
Variant B *71*
Duplicate Structure provides "originatedFrom" Property (in Variant B) *71*
Duplicate Structure Handling of Standard Parts (in Variant B) *73*
Duplicate Structure Enhancements (in Variant B) *74*
CATIA V5 WINDOW CONFIGURATION 75
"Properties" Dialog *75*
Allow deactivated CATProduct and CATPart Instances *77*
PDM STATUS INFORMATION IN THE CATIA TREE 78
Configuration *78*
Usage *80*
PDM STRUCTURE WINDOW CONFIGURATION 80
Displaying Part Structure Instances as separate Nodes *80*
Displaying Attributes in PDM Nodes in several Lines *81*
PWB Window Color *82*
Preselection of Nodes *82*
Context Action "Delete relation" *83*
Comparing PDM Structure Trees *83*
Selecting Nodes in the PDM Structure Window *84*
Alternative Icons for PDM Structure Nodes *84*
EXPAND CONFIGURATION 85
Context Action "Expand" *85*

Context Action “Expand Multiple Levels” *85*
Context Action “Custom Expand” *85*
Context Action “De-Expand” *86*
Context Action “Expand Multiple Levels including Bom Children” *86*
Explicit Display and Control of “AsSaved” and “Current” Expand Resolutions *87*
Check for 'CAD Structure' expand resolution. *87*
MANAGE CONTEXT PRODUCTS 87
Configuration *88*
USE ENVIRONMENT VARIABLE FOR CONTEXT PRODUCT 88
Configuration *88*
NAMING CONFIGURATION - NUMBERING 89
CadDocNumberAttr *89*
PartNumberAttr *89*
Autoname Support using Aras Innovator Sequence Items *89*
Autoname Functionality can use a Server Method *91*
Autoname Functionality in Create Dialog *92*
INITIAL VERSION STRINGS 93
InitialVersionCadDoc *93*
InitialVersionPart *93*
PROMOTE STATES 93
Context Action “Promote” *93*
PromoteSourceStates *93*
PromoteTargetStates *93*
PromotePartSourceStates *94*
PromotePartTargetStates *94*
PromoteCADSourceStates *94*
PromoteCADTargetStates *94*
STANDARD PART FUNCTIONALITY 95
StandardPartAdmin *95*
Standard Part Functionality for BOM Part Structure Data Model *95*
Standard Part Functionality for CAD Document Structure Data Model *97*
MATERIAL 99
Configuration *101*
MANAGEMENT OF CATIA TEMPLATES 101
TemplateFileAdmin *101*
Server Configuration *102*
Client Configuration *103*
TEMPLATE FILE SUPPORT FOR “CREATE PART” WITH TEMPLATES DEPENDING ON THE PART TYPE 105
Configuration *105*
THUMBNAIL CONFIGURATION 107
Create Thumbnails from Types *107*
Create Thumbnails by CATIA *107*
DERIVED FILES CONFIGURATION 108
Derived Files Tabulator *108*
ADDITIONAL CAD DOCUMENT TYPES 109
Design Tables *109*
Archives *112*
CATIA V4 Models *113*
CGRs as native Files *114*
CATIA Catalogs *115*
Support floating Content in Catalog *116*
Configurable Catalog Keywords *117*
Custom Method to handle Catalog References *117*
CATProcess *119*
ATTRIBUTE MAPPING CONFIGURATION 120
Standard Attribute Mapping *120*
Allow Mapping of Part and CAD Property to the same CATIA Standard Attribute . *123*
POSSIBILITY TO USE CATIA USER-DEFINED PROPERTIES IN AUTONAME AND PREPROCDLGATTRS CUSTOM METHODS 123
QUERY / EXPAND PERFORMANCE CAD STRUCTURE MODE WITH PART ATTRIBUTE MAPPING 124

Not setting mapped “CATIA Standard Attribute” Values in the “PDM Create” Dialog *127*
Prefill CATIA Revision Attribute in “Register” / “PDM Create” Dialog *127*
User Attribute Mapping *128*
CATIA User Defined Attributes *130*
Drawing Attribute Mapping *131*
Extended Attribute Mapping Functionality *131*
Inertia Attribute Mapping *132*
PDM to CAD Attribute Mapping only for CATIA Files claimed by the User *133*
ADD TEMP CONFIGURATION 133
Context Action “Add Temp” *133*
AddTemp Prefix *133*
VERSIONING CONFIGURATION 134
Context Action “Create new generation” *134*
Context Action “Delete newest generation” *134*
Context Action “Unlink and delete newest generation” *134*
Only one new Generation of a CAD Document per “Claim” Action *135*
LOAD CONFIGURATION 137
Context Action “Load” *137*
Context Action “Load (Current)” *138*
Context Action “Load in Context” *138*
OPEN CONFIGURATION 138
Context Action “Open in new PDM Window” *138*
Context Action “Open File” *138*
Context Action “Open File with Link” *138*
Context Action “Open File temporarily” *138*
CATDrawing: Loading referenced Data as “Current” *139*
Context Action “Open related drawings” *139*
SHOW NEIGHBORHOOD CONFIGURATION 139
Bounding Box Definition *139*
Context Action “Show Neighborhood” *139*
USE TOOLTIP FOR ATTRIBUTES IN PDM WORKBENCH DIALOGS 140
Configuration *140*
LINK SUPPORT 142
Link Management *142*
Possibility to filter Relation Definitions in Schema File depending on Configuration Settings *143*
Basic Drawing Link Support *143*
Download related 3D File *144*
Option to block the Update if linked File is not saved *144*
Duplicate related Drawings *144*
Basic Multi-Model Link Support *145*
REPRESENTATION TYPES 145
Part CAD Filter *145*
Additional Rep Types *146*
Attach additional Non-Bom CATParts to Part *148*
Support generic Shape Representations *153*
CHECKS 155
Check File Versions in Structure *155*
Check CATParts in Visualization Mode *155*
Check CATIA Part Numbers before Update *155*
Check for CAD Owner *155*
Check for CAD Document CATIA Release at PDM Update *156*
Optional Check for broken Links at Update *156*
Delete Relations of non-loaded Instances *156*
Check CAD Links *156*
Context Action “Check CAD Links” *157*
MISCELLANEOUS CONTEXT ACTIONS 157
Context Action “Properties” *157*
Context Action “Unclaim” *157*
Context Action “Claim” *157*
Context Action “Unclaim all” *157*

Context Action "Claim all" *157*
Disable Confirmation Messages for Claim/Unclaim Actions *157*
Optional Availability of Claim and Unclaim Toolbar Icons in the Toolbar *158*
Context Action "Duplicate" *158*
Context Action "Duplicate-Delete" *158*
Context Action "Duplicate-Supersede" *158*
Context Action "Insert PDM Node" *158*
Insert from Aras Innovator keep "PDM Query" Dialog *158*
Context Action "Replace Node" *158*
Context Action "Execute" *159*
Context Action "Update structure relations" *159*
Context Action "Update parent relation" *159*
Context Action "Newest Generation Info" *159*
Context Action "Update Item" *159*
Context Action "PDM Create in Context" *159*
Context Action "Highlight" *159*
Context Action "Copy element attributes" *160*
Context Action "Delete" *160*
Context Action "Synchronize to BOM" *160*
NON-BOM CATPARTS AND CATPRODUCTS 160
PERFORMANCE OPTIMIZATION FOR NON-BOM CATPARTS IN BOM PART STRUCTURE DATA MODEL 160
Configuration *160*
SUPPORT FOR THE NEW CAD STRUCTURE INSTANCE HANDLING INTRODUCED IN ARAS INNOVATOR 9.4 AND 10.0 161
"CAD IS MASTER FOR INSTANCES" FUNCTIONALITY 161
Configuration *161*
REPAIR INSTANCE WITH USECADISMASTER = TRUE 161
CUSTOM METHOD FOR SET-BASED CREATE OF INSTANCE NAMES 161
Configuration *162*
CONFIGURABLE CATIA COMPONENTS SUPPORT 164
SUPPORT ELECTRICAL / TUBING 164
SUPPORT FOR RELATING A NEW CATIA FILE TO AN EXISTING PART 165
LOCAL WORKSPACE INFORMATION 165
CONFIGURATION OF BOM PART STRUCTURE 166
POSSIBILITY TO CALL A SERVER METHOD FOR A PDM ITEM 167
RECONNECT AT UPDATE 168
CLEAN UP / HOUSEKEEPING OF PWB_XMAP DIRECTORY 170
SETTING CONFIGURATION INFORMATION ON STRUCTURE RELATIONS 171
EXCLUDING CHILD NODE TYPES FOR CREATION 172
USE SERVER METHOD FOR QUANTITY 173
Configuration *173*
RELEASED CACHE MODE 173
HTTP COMPRESSION 177
"OPEN IN CATIA" FROM THE ARAS INNOVATOR CLIENT 178
"OPEN IN CATIA" USES DYNAMIC SETTING HOW TO FETCH PROPERTIES FROM ARAS INNOVATOR 181
Configuration *181*
"OPEN IN CATIA" WITH CONSTRUCTION SPACES 181
Configuration *182*
Access Rights *182*
"OPEN IN ARAS" IN CATIA V5 CLIENT 182
Configuration *182*
Window Management *183*
"OPEN IN ARAS" IN AICD ENVIRONMENT 184
"OPEN IN ARAS" - ALLOW CONFIGURABLE WEB BROWSER 184
Configuration *184*
CONFIGURABLE PROPERTIES FOR CUSTOMMETHOD_PREPROCRELINSTCREATTRS 185
Configuration *185*
CREATING AND DELETING RELATIONS WITHOUT UPDATING THE SOURCE ITEM 185
Configuration *185*
ALIGN CAD STRUCTURES WITH MULTIPLE VERSIONS OF THE SAME CAD 185

Configuration *185*
DENY LOAD IF THE SAME FILE IS LOADED IN DIFFERENT VERSION 185
AUTOMATIC SAVE OF PWB SESSION FILE 186
Configuration *186*
WARNING WHEN THE USER WANTS TO UNCLAIM MODIFIED FILES 186
Configuration *186*
COMPATIBILITY WITH ACI 186
DYNAMIC DIALOG FUNCTIONALITY 189
Configuration *189*
Usage *190*
Activate 'DynContextPromote' functionality by default *192*
GIVING THE PATH TO ROOT INFORMATION FOR SPECIFIC PROBLEMATIC PSN NODES TO THE USER 192

CHAPTER 8 193

CLIENT SCHEMA FILE CONFIGURATION 193

STRUCTURE OF THE SCHEMA FILE 193
Attributes of the Tag "PWBSchema": *193*
NLS Support for Display Names *194*
Configuration Settings *194*
see "142 ..... *197*
"object": 1 - n *198*
"relation": 1 - n *198*
"attribute": 0 - n *198*
"pwbAttribute": 0 - n *198*
"dataSource": 0 - n *198*
PDM ATTRIBUTES AND FORM ATTRIBUTES 198
Description of the Widget Types *200*
Login Form *202*
PWB Workbench toolbar - embed login / logout buttons in a fly-out toolbar. *203*
PDM OBJECTS 203
Description of PDM Objects *204*
Tooltip of PDM Objects *204*
Actions on PDM Objects *204*
Context Actions *205*
Disabling Context Menu Items in the CATIA Structure Window *209*
PDM Object Forms *209*
PDM RELATIONS 210
Description of PDM Relations *210*
"Relationship" Tags *210*
"Left and Right Object" Classes *212*
PDM Relation Forms *212*
DATA SOURCES 212
Data Source "Value" tag *212*
Complete Example of using a Data Source Tag *213*

CHAPTER 9 215

TROUBLESHOOTING 215

INVALID DUPLICATE CAD INSTANCES WITH ARAS 3D CAD TO PDF CONVERSION 215
TYPE OR NAMESPACE "ARASUTIL" COULD NOT BE FOUND … 216
LOCKING CAD DOCUMENTS IN CATIA FAILS WHEN THE CONVERSION SERVER IS INSTALLED 217
CONVERSION TASKS FAIL FOR UPDATED OR NEW CAD FILES FROM PDM WORKBENCH 217
CATIA CRASHES WHEN LOADING LARGE ASSEMBLIES WITH VERY OLD CATIA FILES 218

Table of Figures

PICTURE 1: DIRECTORY STRUCTURE OF THE PDM WORKBENCH INSTALLATION FILES 3
PICTURE 2: WELCOME TO THE INSTALLATION 5
PICTURE 3: LICENSE AGREEMENT 5
PICTURE 4: CHOOSE USERS 6
PICTURE 5: CHOOSE LOCATION OF PDM PACKAGE 6
PICTURE 6: CHOOSE LOCATION OF PDM PACKAGE (WITH PROPOSAL) 7
PICTURE 7: CHOOSE INSTALL LOCATION 7
PICTURE 8: CHOOSE CATIA V5 INSTALLATION 8
PICTURE 9: CHOOSE ORIGINAL CATIA V5 ENVIRONMENT 8
PICTURE 10: CHOOSE EXCHANGE DIRECTORY 9
PICTURE 11: CHOOSE LOCATION OF SOAP TARGET URL 10
PICTURE 12: CHOOSE DATABASE NAME 10
PICTURE 13: CHOOSE DATA MODEL 11
PICTURE 14: SUBSUMPTION 11
PICTURE 15: INSTALLATION PROGRESS 12
PICTURE 16: INSTALLATION FINISHED 12
PICTURE 17: EVENT PROPERTIES 15
PICTURE 18: PDM WORKBENCH TOOLBAR BEFORE THE LOGIN 16
PICTURE 19: THE PDM WORKBENCH TOOLBAR AFTER THE LOGIN 16
PICTURE 20: CATIA V5 GENERAL→GENERAL SETTINGS 17
PICTURE 21: CATIA V5 GENERAL→DOCUMENT SETTINGS 17
PICTURE 22: PDM SESSION CONFIGURATION DIALOG 18
PICTURE 23: PDM WORKBENCH PREFERENCES (ADMINISTRATOR VIEW) 19
PICTURE 24: PDM WORKBENCH PREFERENCES (USER VIEW) 20
PICTURE 25: PDM ARAS INNOVATOR IMPORT UTILITY 23
PICTURE 26: ARAS INNOVATOR SERVER CONFIGURATION VARIABLES 29
PICTURE 27: LOGGING CONFIGURATION VARIABLES 30
PICTURE 28: PWB CONFIGURATION ITEM IN ARAS INNOVATOR 30
PICTURE 29: ITEM TYPE “CAD” 32
PICTURE 30: “BOM INSTANCE” SERVER EVENTS 32
PICTURE 31: TOOLBAR DURING ASYNCHRONOUS FILE UPLOAD 35
PICTURE 32: TOOLBAR AFTER ASYNCHRONOUS FILE UPLOAD 36
PICTURE 33: SAMPLE ROLLBACKCADONERROR CONFIGURATION 36
PICTURE 34: ADD PROJECT AND CUSTOMER TO THE ITEM TYPES “CAD” AND “PART” 42
PICTURE 35: SAMPLE CUSTOMMETHOD_CHECKCATIAENVIRONMENT CONFIGURATION 42
PICTURE 36: SAMPLE USEBOMPARTSTRUCTURE CONFIGURATION 44
PICTURE 37: STRUCTURE IN THE BOM PART STRUCTURE DATA MODEL 44
PICTURE 38: CLASS STRUCTURE OF PART BOM 45
PICTURE 39: PWB CONFIGURATION – SPECIAL PART BOM RELATION 45
PICTURE 40: STRUCTURE IN THE CAD DOCUMENT STRUCTURE DATA MODEL 46
PICTURE 41: CONFIGURATION ERROR AT LOGIN 47
PICTURE 42: SAMPLE BOMPARTSTRUCTURELOCKUNLOCKRELATEDITEM CONFIGURATION 47
PICTURE 43: SAMPLE QUERYORDERBYATTRIBUTE CONFIGURATION 48
PICTURE 44: SAMPLE MAXQUERYRESULTS CONFIGURATION 48
PICTURE 45: QUERY LIST VIEW IN “PDM QUERY” DIALOG WITH DISPLAY STRING 50
PICTURE 46: “LOAD WITH LINKS” AND “LOAD WITH DRAWINGS” CHECK BOXES 50
PICTURE 47: “PDM WORKBENCH OPTIONS” DIALOG WITH HIGHLIGHTED CHECK BOXES 51
PICTURE 48: “PDM QUERY” DIALOG SETTING IN “PDM WORKBENCH OPTIONS” DIALOG 51
PICTURE 49: SAMPLE CUSTOMMETHOD_PREPROCQRYDLGATTRS CONFIGURATION 52
PICTURE 50: SAMPLE SHOWCREATEDIALOGSDURINGUPDATE CONFIGURATION 55
PICTURE 51: COMBINED “PDM CREATE” DIALOG, WITH THE PART DIALOG ABOVE AND THE CAD DOCUMENT DIALOG BELOW 55
PICTURE 52: COMBINED “PDM CREATE” DIALOG, WITH THE PART DIALOG ON THE LEFT AND THE CAD DOCUMENT DIALOG ON THE RIGHT 56
PICTURE 53: “UPDATE TO PDM” DIALOG WITH “CREATE NEW FILE VERSIONS AT UPDATE?” 56
PICTURE 54: OPTIONAL “UNCLAIM AFTER SAVE” BUTTON 57

Picture 55: “Update to PDM” dialog with “Create new parts at update?” 57
Picture 56: Sample CustomMethod_PreProcessCadStructure configuration 58
Picture 57: Sample PreProcessCadStructureBeforeUpdate configuration 58
Picture 58: Sample CustomMethod_PostProcUpdateInfo configuration 59
Picture 59: Sample CustomMethod_PostProcUpdate configuration 60
Picture 60: Folder list 60
Picture 61: Expanding “Is in Folders” in the PDM structure window 61
Picture 62: Expanding “Folder Items” in the PDM structure window 61
Picture 63: Expanded folder items in the PDM structure window 62
Picture 64: Non CAD top level structure with on the fly created CATProducts 63
Picture 65: Update Non-CAD top level structure -> Result SKIPPED 64
Picture 66: Customer-specific row in “Update to PDM” dialog 65
Picture 67: CATIA documents saved recursively following links 66
Picture 68: “Revise” dialog with modifiable attributes “major_rev” and “description” 68
Picture 69: Sample ReviseAs configuration 68
Picture 70: Duplicate structure - Variant A - Preselected list of documents 70
Picture 71: Duplicate structure - Variant A – Duplicate Part and Drawings 70
Picture 72: Duplicate structure - Variant B - Preselected list of documents 71
Picture 73: Item Type “CAD” – Add property “pwb_originated_from” 71
Picture 74: Duplicate Structure Handling of Standard Parts 73
Picture 75: New function in the “Duplicate structure” dialog 74
Picture 76: “Duplicate structure” dialog, hide “CATIA display” column 75
Picture 77: PDM Properties - UpdateItem 76
Picture 78: PDM Properties – Properties 77
Picture 79: CATProduct structure with a deactivated node 77
Picture 80: Sample CustomMethod_PreProcRelCreAttrs configuration 77
Picture 81: PDM status information in the CATIA tree 80
Picture 82: Sample UseSeparateRelationsForInstances configuration 80
Picture 83: Every part instance is shown as separate node 81
Picture 84: PDM Node Attributes displayed in several Lines 82
Picture 85: Default highlight behavior 83
Picture 86: Example icon names 85
Picture 87: Select Context Product 87
Picture 88: Currently used Context Product 87
Picture 89: Sample CustomMethod_ContextProduct configuration 88
Picture 90: Sample CadDocNumberAttr configuration 89
Picture 91: Sample PartNumberAttr configuration 89
Picture 92: Sample Sequence item 89
Picture 93: Sequence items used in example 90
Picture 94: Sequence item “CAD Document” 92
Picture 95: Sample InitialVersionCadDoc configuration 93
Picture 96: Sample InitialVersionPart configuration 93
Picture 97: Sample PromoteSourceStates configuration 93
Picture 98: Sample PromoteTargetStates configuration 93
Picture 99: Sample PromotePartSourceStates configuration 94
Picture 100: Sample PromotePartTargetStates configuration 94
Picture 101: Sample PromoteCADSourceStates configuration 94
Picture 102: Sample PromoteCADTargetStates configuration 94
Picture 103: Sample StandardPartAdmin configuration 95
Picture 104: Subclass “Standard” 95
Picture 105: Making CATParts in a local folder accessible 96
Picture 106: Defining a CATPart as a standard part 97
Picture 107: “Standard Part” check box for CAD Documents 97
Picture 108: “IsStandardPart” user-defined property 98
Picture 109: Three example standard part CATParts 98
Picture 110: User is logged in as administrator 98
Picture 111: The “Standard Part Administrator” identity 99
Picture 112: Material located at CATPart 99
Picture 113: Material located at Composite Parameters 100
Picture 114: Material located at Bodies 100
Picture 115: Sample CustomMethod_SaveUsedMaterials configuration 101

Picture 116: Template file administrator 101
Picture 117: Sample TemplateFileAdmin configuration 102
Picture 118: The “Template File Administrator” identity 102
Picture 119: PWB Configuration variables for template creation 102
Picture 120: “PDM Create” dialog containing “Is Template” checkbox 103
Picture 121: Template file update error message 103
Picture 122: Sample CreateThumbnailsFromTypes configuration 107
Picture 123: Windows Thumbnails 108
Picture 124: “Derived Files” tab 109
Picture 125: Activate multisheet option in CATIA V5 Tools→Options→General→Compatibility→Graphics Formats 109
Picture 126: Sample UseDesignTables configuration 110
Picture 127: Adding first design table 110
Picture 128: Renamed first design table file. 111
Picture 129: Adding second design table 111
Picture 130: Renamed second design table file. 111
Picture 131: Display of related design tables in the PWB structure window 111
Picture 132: Display of related design tables in the Aras Innovator web client 112
Picture 133: Sample UseArchives configuration 112
Picture 134: Add Mechanical/Archive to CAD 113
Picture 135: Add Mechanical/CatiaV4Model to CAD 114
Picture 136: Add Mechanical/cgr to CAD 115
Picture 137: Add Mechanical/catalog to CAD 116
Picture 138: Sample UseFloatingCatalogContent configuration 117
Picture 139: Configurable Catalog Keywords 117
Picture 140: Sample CatalogPartIdentifier configuration 117
Picture 141: Sample CatalogCadIdentifier configuration 117
Picture 142: Sample CustomMethod_ProcessCatalogContent configuration 118
Picture 143: Subclass /CAD/Mechanical/CATProcess 119
Picture 144: Subclasses /CAD Structure/CATProcessProduct and /CAD Structure/CATProcessResource 120
Picture 145: Standard attributes in the “Properties” dialog 121
Picture 146: Configuration of standard attributes in Aras Innovator 121
Picture 147: Standard attributes in the “Properties” dialog of the PDM node 122
Picture 148: Standard attributes in Aras Innovator window 122
Picture 149: Mapping of CATIA attribute Nomenclature from CAD and Part 123
Picture 150: PWB Configuration – Standard Attribute Mapping 127
Picture 151: CATIA V5 Properties of the CATPart and the “PDM Create” dialog 127
Picture 152: Configuration of user-defined attributes in Aras Innovator 128
Picture 153: User-defined attributes in the “Properties” dialog of the PDM node 129
Picture 154: User-defined attributes in Aras Innovator window 129
Picture 155: User-defined attributes in the “Properties” dialog 130
Picture 156: Configuration of drawing attributes in Aras Innovator 131
Picture 157: Example PWB Configuration attribute mapping 131
Picture 158: Sample inertia attributes mapping 132
Picture 159: Item Type “Part” 134
Picture 160: Switching on the “One Generation per Claim” functionality 135
Picture 161: Sample CustomMethod_GetCustomItemInfo configuration 135
Picture 162: Display help for PWB attributes 140
Picture 163: Sample UseAllCatiaLinkTypes configuration - “false” 142
Picture 164: Sample UseAllCatiaLinkTypes configuration - “true” 142
Picture 165: Sample UseOrigCatiaLinkTypeName-Drawing and UseOrigCatiaLinkTypeName-Reference configuration 143
Picture 166: Information about CATParts to be updated 144
Picture 167: Example rep type attribute 145
Picture 168: Sample rep type value list configuration 145
Picture 169: Sample Part CAD filter configuration 146
Picture 170: Different rep type values on CAD Documents in a structure 146
Picture 171: Two CATParts with different rep types related to the same part loaded at the same time 147
Picture 172: Create / Update list of Rep Types 147
Picture 173: Assign Rep Type List to CAD 147

Picture 174: Change pwb_rep_type 148
Picture 175: Configuration on Item Type “CAD” 149
Picture 176: List of valid representation types 149
Picture 177: Configurations for Representation Types in the PWB Configuration 150
Picture 178: Manage Representations 154
Picture 179: Class Structure “CAD” – Added “CATShape” and “wrl” 154
Picture 180: PWB Configuration – “CreateThumbnailsFromTypes” 154
Picture 181: Sample DeleteNotLoadedInstances configuration 156
Picture 182: Sample CadLinkCheckQueryAttribute configuration 156
Picture 183: Sample UseNonBom3DCadDocs configuration 160
Picture 184: Sample UseCadInstance configuration 161
Picture 185: Sample CustomMethod_PreProcBomInstAttrs configuration 162
Picture 186: BOM Configuration Management Type Overview 166
Picture 187: BOM Configuration Management Types in Aras Innovator 167
Picture 188: PWB Configuration item setting “ReconnectAtUpdateMethod” 168
Picture 189: Item Type “CAD” – Add property “pwb_orig_cad_partnumber” .... **Fehler! Textmarke nicht definiert.**
Picture 190: Sample OrigCadPartnumberAttr configuration 168
Picture 191: Sample UseSeparateRelationsForInstances configuration 171
Picture 192: All PDM Types can be selected 172
Picture 193: Only a sub-set of the PDM Types can be selected 172
Picture 194: Sample UpdateQuantityMethod configuration 173
Picture 195: Sample CadCacheFileItemProperty configuration 174
Picture 196: Setting the derived file creation to “view_file” 175
Picture 197: Defining the CATIA Released Cache directory 176
Picture 198: Setting the Released Cache PWB options 176
Picture 199: IIS – Compression settings 177
Picture 200: IIS - Install dynamic compression module 178
Picture 201: Import utility 179
Picture 202: Innovator variables 179
Picture 203: File “deepLinking.ts or .js” 182
Picture 204: Empty tab 183
Picture 205: Aras Innovator web client with Client URL 183
Picture 206: Window title “Aras Innovator” 184
Picture 207: Sample ArasWindowTitleSubString configuration 184
Picture 208: Warning dialog at “Unclaim” 186
Picture 209: Sample CustomMethod_GetCustomItemInfo configuration 187
Picture 210: Sample dynamic dialog custom server method 189
Picture 211: CAD Document in “Preliminary” state 190
Picture 212: “Custom Promote” custom action 190
Picture 213: Lifecycle states in dialog based on “Preliminary” 191
Picture 214: Lifecycle states in browser based on “Preliminary” 191
Picture 215: Result of “Custom Promote” action 191
Picture 216: Lifecycle states in dialog based on “Released” 192
Picture 217: Lifecycle states in browser based on “Released” 192
Picture 218: Single Line Editor Widget, update mode 200
Picture 219: Single Line Editor Widget, output mode 200
Picture 220: Multi Line Editor Widget, update mode 201
Picture 221: Combo Box Widget, select mode 201
Picture 222: Single Check Box Widget, select mode 201
Picture 223: Check Boxes Widget, select mode 201
Picture 224: Radio Buttons Widget, select mode 201
Picture 225: Single Selector List Widget, select mode 202
Picture 226: Restore contents for "PDM Workbench" 203
Picture 227: PDM Node in PWB window 204
Picture 228: Tooltip of PDM Node in PWB window 204
Picture 229: Select PDM object type in “PDM Query” dialog 205
Picture 230: Context actions for the type /Part/Assembly - Example 206
Picture 231: Relation icon with relationship and description attribute 211
Picture 232: Dynamic CAD Converter enabled 215
Picture 233: Name space could not be found 217

PICTURE 234: "ARAS 3D CAD TO PDF CONVERSION" RULE 217
PICTURE 235: UNCHECK "LOCK DEPENDENCIES ARAS 3D CAD TO PDF CONVERSION" RULE 218

CHAPTER 1

Overview

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

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 37

Server Installation of Aras Innovator on the following operating systems:

- Windows Server 2012, Windows Server 2016, Windows Server 2019, Windows Server 2022
- Detailed information see *Aras Documentation*: “Aras Innovator xx Platform Specifications”

CATIA V5 Client

Hardware

The T-Systems CATIA V5 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 Dassault Systèmes for CATIA V5.

Please refer to:

https://www.3ds.com/support/hardware-and-software/hardware-and-software-configurations/
for Graphics and CPU recommendations.

Select the respective CATIA V5 Release and Windows 10 64 bit as Operating System.

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

Software

On the CATIA 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

CATIA V5

CATIA V5 Client V5-6R2022 SP1, CATIA V5 Client V5-6R2023, CATIA V5 Client V5-6R2024, and CATIA V5 Client V5-6R2025 on the following operating systems:

- Windows 10 (64 Bit) for CATIA V5 Client V5-6R2022 SP1, CATIA V5 Client V5-6R2023, CATIA V5 Client V5-6R2024, and CATIA V5 Client V5-6R2025

Important notice:

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

Installation Steps

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

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

- Client installation: CATIA V5 Add-in (chapter 2)
- Client installation: License Manager (For the installation of “licman21” please refer to the *Licman 2.1 Installation Manual*.)
- Server installation: PDM Workbench data model and server methods (chapter 3)
- Server installation: PDM Workbench server DLL (chapter 4)

CHAPTER 2

Adapting CATIA V5

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

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

The **PWBCATV5_Rxx_xx** module includes all of the supported platform data in a compressed file. Thus, you should choose an installation location for all CATIA V5 clients.

In the following example sections it is supposed that the software will be installed within the directory **C:\Program Files\T-Systems\PWBCATV5_Rxx_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**.

Loading PWBCATV5 Software from CD-ROM

Windows 10

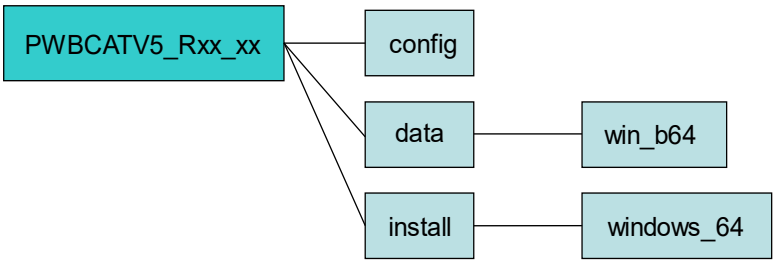
Use the Windows Explorer to locate the **D:\pwbcatv5\PWBCATV5_Rxx_xx.zip** file on the CD. Extract the content of the archive file to a temporary installation location.

PWBCATV5 Installation

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

Configuring the Installation

The **PWBCATV5_Rxx_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 PWBCATV5 module for the supported operating system mnemonics.

The supported operation system and its mnemonic is:

Windows 10 (64 Bit)	win_b64

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 **PWBCATV5_Rxx_xx\install\windows_64** of the installation package if you have installed the 64-Bit version of CATIA V5.

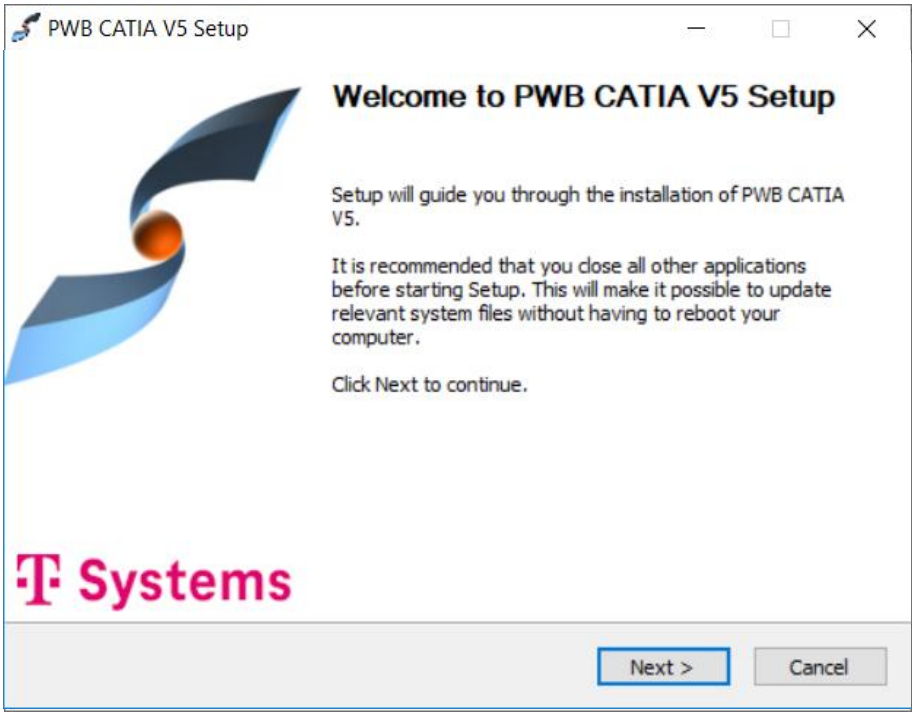
On **Windows 10** the User Account Control (UAC) will be triggered and you will have to agree that the setup program may make changes to the computer. The installer is signed with a "**T-Systems International GmbH**" certificate to ensure its integrity and source.

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

The licman21 license manager has to be installed on the CATIA V5 client host. For the installation of the license manager please refer to the *Licman 2.1 Installation Manual*.

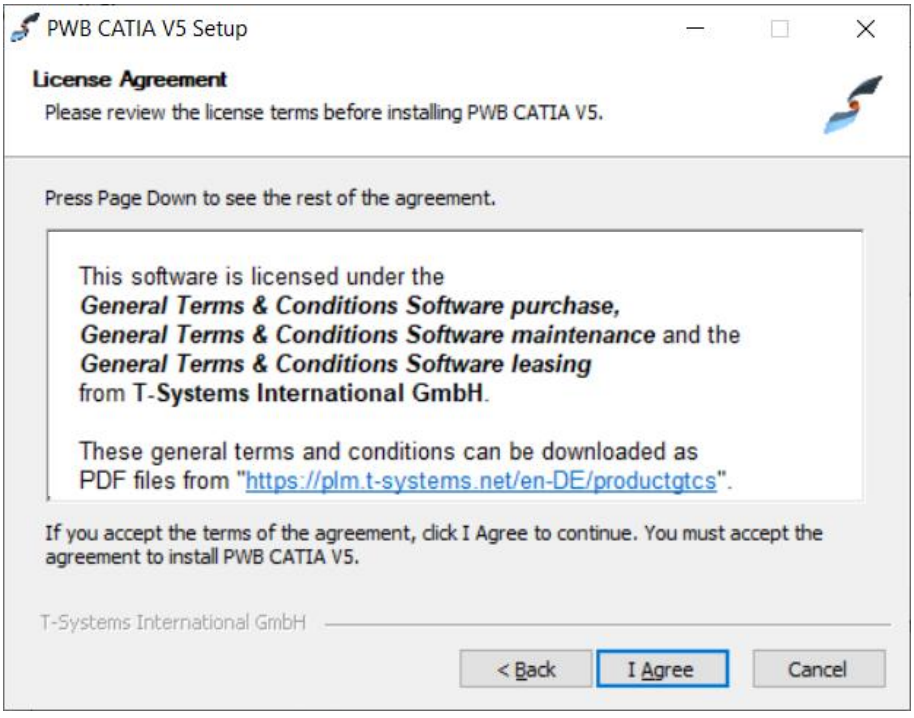
In the following the setup is shown step-by-step.

Installation process:



Picture 2: Welcome to the Installation

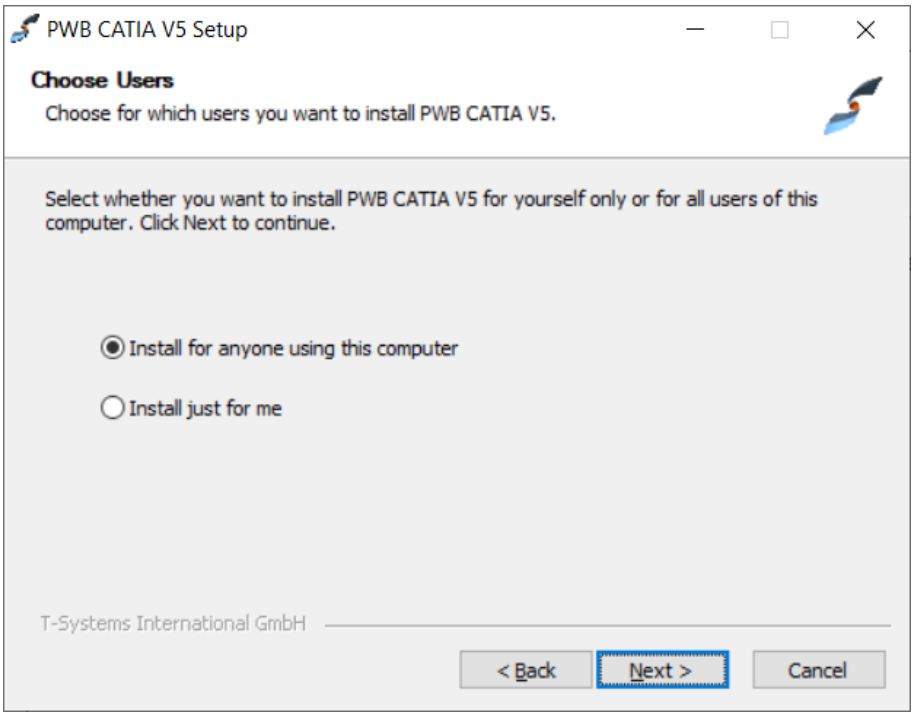
The installer software asks to approve the license terms (see *Picture 3: License Agreement*).



Picture 3: License Agreement

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

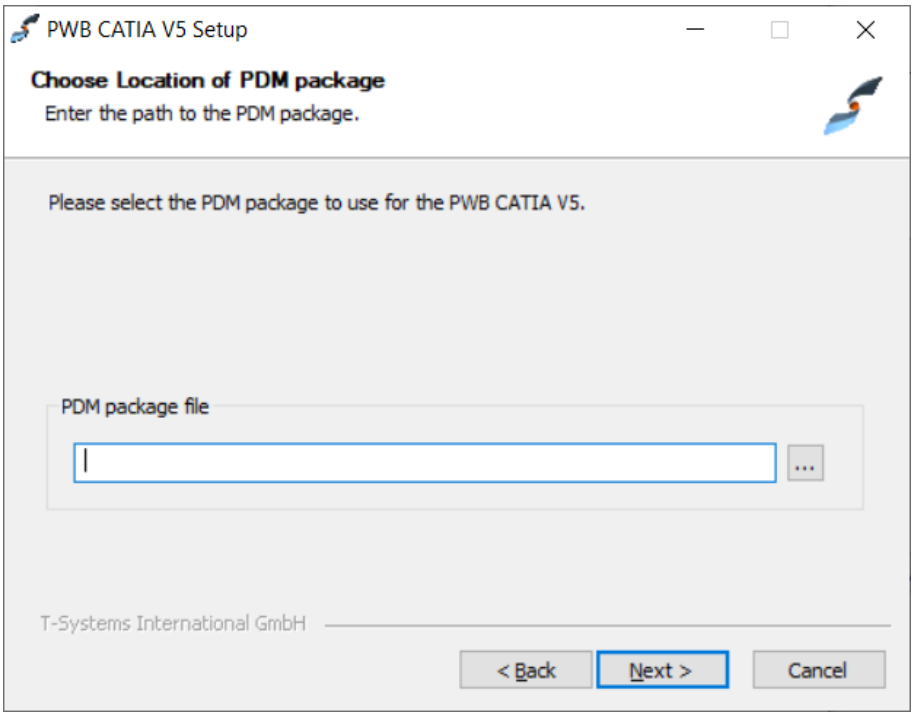
Next the installer will ask you to define the scope of the installation (see *Picture 4: Choose Users*). You can choose between an installation for anyone using the computer or just for the current user.



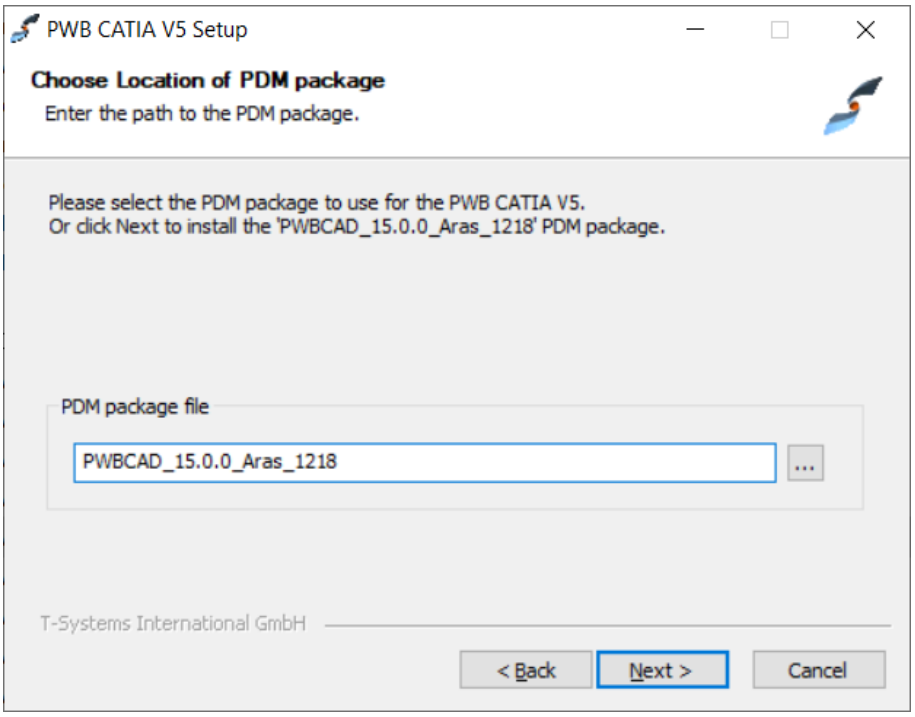
Picture 4: Choose Users

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

The installer asks for the location of the PDM package to use (see *Picture 5: Choose 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)*).



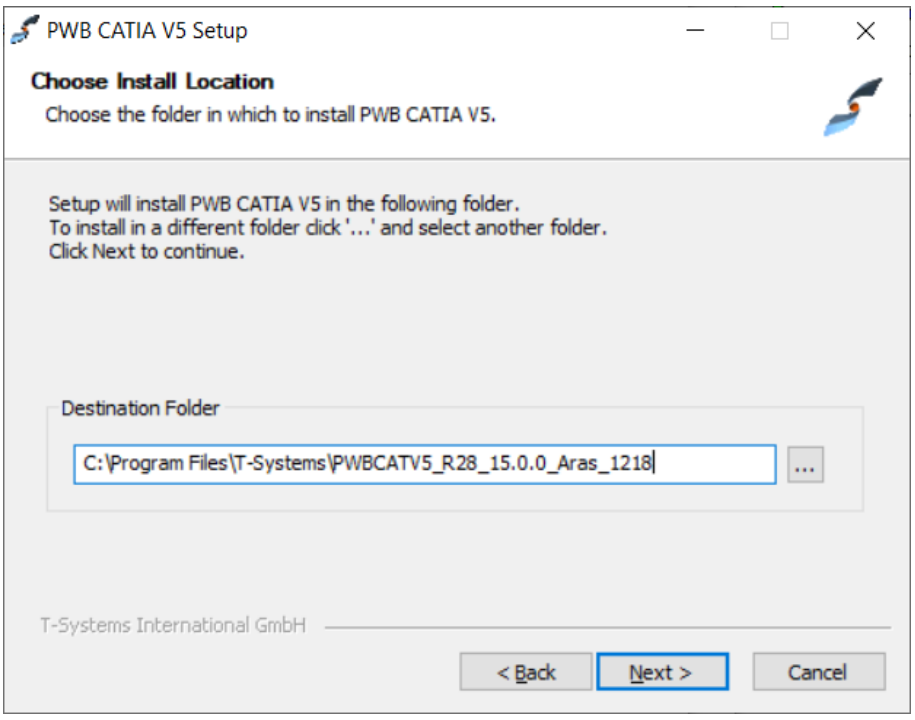
Picture 5: Choose Location of PDM package



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

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

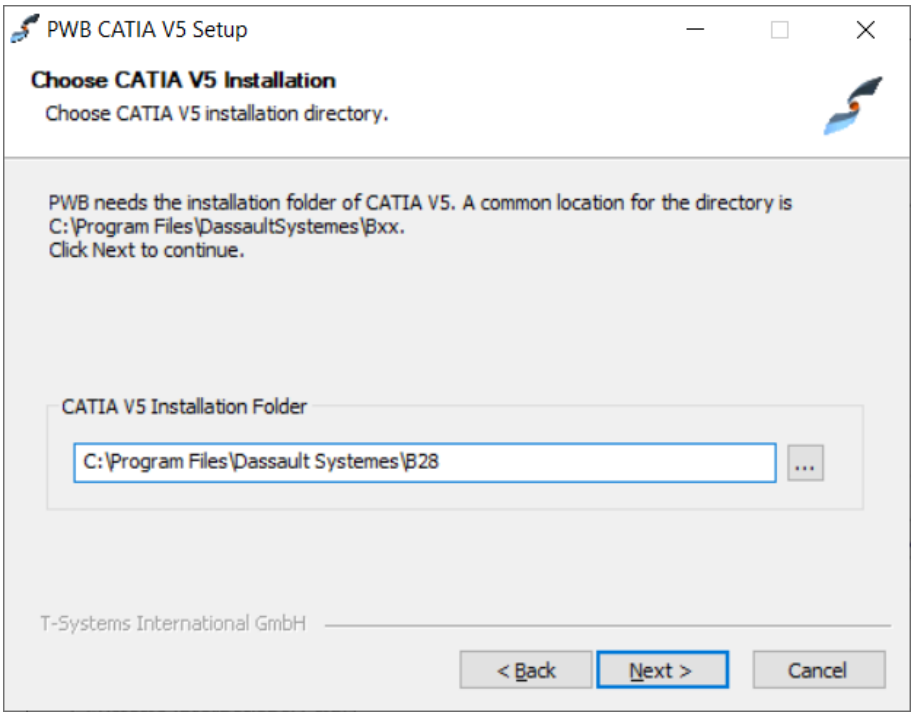
Next the installer will ask you for the target directory for the installation. You can use the given standard location or choose any other location (see *Picture 7: Choose Install Location*). The chosen folder must be empty or not existent.



Picture 7: Choose Install Location

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

The installation path of the CATIA to use needs to be specified (see *Picture 8: Choose CATIA V5 Installation*).

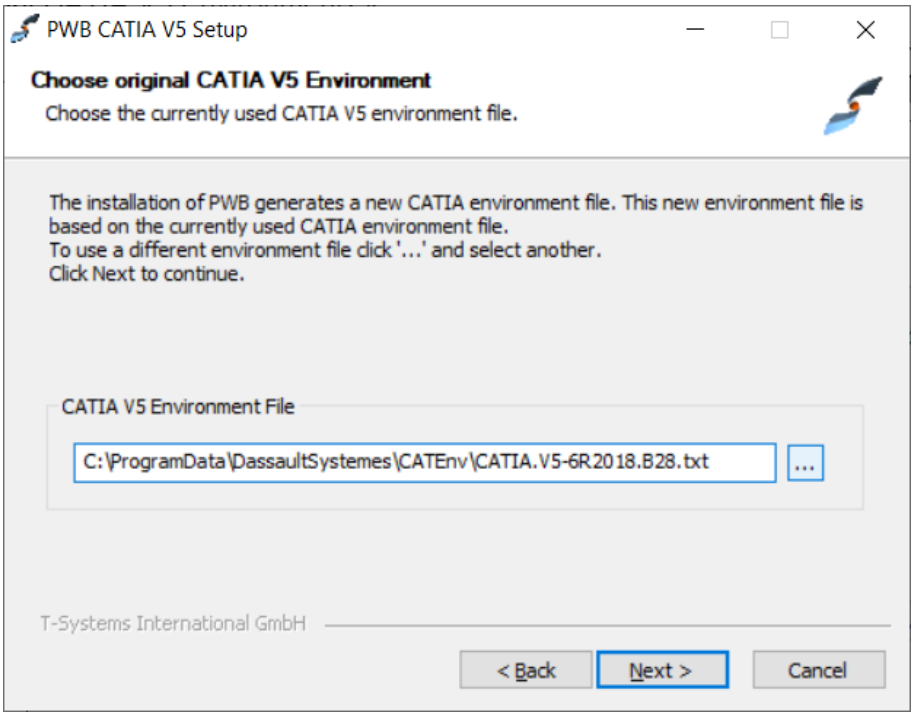


Picture 8: Choose CATIA V5 Installation

The installer software asks for the following input: **Original CATIA V5 Environment**.

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

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



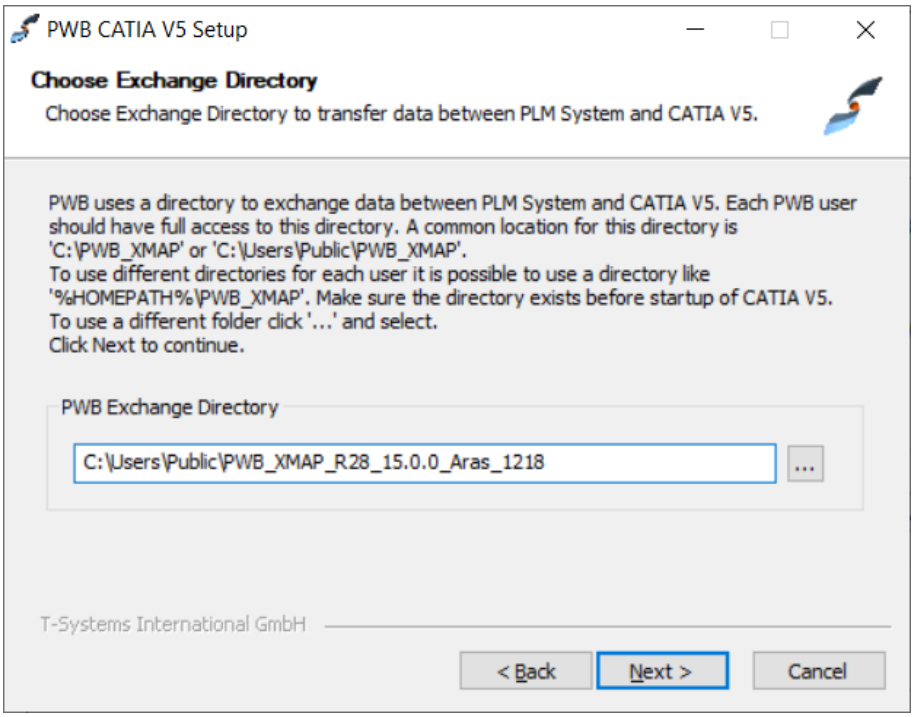
Picture 9: Choose original CATIA V5 Environment

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

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

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

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

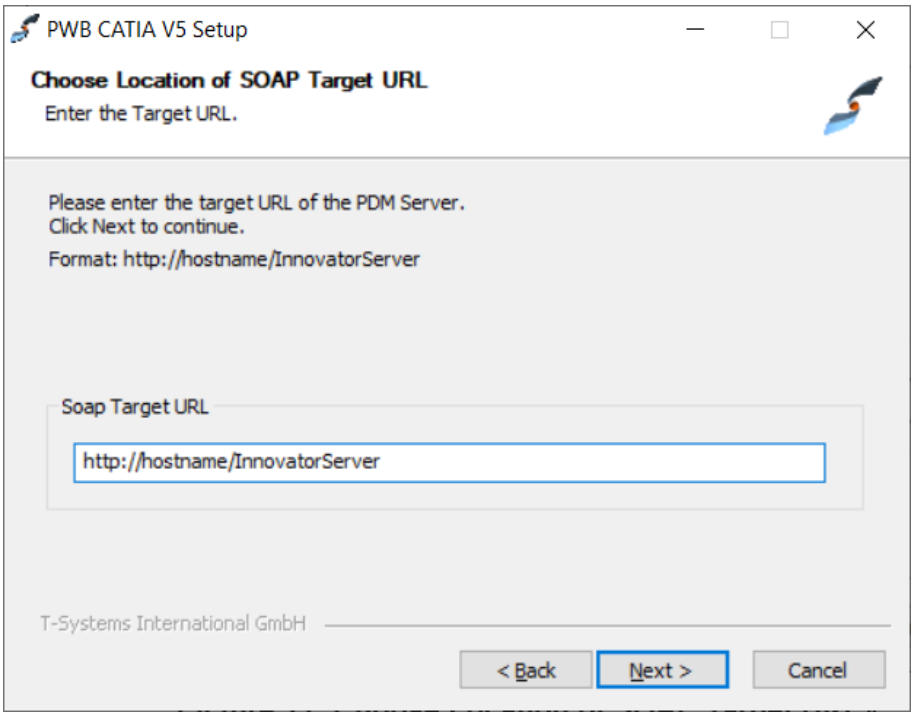


Picture 10: Choose Exchange Directory

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 11: Choose Location of SOAP Target URL*).

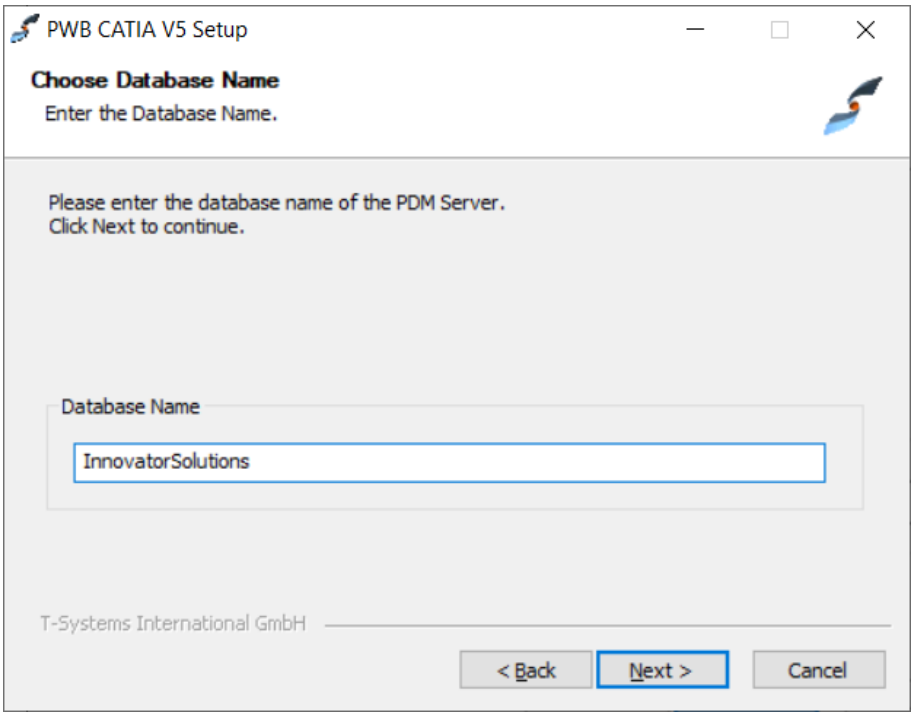
This URL defines the host on which the PDM Server is reachable.



Picture 11: Choose Location of SOAP Target URL

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

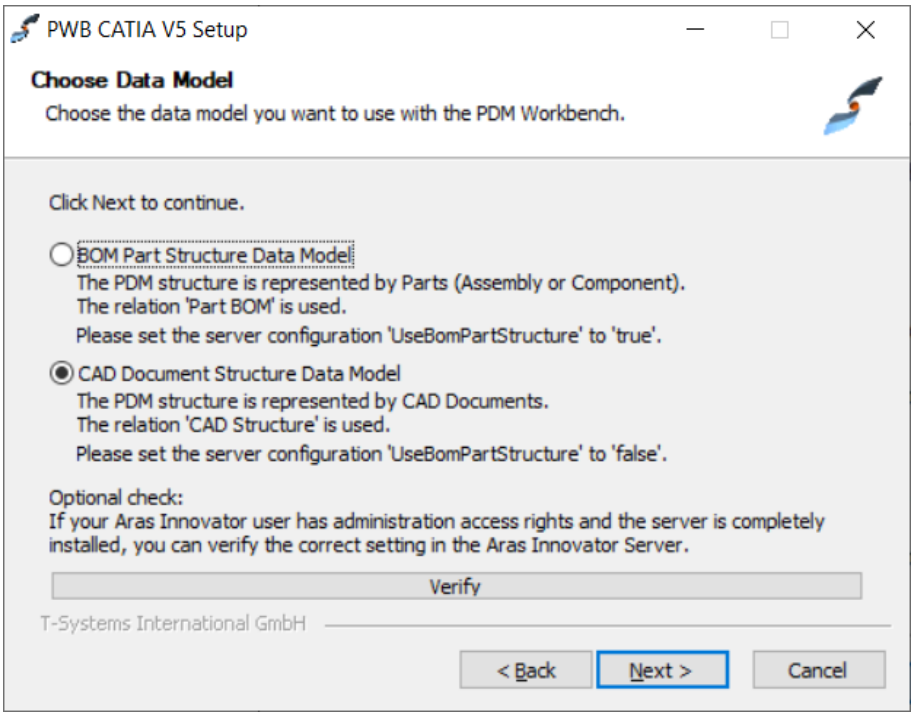
Now you have to add the Database Name (see *Picture 12: Choose Database Name*).



Picture 12: Choose Database Name

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

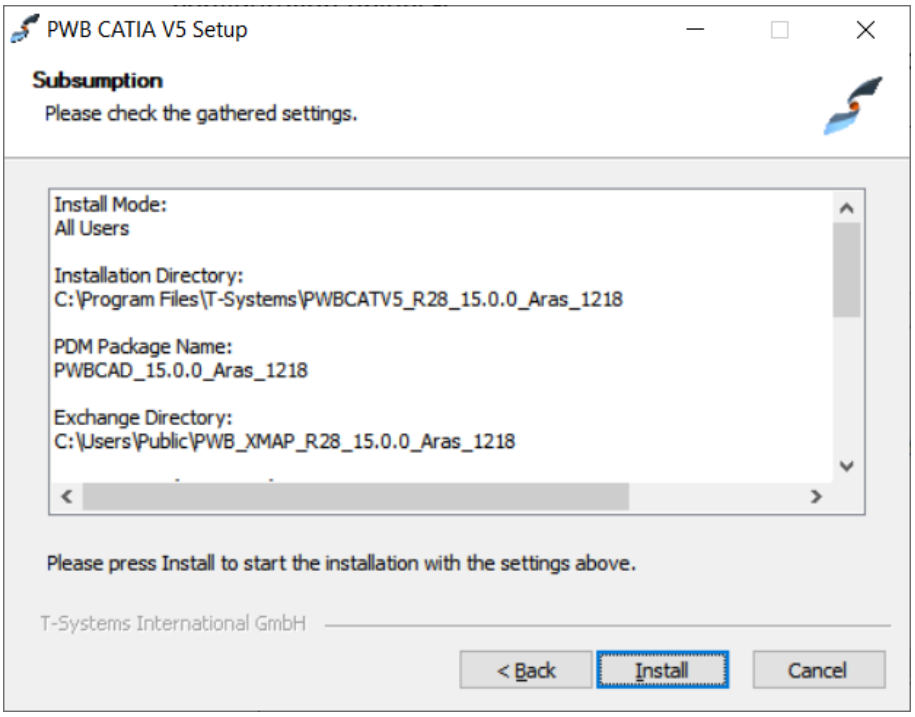
Finally you have to choose the data model to be used by the PDM Workbench (see *Picture 13: Choose Data Model*).



Picture 13: Choose Data Model

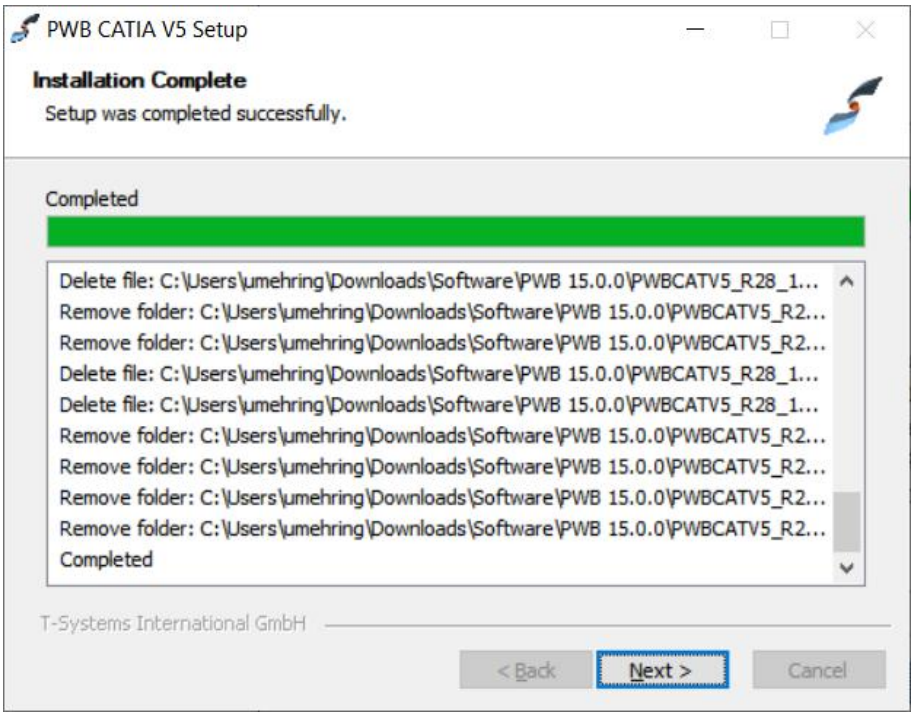
With the optional check you can directly log in into Aras Innovator with an administrative user and verify the setting of the data model (UseBomPartStructure) in the PWB configuration object.

After that you see the subsumption of your inputs before confirming them (see *Picture 14: Subsumption*).

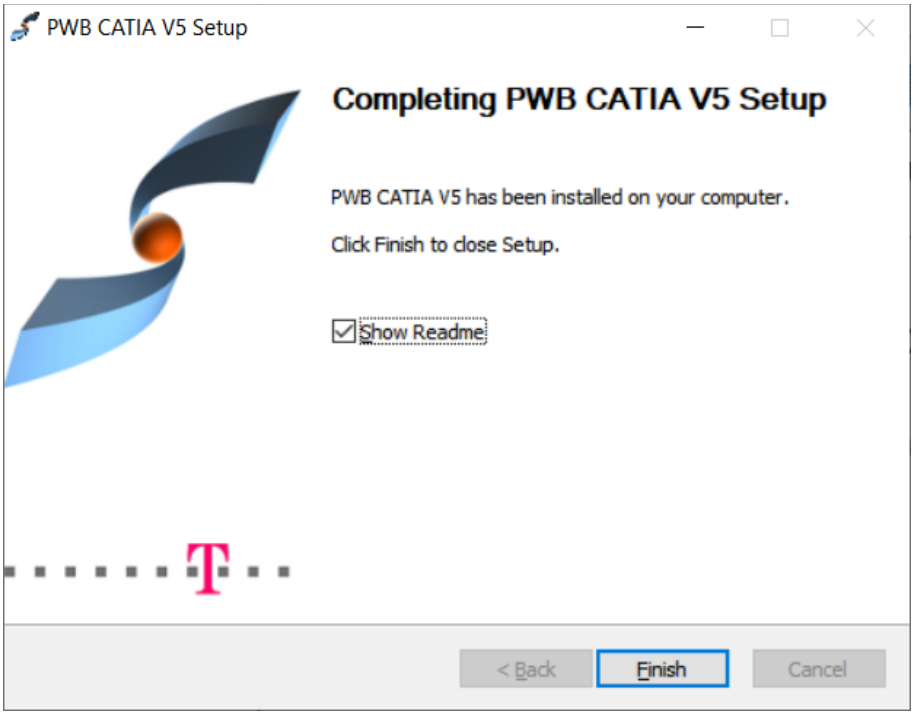


Picture 14: Subsumption

The installer will proceed in its process. The taken actions will be journalized (see *Picture 15: Installation progress*).



Picture 15: Installation progress



Picture 16: Installation finished

Silent Installation

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

Parameters

The following parameters are available for the silent installation:

Parameter name	Description	Sample value
/S	Activates the silent mode.	
/User= value	Installation only for yourself (“User”) or for all users of the computer (“Admin”). Default is the highest possible value.	Admin
/PdmPackageNamePath= (File full path)	The full path of the zip file or the directory which includes the PDM package.	D:\aras_client\PWBCAD_15.0.0_Aras_1218.zip D:\aras_client\PWBCAD_15.0.0_Aras_1218
/DeletePdmPackageInstallDir= (true|false)	Configures if the temporary PDM package installation directory will be deleted at the end of the installation process. The default value is “true”.	false
/TempDir= (Directory path)	Defines the temporary directory to which the PDM package will be unzipped/copied during the installation process. In the defined directory a sub-directory “pdm” will be created.	C:\temp
/CatiaInstDir= (Directory path)	The directory of the CATIA V5 installation.	C:\Program Files\Dassault Systemes\B28
/CatiaEnvFile= (File full path)	The full path of the currently used CATIA environment file.	C:\ProgramData\DassaultSystemes\CATEnv64\CATIA.V5-6R2018.B28.txt
/ExchangeMap= (Directory path)	The directory of the Exchange Directory.	C:\Users\Public\PWB_XMAP
/SoapTargetURL= (URL)	The SOAP target URL of the Aras Innovator server.	http://localhost/InnovatorServer
/DatabaseName= (Database Name)	The Database Name of the Aras Innovator server.	InnovatorSolutions
/DataModel = (CAD|BOM)	The data model. (Default value: “CAD”)	CAD
/D=(Directory path)	The target directory of the installation.	C:\Program Files\T-Systems\PWBCATV5_R28_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 parameter “/DeletePdmPackageInstallDir” is optional. The default value will be “true”.

The parameter “/TempDir” is optional. The temporary installation location directory “PWBCATV5_Rxx_xx\pdm” will be used as default value.

The both values for the CATIA installation are optional; the values 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 “/DataModel” is optional. The value “CAD” 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 “PWBCATV5_Rxx_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
/DeletePdmPackageInstallDir=false /TempDir="C:\temp"
/CatiaInstDir="C:\Program Files\Dassault Systemes\B28"
/CatiaEnvFile="C:\ProgramData\DassaultSystemes\CATEnv64\CATIA.V5-
6R2018.B28.txt" /ExchangeMap="C:\Users\Public\PWB_XMAP"
/SoapTargetURL="http://localhost/InnovatorServer"
/DatabaseName="InnovatorSolutions" /DataModel=BOM /D=C:\Program
Files\T-Systems\PWBCATV5_R28_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.

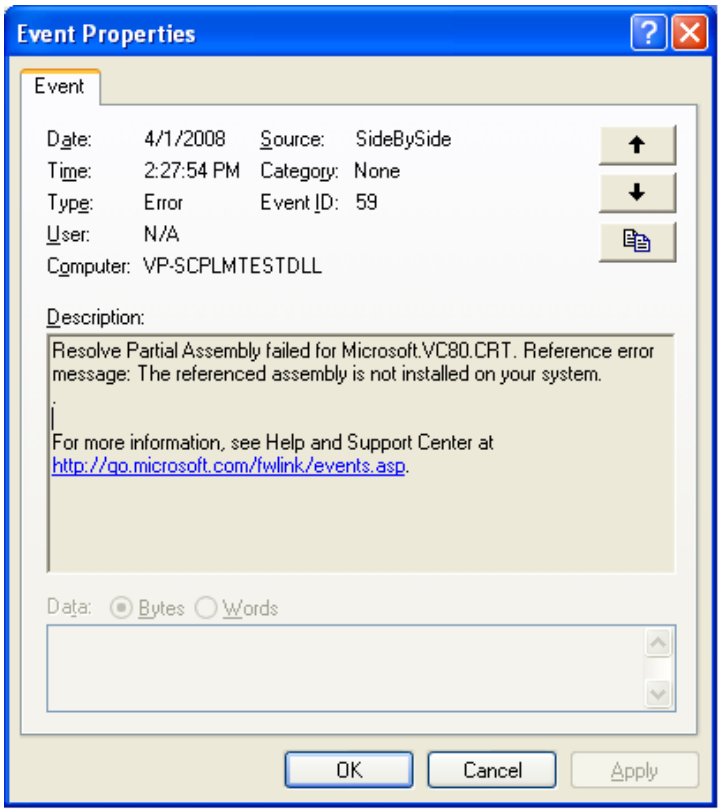
Start the silent un-installation with a command line like this example:

```
"C:\Program Files\T-
Systems\PWBCATV5_R28_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.

Troubleshooting R18 (64 Bit)

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



Picture 17: Event Properties

This error indicates that you need additional Windows runtime libraries.

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

```
PWBCATV5_R17_8.0.0\Windows_Runtime\x64\
(required for 64 Bit CATIA only)
PWBCATV5_R18_8.0.0\Windows_Runtime\x64\
(64 Bit CATIA)
PWBCATV5_R18_8.0.0\Windows_Runtime\x86\
(32 Bit CATIA)
```

There are two possibilities to install the new runtime:

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

This may need system privileges.

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

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

or

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

For 32 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x86.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_32V00\intel_a\code\bin\Microsoft.VC80.CRT`

For 64 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x64.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_32V00\win_b64\code\bin\Microsoft.VC80.CRT`

Testing the Installation

Windows

Windows 10: Use **Start → T-Systems PWB CATIA V5 Rxx yyy Innovator zzz → pwb_start** to launch CATIA V5.

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

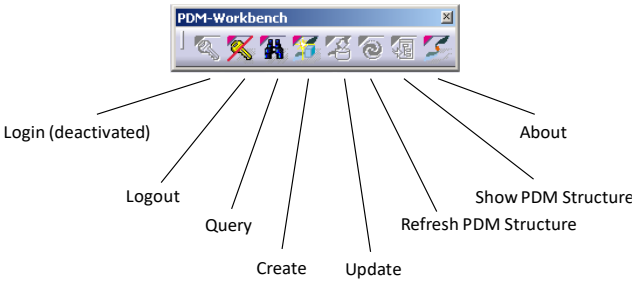
```
debug on (level 1)
PDM Workbench: Module Number 1030 :
license successfully allocated
```

The License Module Number may vary.

Within CATIA V5 the following toolbar has to be visible:



Picture 18: PDM Workbench toolbar before the login

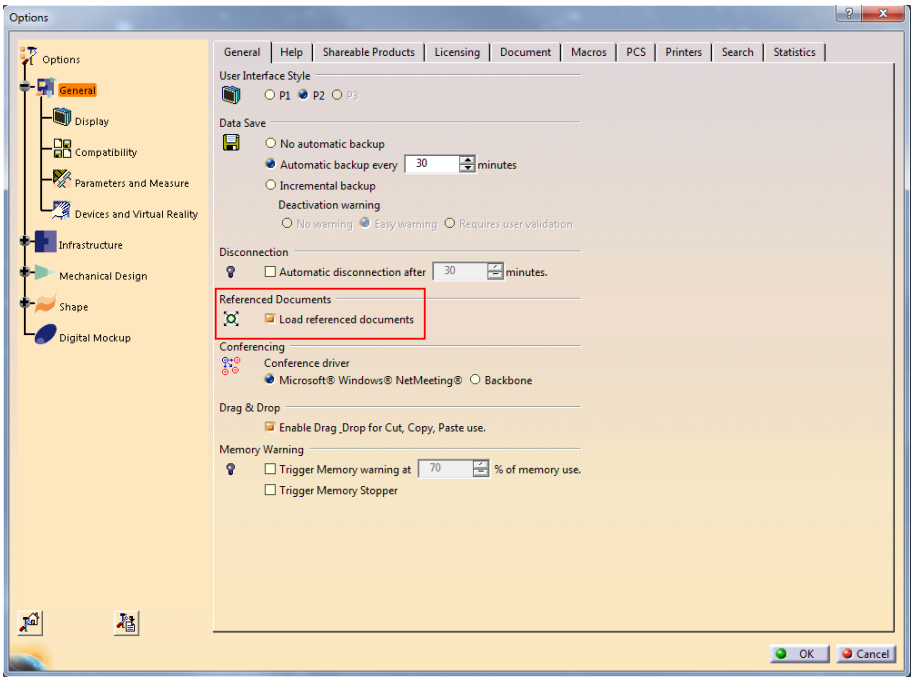


Picture 19: The PDM Workbench toolbar after the login

CATIA V5 Configuration

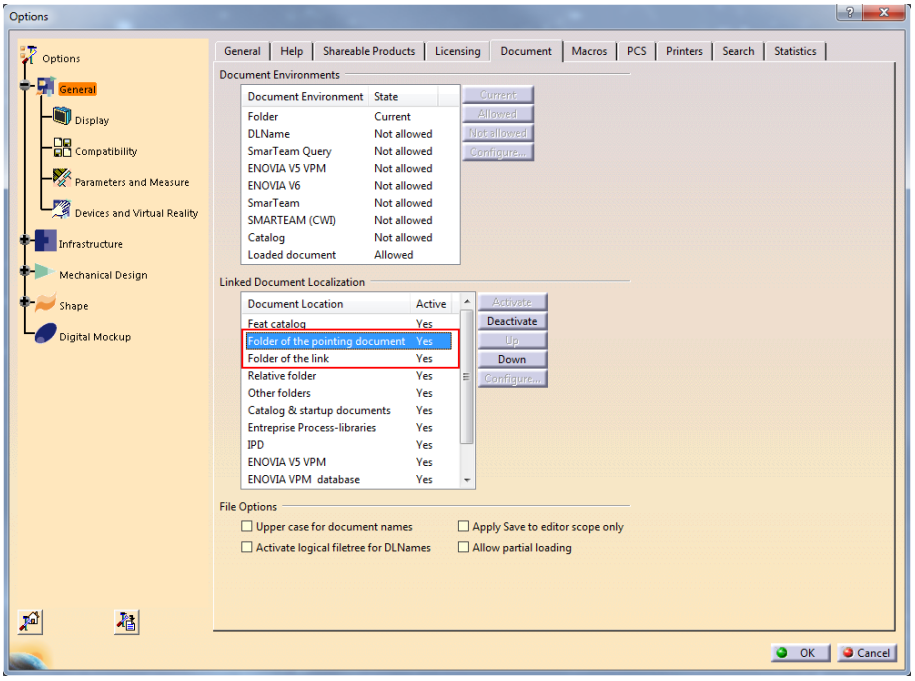
In the CATIA V5 Settings the following options have to be set as described below:

The *Load Referenced documents* option must be set in **Tools→Options→General** settings (see *Picture 20: CATIA V5 General→General Settings*).



Picture 20: CATIA V5 General→General Settings

In the *Linked Document Localization* the Options ***Folder of the pointing document*** and ***Folder of the link*** must be set to yes, and should be in this order (see *Picture 21: CATIA V5 General→Document Settings*).



Picture 21: CATIA V5 General→Document Settings

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

Switching to the PWBSchema File suited for BOM Part Structures

The PDM Workbench installation package for the Aras Innovator clients also contains the PDM Workbench Schema file suited to the BOM Part Structure Data Model. It is named “PWBSchema_Aras_PartStruc.xml” and it resides in the “config” directory of the PDM specific installation package **PWBCAD_xx_Aras_xx.zip**, beside the files “PWBSchema.dtd” and “PWBSchema.xml”.

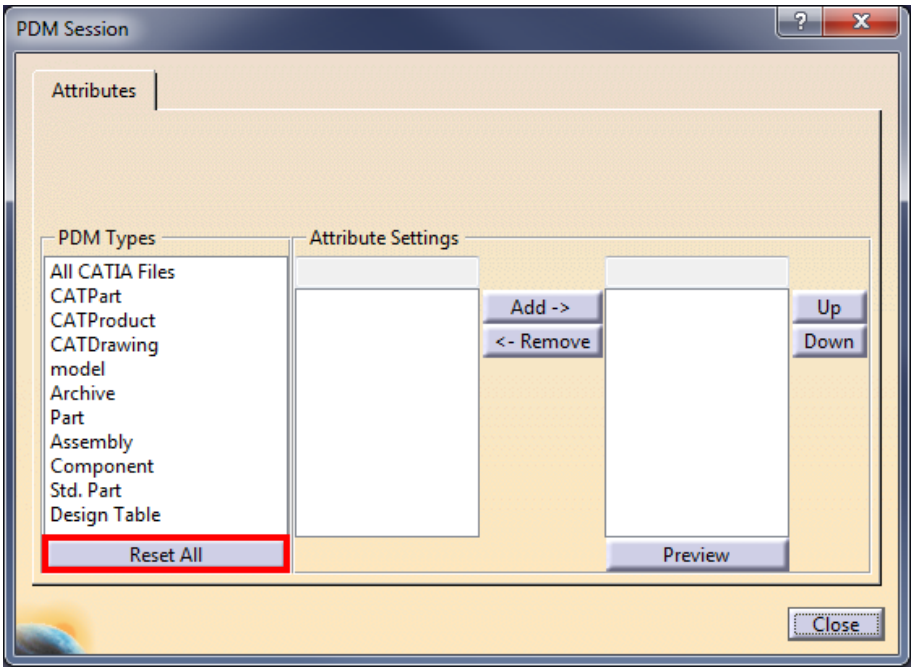
In order to use the BOM Part Structure Data Model please replace the Schema file in the installed PDM Workbench client directory (e.g. “C:\Program Files\T-Systems\PWBCATV5_R28_13.0.0_Aras_1209\config\PWBSchema.xml”) with the other configuration file. Please also make sure that the PDM Workbench environment file “catia_env.bat” contains the correct reference to the Schema file (e.g. “SET PWB_SCHEMA_FILE=%PWBDIR%\config\PWBSchema.xml”).

PWBSchema Modification

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

Please log in into the PDM system.

Choose in CATIA V5 **Tools→Options** and there *General – Compatibility – PDM Workbench*. Click on “Customize List View”. The Configuration dialog opens (see *Picture 22: PDM Session Configuration dialog*).



Picture 22: PDM Session Configuration dialog

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

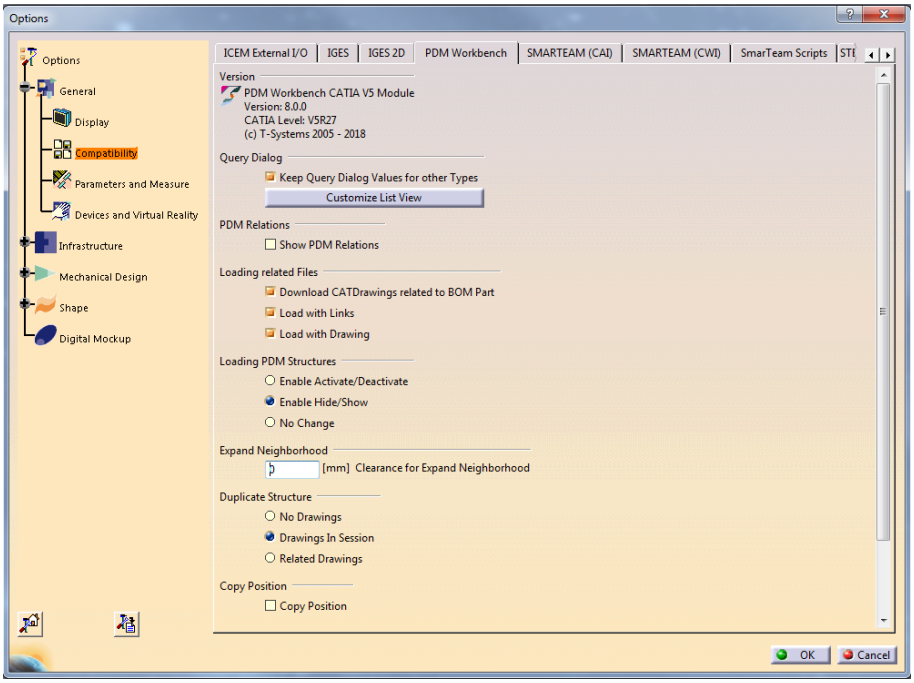
Setting of Environment Variables

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

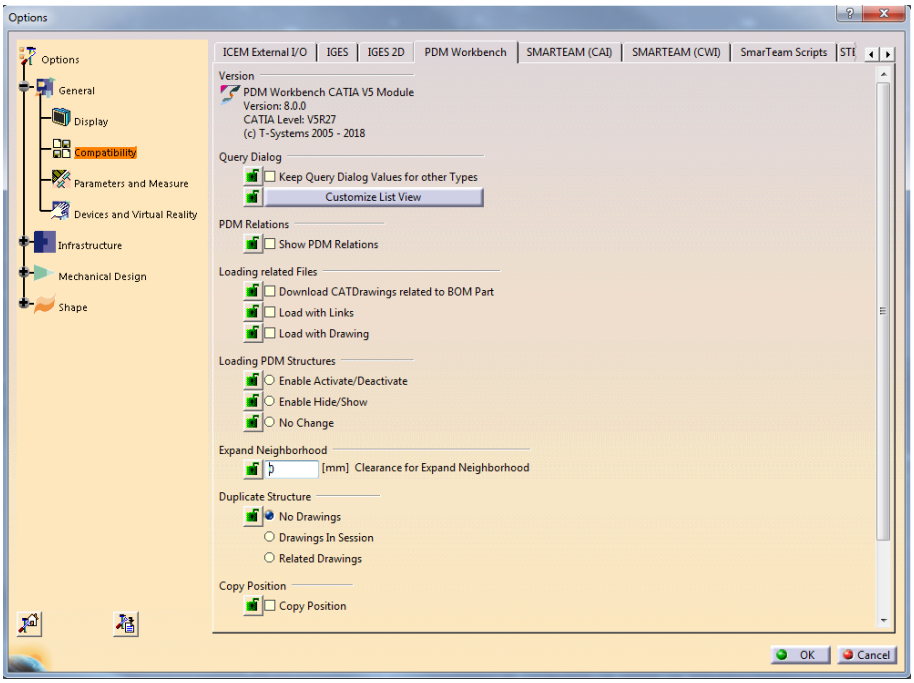
Environment variable	Comment
PWB_XMAP	The location of the exchange map directory. The exchange map directory must be unique for every started CATIA session on the same client.
PWB_SCHEMA_FILE	Path including file name of the XML configuration file.
PWB_SOAP_TARGET_URL	The URL of the web service, e.g. 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.
PWB_ADDTEMP_PREFIX	The prefix for the rename of the Part Numbers and File Names for the “Add Temp” and “Open File Temporary” command. Default value is “TMP”.

Administrative Lock for PDM Workbench Preferences

An administrator can lock the PDM Workbench preferences similar to other CATIA preferences, using the CNEXT –admin interface:



Picture 23: PDM Workbench preferences (administrator view)



Picture 24: PDM Workbench preferences (user view)

The locked preferences will have the administrator's default values and cannot be changed by the users.

Installation without PWB specific Environment Variables

By default, the PWB 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.*`, `PwbArasFileClient.*`, `PwbArasSoapClient.*` and some specific dll files for Aras Innovator) 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 Document 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 Innovator 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 in the same directory as `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 “ToCatia”. It is possible to set the exchange directory in the start script of CATIA V5 like:

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

The “ToCatia” functionality uses the Windows temporary directory for some temporary files.

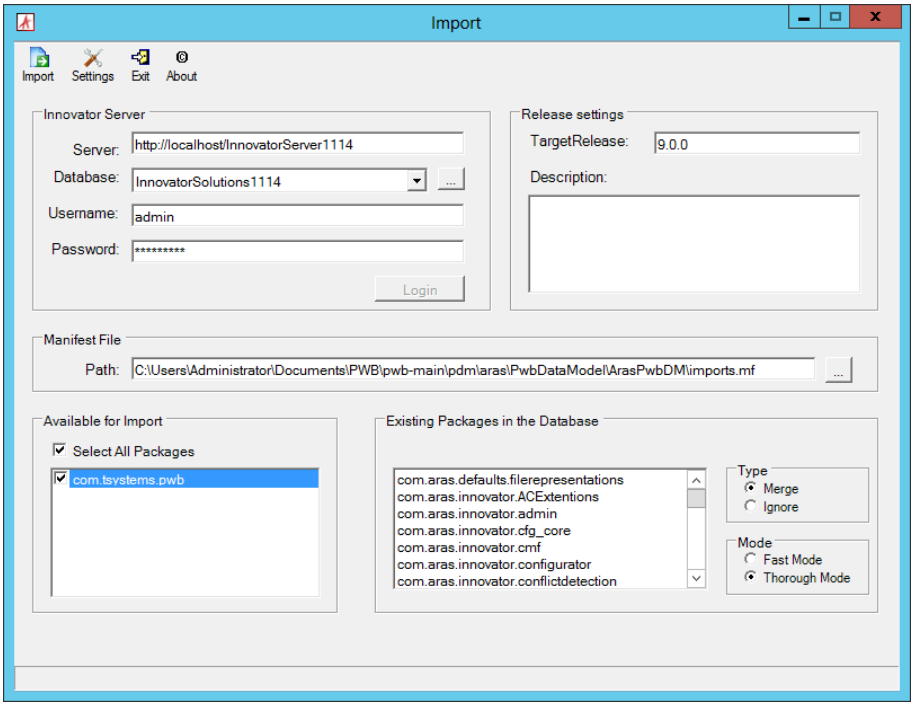
CHAPTER 3

PDM Workbench Data Model

Installation

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

For this the “PwbDataModel_[XX.X].zip” file needs to be unpacked first. Then at least four packages need to be imported to Aras Innovator with the Aras Innovator import utility[1]:



Picture 25: 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 → Downloads and Support → Additional Utilities → Package Import/Export Utilities

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

in the import utility in this order and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 25: PDM Aras Innovator import utility*).

BOM Part Structure Configuration

For the “BOM Part Structure Configuration” functionality select the manifest files

- PwbDataModel_[XX.X]\Specific\StructureConfig\BomConfig\imports.mf
- PwbDataModel_[XX.X]\Specific\StructureConfig\BomConfig_PLM\imports.mf
- PwbDataModel_[XX.X]\Specific\StructureConfig\BomConfig_Core\imports.mf
- PwbDataModel_[XX.X]\Specific\StructureConfig\BomConfig_Cui_default\imports.mf

in the import utility in this order and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 25: PDM Aras Innovator import utility*).

Open in CATIA

For the “Open in CATIA” functionality select the manifest file

- PwbDataModel_[XX.X]\Specific\OpenInCAD\OpenInCATIA\imports.mf

in the import utility and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 25: PDM Aras Innovator import utility*).

If you have installed the “BOM Part Structure Configuration” functionality, you also need to select the manifest file

- PwbDataModel_[XX.X]\Specific\OpenInCAD\OpenInCATIA_BomConfig\imports.mf

in the import utility and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 25: PDM Aras Innovator import utility*).

Synchronize CAD Structure to BOM

For the “Synchronize CAD Structure to BOM” functionality select the manifest file

- PwbDataModel_[XX.X]\Specific\SynchronizeToBOM\imports.mf

in the import utility and perform the import (Type is “Merge”, Mode is “Thorough Mode”) (see *Picture 25: 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

Client Customization

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

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

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

Display Names

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

```
$CATIA_INSTALL_DIR\intel_a\resources\msgcatalog
```

or

```
$CATIA_INSTALL_DIR\win_b64\resources\msgcatalog
```

and the sub directories for the different languages.

There are several NLS files for the dialogs and commands.

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

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

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

- PWBUserErrors.CATNls

The displays in these NLS files can be changed.

Icons

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

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

or

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

The icon names correspond to the names used in the PDM Workbench Schema file.

Data Model Definition

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

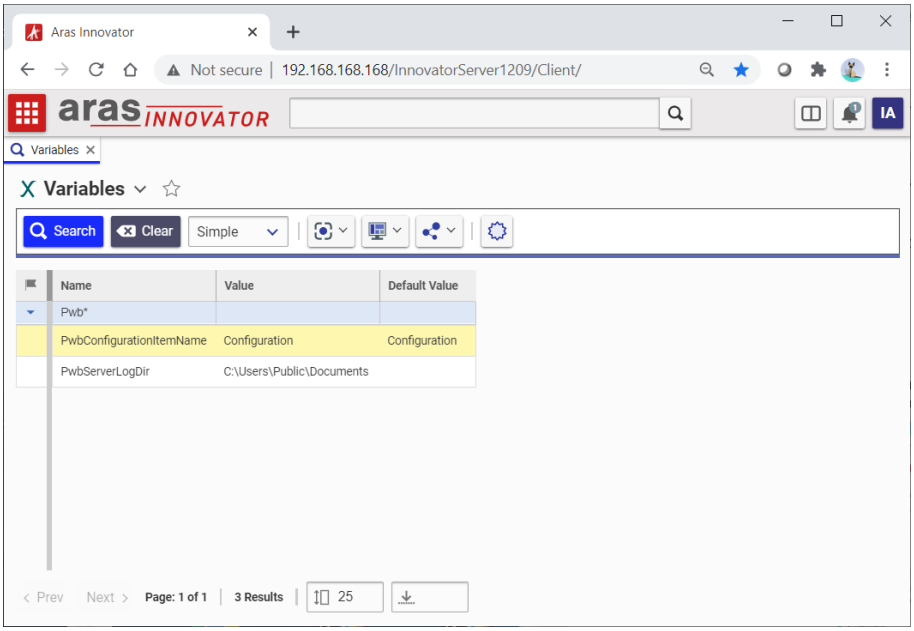
CHAPTER 6

Server Configuration

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

Configuration Variables

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



Picture 26: Aras Innovator server configuration variables

- PwbConfigurationItemName

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

- PwbServerLogDir

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

PWB Logging on Server

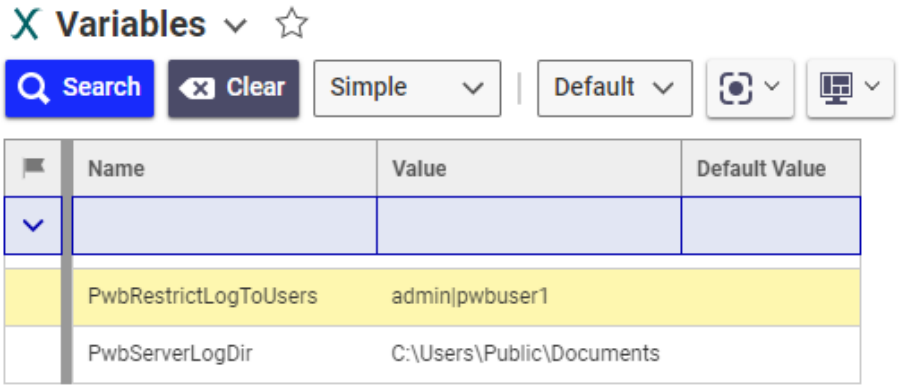
The name of the logfile holds the database name:
`<configured log dir>\PwbServerLog_<database-name>_<login-user>.txt`

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.

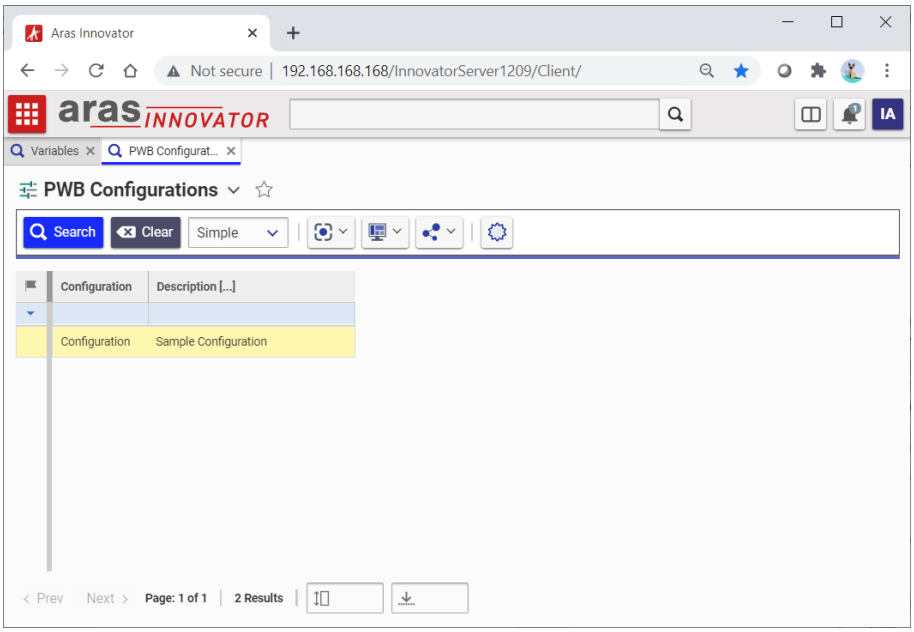
To restrict the logging to a specific set of users, set the Aras Innovator variable “PwbRestrictLogToUsers” to a list of names (separated by “|”).



Picture 27: Logging configuration variables

Configuration Items

In order to define the environment variables and to configure the mapping of attributes between Aras Innovator and CATIA V5 a special PWB Configuration item (see *Picture 28: PWB Configuration item in Aras Innovator*) has to be used:



Picture 28: PWB Configuration item in Aras Innovator

Replacement of Class "ArasUtil" with Class "PwbServerAddin.PwbServerApi" in Custom Server Methods

The usage of the class "ArasUtil" in custom server methods is deprecated and should be replaced with calls to "PwbServerAddin.PwbServerApi". From PDM Workbench versions later than 5.0 on the usage of the class "ArasUtil" will not be possible anymore.

The definition

```
ArasUtil ArasUtilObj = new ArasUtil();
```

should be replaced with

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

The corresponding methods are the same, for the most part, except for two:

"GetValueByName" has been renamed to "GetValueOfVariable", and "HasIdentity" has been renamed to "CurrentUserHasIdentity".

These are the methods of "PwbServerApi":

```
// Converts a string passed to the method to a dictionary.
IDictionary<string, string>
   DialogAttrsStringToDictionary(string ItemXmlString)

// Converts an item passed to the method to a dictionary.
IDictionary<string, string>
   DialogAttrsItemToDictionary(Item DialogItemObj)

// Converts a dictionary that may have been modified in the method back
// to an item, in order to return the information to the connector
// process.
Item DialogAttrsDictionaryToItem(
   IDictionary<string, string> DialogAttrs)

// Retrieves the value of an Innovator variable.
string GetValueOfVariable(string Name)

// Tests whether the user has a specific identity.
bool CurrentUserHasIdentity(string IdentityName)

// Retrieves the number of relationships of a specific type of an item.
int GetRelationshipCount(Item ParentItem, string RelationshipType)

// Retrieves the ID of the thumbnail file of a CAD Document item.
string GetThumbnailFileId(Item ItemObj)

// Retrieves the next string of the sequence.
// Used in the Autoname functionality.
string GetNextAutonameSequence(string SequenceName)

// Converts a string to a boolean value.
bool ParseBoolean(string BoolValue)

// Checks whether a string variable has the null value or is empty.
bool IsNullOrEmpty(string Str)

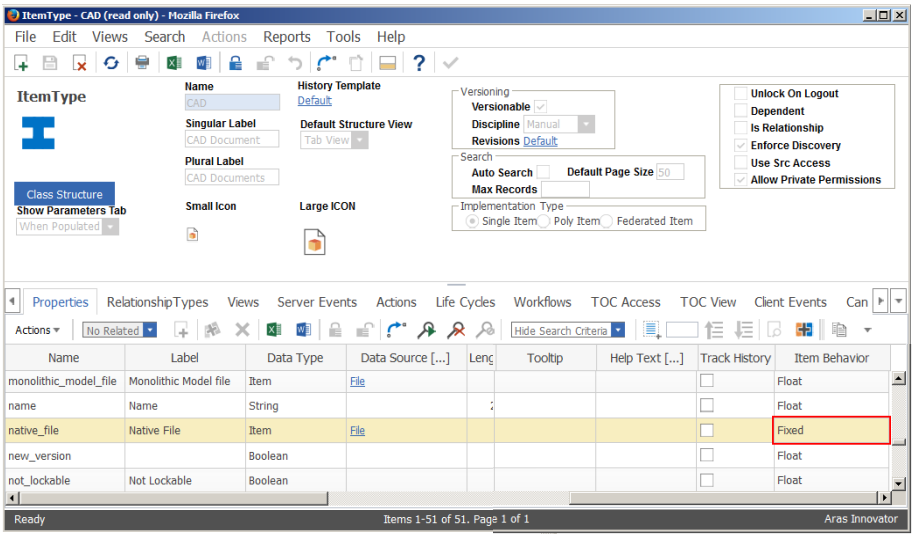
// Throws an exception if an item is null or Item.isError() is true.
void CheckItem(Item ItemObj)

// Throws an exception if an item is null or Item.isError() is true
// and adds a string to the error message.
void CheckItem(Item ItemObjToCheck, string ErrorStringIfFailed)
```

```
// Retrieves the version number of the PWB server API.
string GetApiVersion()
```

Making sure CAD/native_file Definition is "Fixed"

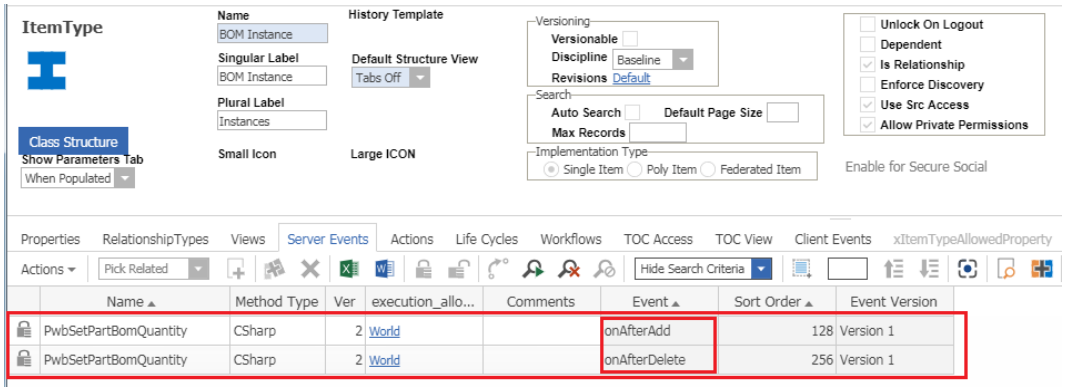
Please set the "Item Behavior" of the property "native_file" for the Item Type "CAD" to "Fixed", if it doesn't already have that value (see *Picture 29: Item Type "CAD"*).



Picture 29: Item Type "CAD"

Automatically updating the "Part BOM" Quantity if Instances are deleted in the Aras Innovator Web Client

For performance reasons the server events for updating the "Part BOM" quantity are not created anymore when the PWB data model is imported. If the quantity needs to be updated when instances are deleted in the Aras Innovator Web Client please add the server events "onAfterAdd" and "onAfterDelete" for the method "PwbSetPartBomQuantity" for the Item Type "BOM Instance".



Picture 30: "BOM Instance" Server Events

CHAPTER 7

Configurations for specific functionalities

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

Standard Configuration

Exchange Map

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

Example:

```
<xmap value="C:\PWB_XMAP" sizeThreshold="0" />
```

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 CATIA V5 environment variable does not exist.

If the attribute *sizeThreshold* exists in the schema and represents a numerical value larger than 0, the PDM Workbench will execute some clean-up actions at the end of the Load and Update commands. It means a size in megabyte. If the sum of the file sizes of all CATIA and jpg files in the exchange map directory is larger than this threshold, then some files, **which are not needed by your current CATIA session**, are automatically deleted - from oldest to newest - until the remaining file size sum is less than the threshold.

If you want to switch off the automatic clean-up actions, you can set the sizeThreshold to “0” or remove the attribute *sizeThreshold* from the PWB xml schema.

Optional.

SOAP target URL

In the Schema file the URL of the server process, that the PDM Workbench 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 CATIA V5 environment variable does not exist.

Optional.

SOAP Client Call Path Environment Variable

In the Schema file the name of the environment variable which contains the path to the Aras Innovator SOAP client executable has to be configured.

Example:

```
<externalSoapClientCallPathEnvVar
   value="PWB_SOAPCLIENT_PATH_ARAS" />
```

This variable is defined in the start script of the PDM Workbench, e.g.

```
SET PWB_SOAPCLIENT_PATH_ARAS=
   %PWBV5DIR%\code\bin\PwbArasSoapClient.exe
```

Mandatory.

File Client Call Path Environment Variable

In the Schema file the name of the environment variable which contains the path to the Aras Innovator file client executable has to be configured.

Example:

```
<externalFileClientCallPathEnvVar
   value="PWB_FILECLIENT_PATH_ARAS" />
```

This variable is defined in the start script of the PDM Workbench, e.g.

```
SET PWB_FILECLIENT_PATH_ARAS=
   %PWBV5DIR%\code\bin\PwbArasFileClient.exe
```

Mandatory.

Maximum Expansion Level

In the Schema file the maximum expansion level can be defined. It describes how many levels, starting from the root node, will be expanded by the PDM Workbench.

Example:

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

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

Optional.

Read only

In the Schema file it can be configured that the loaded CATIA files are set to read-only if the PDM nodes are not modifiable.

Example:

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

Default value: “false”

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

Reload after Update

In the Schema file it can be configured that the CATIA document will be reloaded after the update.

Example:

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

Default value: “false”

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

Allow loading different File Versions

In the Schema file it can be configured that it is allowed to load different file versions in the CATIA session.

Example:

```
<allowLoadingDifferentFileVersion value="false" />
```

Default value: “true”

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

Show out-of-date Link Info

In the Schema file it can be configured that the relation should be marked as outdated if a newer object exists.

Example:

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

Default value: “false”

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

Update after Create new CATIA file

In the Schema file it can be configured that the update should be started after the create of a new CATIA file.

Example:

```
<startUpdateAfterCreateNewCatiaFile value="false" />
```

Default value: “true”

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

Upload changed Files at Sync Failure

In the Schema file it can be configured that the changed files are uploaded even if an error occurs in the update process.

Example:

```
<uploadChangedFilesAtSyncFailure value="false" />
```

Default value: “false”

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

Asynchronous File Upload

It is possible to perform the file upload and the following steps like optional post-processing steps, after the new or modified CATIA files have been saved in the local PWB_XMAP directory, asynchronously. That means that the user can continue to work with CATIA after that step, only the connector functionality is deactivated during this time. This is shown by all the connector icons being deactivated, and the PDM Save icon blinking:



Picture 31: Toolbar during asynchronous file upload

The connector functionality will be active again after the update process has finished:



Picture 32: Toolbar after asynchronous file upload

This functionality can be switched on with the Schema file setting

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

Rollback on File Upload Error

This functionality is only available in CAD Document Structure Data Model.

On some Aras Innovator installations there are occasional problems to upload a file to Aras Innovator. This causes incomplete CAD Documents with missing native files.

If this functionality is activated it will roll back the incomplete CAD Documents in the case of an error.

To enable the feature you have to set “RollbackCadOnError” to “true” in the PWB configuration on Aras Innovator.

RollbackCadOnError	true

Picture 33: Sample RollbackCadOnError configuration

Required Attributes

In the Schema file it can be configured if the wildcard (e.g. “*”) is an invalid value for required attributes.

If this setting is active, it defines values as not valid for required query dialog attributes. For instance, if the item_number attribute is defined as required, then the setting above defines the query value of “*” as invalid for that attribute.

Example:

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

Default value: “false”

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

Maximum Description Attribute Length

In the Schema file the maximum length of the description attribute can be defined.

Example:

```
<maximumDescriptionAttributeLength value="10" />
```

Optional.

Show Data Source NLS Values

In the Schema file it can be configured if the NLS values of the data source should be shown.

Example:

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

Default value: “false”

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

Rename existing Files in Exchange Map

In the Schema file it can be configured if the existing files in the exchange map should be renamed at load.

Example:

```
<renameExistingExMapFilesAtLoad value="false" />
```

Default value: “true”

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

Auto Login on Session expired

In the Schema file it can be configured if an auto login should be performed if the session has been expired.

Example:

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

Default value: “false”

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

Session Settings (contains Setting for Windows Single-Sign-On)

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

Example:

```
<sessionSettings>
   <-- Set one of these two values as default.
       It can be changed in the CATIA session with
       Tools->Options->Compatibility->PDM Workbench. -->
   <queryMode name="listViewWindow" />
   <queryMode name="pwbWindow" />
   <-- Set one of these two values as default.
       It can be changed in the CATIA session with
       Tools->Options->Compatibility->PDM Workbench. -->
   <relationDisplayMode name="relDisplay" />
   <relationDisplayMode name="noRelDisplay" />
   <-- Set one of these three values. -->
   <passwordEncryption name="WinAuth" />
   <passwordEncryption name="MD5" />
   <passwordEncryption name="none" />
</sessionSettings>
```

Optional.

To enable single-sign-on on Windows please change the setting *sessionSettings/passwordEncryption/name* from “MD5” to “WinAuth”.

The precondition is that single-sign-on has been configured correctly in Aras Innovator.

If single-sign-on has been configured then the “PWBLoginUser” and “PWBLoginPassword” definitions should be removed from the login dialog, because they are not needed anymore:

```
<form name="Login" info="ShowOnlyLoginData" >
   <frame displayName="NLS_UserData">
      <pwbFormAttribute name="PWBLoginUser"
                        widgetType="SingleLineEditor"
                        mode="update" visibleLength="10"
                        required="true" entryAllowed="true" />
      <pwbFormAttribute name="PWBLoginPassword"
                        widgetType="SingleLineEditor"
                        mode="update" visibleLength="11"
                        required="false" />
      <formAttribute name="LoginDatabase" widgetType="ComboBox"
                     mode="update" visibleLength="10"
                     required="true" entryAllowed="false" />
   </frame>
</form>
```

Start Login during startup

It is possible to launch the PWB Login process directly during startup of CATIA.

Add the option “-e PWBLoginCmd” to the startup command of CATIA in the file pwb_start.bat:

```
start CNEXT -env pwbcatiaenv.txt -direnv "%PWBDIR%\config" -e PWBLoginCmd
```

Hide login dialog for Single-Sign-On

If single-sign-on has been configured and there is no user entry needed, it is possible to hide the login dialog completely using the following settin in the settings section of the PWB Schema file:

```
<settings>
   …
   <setting name="ShowLoginDialog" value="false" />
   …
</settings>
```

Use Custom Login Token Provider

For single-sign-on, it is possible to use a custom login token provider to supply the access tokens.

Configuration

The custom login token URL has to be configured in the Schema file:

```
<sessionSettings>
   <passwordEncryption name="WinAuth"
               customLoginTokenUrl="http://localhost:5000/token"
/>
</sessionSettings>
```

When using this setting it is also required to copy the DLL Newtonsoft.Json.dll from the Aras Innovator server to the binary path of each client (`<PWB_install_dir>\win_b64\code\bin`).

Language Settings

In the Schema file the name and date format of the installed languages (NLS files) have to be configured.

Example:

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

Mandatory.

Date Format

In the Schema file the definition of the date format for the installed languages has to be stored.

Example:

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

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

Mandatory.

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.

Mark superseded Nodes

In the Schema file it can be configured if superseded nodes should be marked as outdated.

If this setting is set to “true” then superseded nodes (nodes where a newer generation exists) are colored magenta, like outdated nodes, where the CAD file has been modified by another user, instead of black.

Example:

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

Default value: “false”

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

Part Classes

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

Example:

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

Optional.

Select Type of additional Parts in CAD Document Structure Data Model

In the Schema file you can enable the functionality to select the type of additional parts in the CAD Document Structure Data Model.

You have to add the attribute `selectPartClassName="true"` to the *partClasses* XML tag.

Example:

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

Optional.

Toolbar Configuration

In the Schema file the toolbar can be configured. It can be defined which toolbar entries have to be removed.

Example:

```
<removeToolbarIcons>
   <!-- available entries:
        "LocalWorkspace", "Register", "PreProcessCadStructure",
        "Update", "Synchronize", "Refresh", "SetSessionConfig",
        "NewPwbWindow", "About" -->
   <icon name="LocalWorkspace" />
   <icon name="Register" />
   <icon name="PreProcessCadStructure" />
   <icon name="Synchronize" />
   <icon name="NewPwbWindow" />
   <icon name="LocalSession" />
</removeToolbarIcons>
```

Optional.

Customer-Specific Environment

If customers use different CATIA releases, they can configure in the start script of the PDM Workbench which environment should be used. The designer will not be able to modify CAD files which have been created in different customer environments.

The CATIA environment can be defined in the Schema file. This definition can be done by one or more attributes, e.g. customer and project.

For `loadOthersReadOnly="true"`, the user can query for any PDM object. PDM objects which do not fit to the configured environment variables are always loaded in “ReadOnly” mode. For this configuration the settings of `defaultValue` are ignored.

Example:

```
<catiaEnvironment loadOthersReadOnly="true">

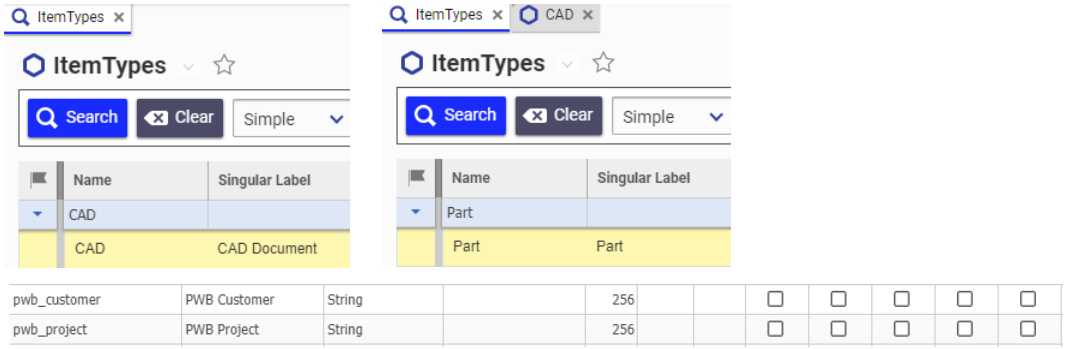
   <envAttribute displayName="NLS_env_customer"
pdm="pwb_customer"
                 env="PWB_ENVIRONMENT_CUSTOMER"
                 defaultValue="Default Customer" />

   <envAttribute displayName="NLS_env_project" pdm="pwb_project"
                 env="PWB_ENVIRONMENT_PROJECT"
                 defaultValue="Default Project"/>

</catiaEnvironment>
```

The values of these environment variables `PWB_ENVIRONMENT_CUSTOMER` and `PWB_ENVIRONMENT_PROJECT` must be defined in the start script of the PDM Workbench.

At the server for the Item Types “CAD” and “Part” the properties `"pwb_customer"` and `"pwb_project"` must be added:

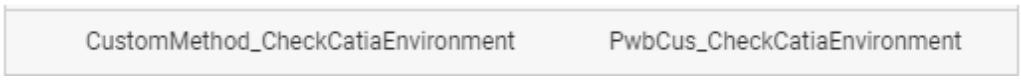


Picture 34: Add project and customer to the Item Types “CAD” and “Part”

Note that the properties `"pwb_customer"` and `"pwb_project"` are only examples and more properties could be added with different names.

With `loadOthersReadOnly="true"` it is possible to use a custom server method to check if an Item fits to the currently used CATIA environment.

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



Picture 35: 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 Case "Query" (loadOthersReadOnly="true")

The user can query for any PDM object.

Use Case "Update"

Whenever the user creates new PDM items the strings from the environment variables are written into the specific PDM item attributes. Only new objects or PDM objects fitting to the configured environment variables can be updated.

Use Case "Load" (loadOthersReadOnly="false")

In the "Load" or "Open" on PDM items a check is performed whether the value string of the specific item attributes is the same as the one read from the corresponding environment variables.

Only if the value is the same the files will be loaded in CATIA, otherwise the user will get an error message that the environment does not fit.

Use Case "Load" (loadOthersReadOnly="true")

The "Load or "Open" will open all files in CATIA. If a PDM object does not fit to the configured environment settings, it is loaded in "ReadOnly" mode.

Data Model Configuration

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

The attribute has to be set to “true” in order to use the BOM Part Structure Data Model. Otherwise the CAD Document Structure Data Model will be used.

UseBomPartStructure	true

Picture 36: Sample UseBomPartStructure configuration

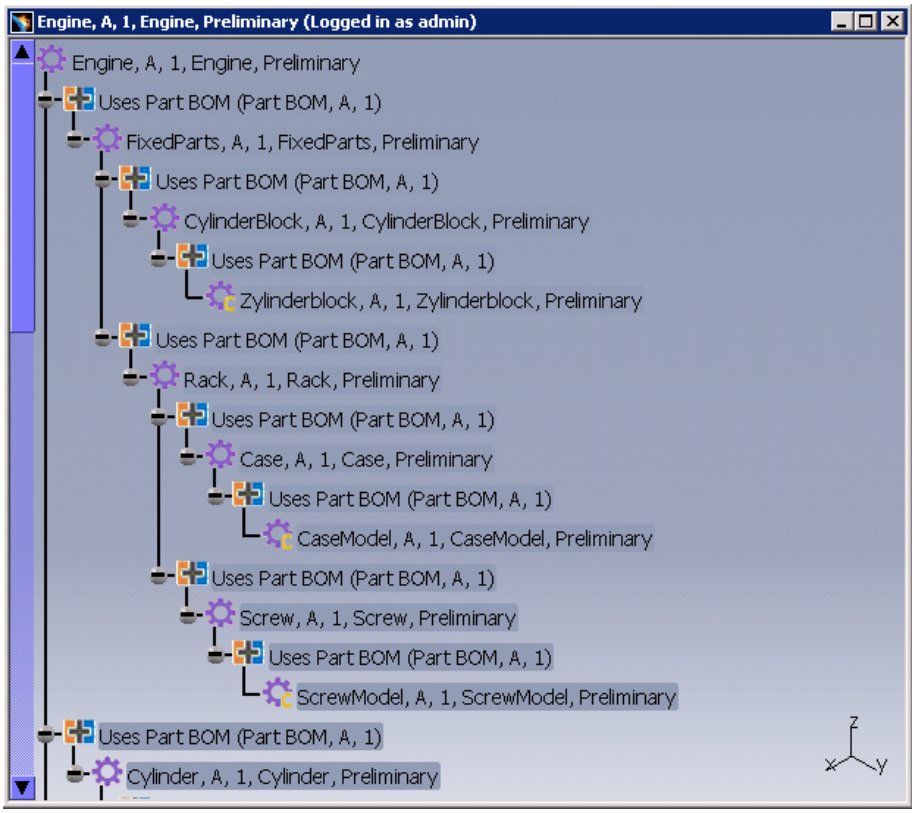
Default value: “false”

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 (Assembly or Component). The relation “Part BOM” is used.

Each Part is described by a CAD Document which includes the CATIA file for a CATProduct, CATPart, or CATDrawing (see *Picture 37: Structure in the BOM Part Structure Data Model*).

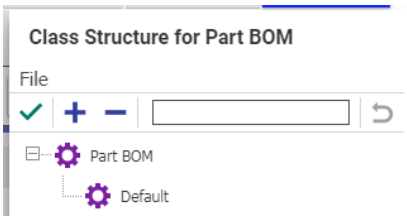


Picture 37: Structure in the BOM Part Structure Data Model

Support Part BOM Classification

It is possible to configure a relation type and classification for the BOM structure.

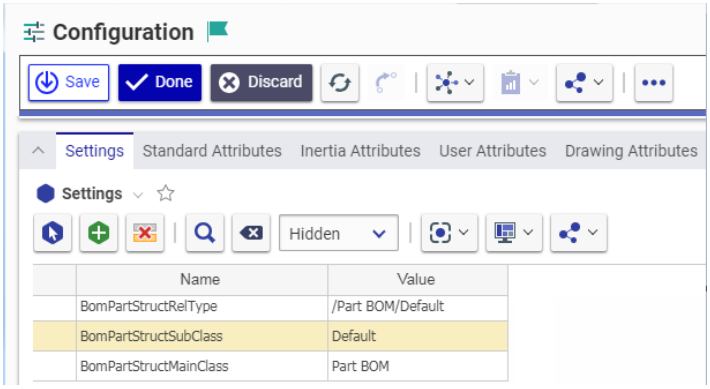
You have to create a special Part BOM type like “/Part BOM/Default” in the class structure of Part BOM:



Picture 38: Class structure of Part BOM

To use the special Part BOM relation “/Part BOM/Default” you have to set the PWB Configurations:

```
BomPartStructRelType    = "/Part BOM/Default"
                          (default value: "Part BOM")
BomPartStructSubClass   = "Default" (default value: "")
BomPartStructMainClass = "Part BOM" (default value: "Part BOM")
```



Picture 39: PWB Configuration – Special Part BOM relation

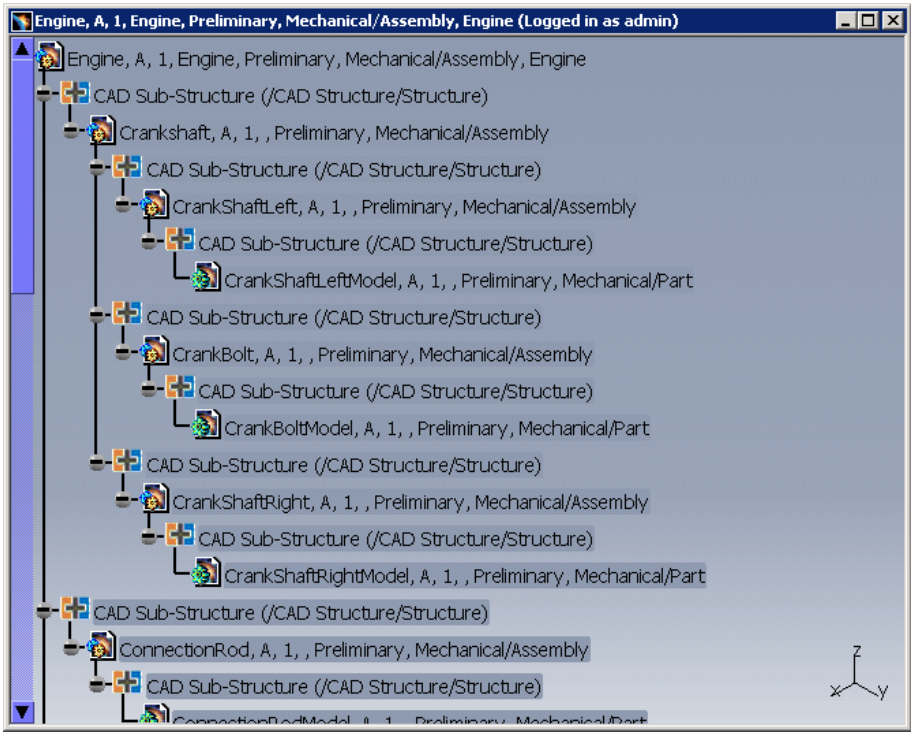
On the client you have to replace all occurrences of “Part BOM” in the Schema file by “/Part BOM/Default”.

Do not forget the relations “Part BOM_LeftToRight” and “Part BOM_RightToLeft:” “Part BOM/Default_LeftToRight”, “Part BOM/Default_RightToLeft”.

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 CATIA file for a CATProduct, CATPart, or CATDrawing (see *Picture 40: Structure in the CAD Document Structure Data Model*).



Picture 40: Structure in the CAD Document Structure Data Model

Client/Server "UseBomPartStructure" Setting Compatibility Check

There is an optional attribute in the PWBSchema tag of the Schema file:
`UseBomPartStructure=true|false`

If it exists then, during login, it is checked against the "UseBomPartStructure" setting on the server. If the values do not fit login is not possible.

Configuration

Here the value is set to "false" in the PWBSchema file:

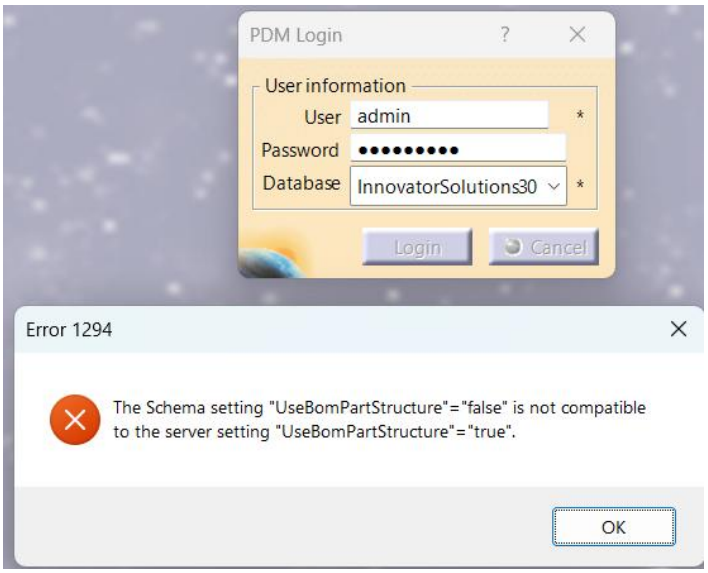
```
<PWBSchemata>

   <PWBSchema system="Aras" customization="Aras" ...
              UseBomPartStructure="false" >

      ...
```

Usage

If the setting on the server is "true" the user will get this error when trying to log in:



Picture 41: Configuration error at login

BOM Part Structure Configuration

The PWB Configuration item setting “BomPartStructureLockUnlockRelatedItem” sets the behavior of “Claim” and “Unclaim”.

Normally the “Claim” and “Unclaim” context menus in the CATIA window claim and unclaim only the CAD Document. Also claiming and unclaiming a part or a CAD Document in the PDM structure window only claims the item where the action was performed on. With this configuration setting this behavior changes such that claiming a part also claims its related CAD Documents, or claiming a CATPart or CATProduct CAD Document claims its related part, or both of these actions are activated. So, for instance, if this setting is set to “part” or to “both” then in the BOM Part Structure Data Model claiming a CAD Document, either explicitly in the PDM structure window, or in the CATIA window, also claims the corresponding Part item.

BomPartStructureLockUnlockRelatedItem	cad

Picture 42: Sample BomPartStructureLockUnlockRelatedItem configuration

Default value: “both”

Optional. Possible values: “part”, “cad”, or “both”.

Different Life Cycle States for CAD and Part with “Promote”

The promotion of the Part and CAD Document items is optionally independent.

The setting “BomPartStructurePromoteRelatedItem”, which has the default value “cad”, can be set to “” (the empty string). With this setting promoting the Part item does not automatically also promote the related “CAD” item.

Query Configuration

Configuring the Size of the "PDM Query" Dialog

In the Schema file it is possible to configure the size of the "PDM Query" dialog and its result column widths.

Left Part of "PDM Query" dialog in Pixel - Right part of "PDM Query" dialog [lines / characters].

Example:

```
<queryDialogDimensions attributeFrameHeight = "375"
                       attributeFrameWidth = "320"
                       resultListHeight = "25"
                       resultListWidth = "60" />
```

Optional.

In the definition of each "PDM Query" dialog the result column width can be defined for each attribute (default value: "15").

Example:

```
<form name="Query">
   <formAttribute name="item_number"
                  ...
                  resultColumnWidth = "15" />
   ...
</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 43: Sample QueryOrderByAttribute configuration

Optional.

MaxQueryResults

The PWB Configuration item setting "MaxQueryResults" defines the maximum number of items that are retrieved in a single query. If more items exist the user is informed about that fact.

MaxQueryResults	100

Picture 44: Sample MaxQueryResults configuration

Optional.

Default Sort Criteria for Query Results

In the Schema file for every PDM Item Type an attribute can be configured to be the default sort order attribute. Query results for that Item Type are automatically sorted by that attribute.

The “object” definition can contain an optional XML attribute: “defaultSortAttribute”.

Example:

```
<object name="Part" displayName="NLS_Part"
        icon="Aras_Part" defaultSortAttribute="item_number" >
```

This attribute has to contain the internal name (not the CATNls name) of an attribute of the Item Type, e.g. “item_number”. The query result will be sorted by the values of that attribute.

Display Custom Query Buttons

In the Schema file the layout of the “PDM Query” dialog can be configured.

If the value is set to “true” the buttons will be displayed on the left side of the bottom of the dialog. In the default layout (value is “false”) the buttons are on the right side of the dialog.

Example:

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

Default value: “false”

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

Modal Dialog

In the Schema file it can be configured that the “PDM Query” dialog is modal. In this case the dialog has to be closed in order to continue working with the PDM Workbench.

Example:

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

Default value: “false”

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

Check if no Query Attribute filled

In the Schema file it can be configured if a query with empty attributes should be possible.

If the value is set to “false” it is possible to perform a query without filling an attribute in the “PDM Query” dialog.

If the value is set to “true” it is necessary to fill at least one of the attributes in the “PDM Query” dialog.

Example:

```
<checkIfNoQueryAttrsFilled value="false" />
```

Default value: “true”

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

Display String in Query List View

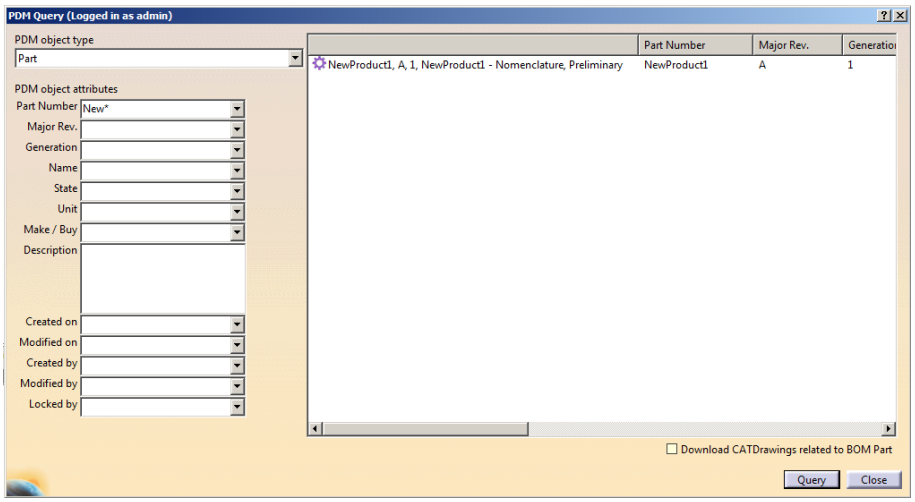
In the Schema file it can be configured if the display string of the object should be shown after the icon in the query list view.

Example:

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

Default value: “false”

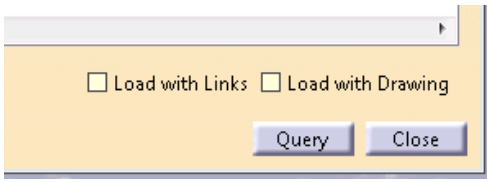
Optional. Possible values: “true”, or “false”.



Picture 45: Query list view in “PDM Query” dialog with Display String

Automatically loading CATDrawings or linked CATParts

It is possible to automatically load linked CATDrawings or CATParts when a CATPart or a CATProduct structure is loaded. There are two check boxes on the “PDM Query” dialog where this behavior can be set.



Picture 46: “Load with Links” and “Load with Drawings” check boxes

The check boxes have to be enabled to be displayed:

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

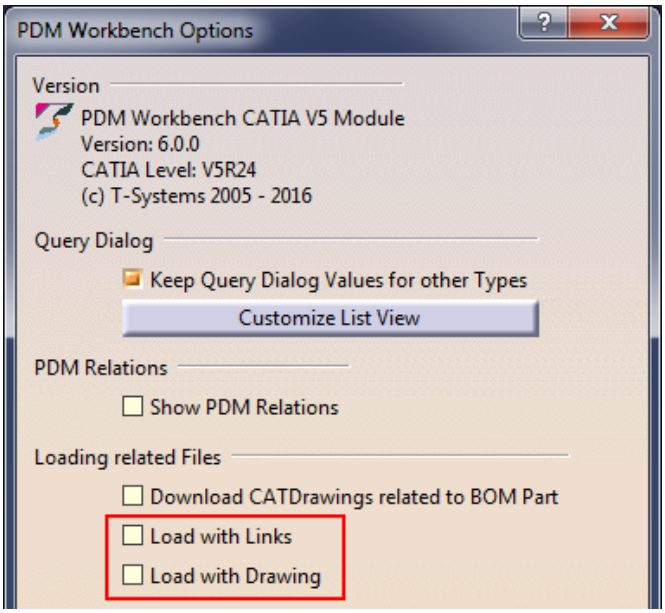
```
<showLoadWithDrawingInQuery value="true" />
```

As default they are not displayed.

Default value: “false”

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

The default value of the check boxes can be set in the “PDM Workbench Options” dialog:

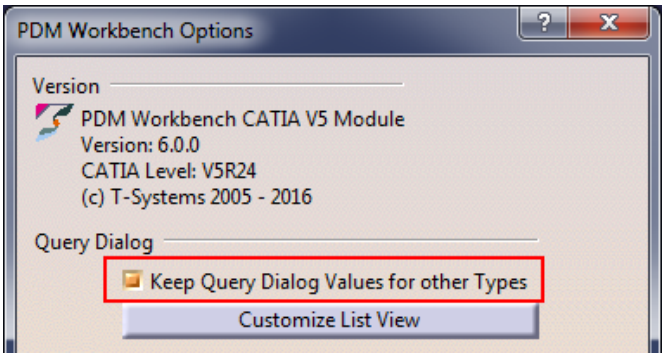


Picture 47: "PDM Workbench Options" dialog with highlighted check boxes

Keep Filter Attribute Values when changing the Type

If the value of an attribute is set, and if the new type has an attribute with the same name, then the value is kept for the new type.

If the old behavior is wanted (not keeping the attribute values) then the check box "Keep Query Dialog Values for other Types" has to be un-checked in the "PDM Workbench Options" dialog:



Picture 48: "PDM Query" dialog setting in "PDM Workbench Options" dialog

"Select Date" Enhancement in the "PDM Query" Dialog

It can be complicated to write a date in the correct format into the date fields of the "PDM Query" dialog. With this functionality you can select the dates from a calendar widget.

In the Schema file the widget type of the attributes which represent a date, like "created_on" or "modified_on", can be changed to "ComboBoxDate":

```
<form name="Query">

   ...

   <formAttribute name="created_on"
                  widgetType="ComboBoxDate" mode="update"
                  visibleLength="15" required="false"
                  listViewRelevant="true"
                  entryAllowed="true"
                  dataSource="RecentlyUsedValueDataSource" />
```

```
    <formAttribute name="modified_on"
                   widgetType="ComboBoxDate" mode="update"
                   visibleLength="15" required="false"
                   listViewRelevant="true"
                   entryAllowed="true"
                   dataSource="RecentlyUsedValueDataSource" />

    ...
</form>
```

If this functionality is used, a server method which pre-processes the “PDM Query” dialog attribute values has to be defined and configured in the settings of the PWB Configuration item.

For example, if the server method is named “PwbCus_PreProcQryDlgAttrs”, then a setting with the name “CustomMethod_PreProcQryDlgAttrs” has to be defined, and it has to contain the name of the server method as the value:

CustomMethod_PreProcQryDlgAttrs	PwbCus_PreProcQryDlgAttrs

Picture 49: Sample CustomMethod_PreProcQryDlgAttrs configuration

This is the code of that method which processes the new date widget functionality:

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

   string PdmType = this.getProperty("Type");
   string PdmClassification = this.getProperty("Classification");

   // Get all property names and data_types of that ItemType
   IDictionary<string, string> PropertyDict = null;
   PropertyDict = GetAllPropertyNamesTypes(InnovatorObj, PdmType);
   if (null == PropertyDict)
   {
      return InnovatorObj.newError("Could not retrieve properties of ItemType " +
PdmType);
   }

   // Get the query dialog attributes
   IDictionary<string, string> InputDialogDict = null;
   Item InputDialogItem = this.getPropertyItem("QueryDialogAttrs");
   if (InputDialogItem == null)
   {
      return InnovatorObj.newError("Could not retrieve query dialog attributes !");
   }
   else
   {
      InputDialogDict = PwbServerApiObj.DialogAttrsItemToDictionary(InputDialogItem);
      if (InputDialogDict == null)
      {
         return InnovatorObj.newError("Could not convert query dialog attributes !");
      }
   }

   // Define the resulting output dialog attributes
   IDictionary<string, string> OutputDialogDict = new Dictionary<string, string>();

   // Only adding to the output dialog if the attribute value has been changed
   foreach (KeyValuePair<string, string> Attribute in InputDialogDict)
   {
      string AttributeType = PropertyDict[Attribute.Key];

      if (AttributeType.Equals("date"))
      {
         // Check and convert date property values, if necessary
         // Example: Replace a '/' character by the '-' character)

         // Parse for specific syntax
         string ErrorString =
```

```
            "Only  \"DATE\",  \"< DATE\",  \"> DATE\",  \"DATE < DATE\",  or  \"DATE <<
DATE\"  " +
            "is allowed, where DATE can be a string like \"2017-11-03\".";

         string StringFromDialog = Attribute.Value;
         string[] DateValueArray = StringFromDialog.Split(' ');

         if ((DateValueArray.Length > 3) || (DateValueArray.Length < 1))
         {
            throw new Exception(ErrorString);
         }

         if (DateValueArray.Length == 1)
         {
            string AttrValAfterReplacement = Attribute.Value.Replace('/','-');

            if (AttrValAfterReplacement != Attribute.Value)
            {
               string AmlAttrValAfterReplacement =
                  PwbServerApiObj.ReturnRegularAttrQueryString(
                     "", Attribute.Key, AttrValAfterReplacement);

               OutputDialogDict.Add(Attribute.Key, AmlAttrValAfterReplacement);
            }
         }
         else if (DateValueArray.Length == 2)
         {
            // AML for 'later than' and 'earlier than'
            // AmlString += "<created_on condition=\"gt\">2016-10-
11T00:00:00</created_on>";
            // AmlString += "<created_on condition=\"lt\">2016-10-
11T00:00:00</created_on>";

            string Sign = DateValueArray[0];
            if (!((Sign == "<") || (Sign == ">")))
            {
               throw new Exception(ErrorString);
            }

            string DateStr = DateValueArray[1];
            string DateStrAfterReplacement = DateStr.Replace('/','-');

            if (Sign == "<")
            {
               if (-1 == DateStrAfterReplacement.IndexOf('T') )
               {
                  DateStrAfterReplacement+="T23:59:59";
               }

               string AmlAttrValAfterReplacement =
                  PwbServerApiObj.ReturnLessThanAttrQueryString(
                     "", Attribute.Key, DateStrAfterReplacement);

               OutputDialogDict.Add(Attribute.Key, AmlAttrValAfterReplacement);
            }
            else // ">"
            {
               string AmlAttrValAfterReplacement =
                  PwbServerApiObj.ReturnMoreThanAttrQueryString(
                     "", Attribute.Key, DateStrAfterReplacement);

               OutputDialogDict.Add(Attribute.Key, AmlAttrValAfterReplacement);
            }
         }
         else // Length == 3
         {
            // AML for 'between two dates'
            // AmlString += "<created_on condition=\"between\">2016-09-27T00:00:00 and
2016-11-17T00:00:00</created_on>";

            string Sign = DateValueArray[1];
            if (!((Sign == "<") || (Sign == "<<")))
            {
               throw new Exception(ErrorString);
            }
```

```
            string DateStr1 = DateValueArray[0];
            string DateStrAfterReplacement1 = DateStr1.Replace('/','-');

            string DateStr2 = DateValueArray[2];
            string DateStrAfterReplacement2 = DateStr2.Replace('/','-');
            if (-1 == DateStrAfterReplacement2.IndexOf('T') )
            {
               DateStrAfterReplacement2+="T23:59:59";
            }

            string AmlAttrValAfterReplacement =
               PwbServerApiObj.ReturnBetweenAttrQueryString(
                  "", Attribute.Key, DateStrAfterReplacement1,
DateStrAfterReplacement2);

            OutputDialogDict.Add(Attribute.Key, AmlAttrValAfterReplacement);
         }
      }
      else
      {
         // Only add the attribute if it is a date,
         // and if it has been changed during the processing
      }
   }

   Item OutputDialogItem =
PwbServerApiObj.DialogAttrsDictionaryToItem(OutputDialogDict);

   return OutputDialogItem;
}


// Get the name and data_type of all properties of an ItemType
private IDictionary<string, string> GetAllPropertyNamesTypes(Innovator Inn, string
ItemType)
{
   IDictionary<string, string> PropertyDict = null;

   string AML = "<AML><Item action='get' type='Itemtype' select='name'><name>"
              + ItemType + "</name><Relationships>"
              + "<Item action='get' type='Property' select='name,data_type'></Item>"
              + "</Relationships></Item></AML>";

   Item ItemTypeItem = Inn.newItem();
   ItemTypeItem.loadAML(AML);
   Item ResultItem = ItemTypeItem.apply();

   Item PropertyItems = ResultItem.getRelationships();

   PropertyDict = new Dictionary<string, string>();
   int count = PropertyItems.getItemCount();
   int i;
   for (i=0; i<count; i++)
   {
      Item PropertyItem = PropertyItems.getItemByIndex(i);
      string PropertyName = PropertyItem.getProperty("name");
      string PropertyDataType = PropertyItem.getProperty("data_type");
      PropertyDict.Add(PropertyName, PropertyDataType);
   }

return PropertyDict;
```

Special Context Menu for Multi-Select in Query Window

If multiple lines are selected the context menu contains the common subset of the node types.

Update Configuration

Optionally only showing the "PDM Update" Dialog when new Files exist

Can be switched on by the setting in the Schema file:

```
<setting name="ShowUpdateDialogOnlyIfNeeded" value="true"/>
```

"PDM Create" Dialogs

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

ShowCreateDialogsDuringUpdate	true

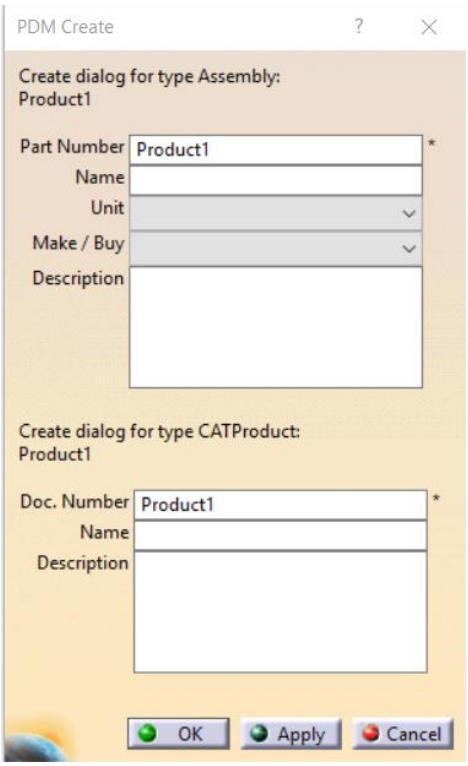
Picture 50: Sample ShowCreateDialogsDuringUpdate configuration

Default value: "true"

Optional. Possible values: "true", or "false".

Show "PDM Create" Dialogs with Multi Column

If a Part and a CAD Document are created together, the "PDM Create" dialog shows the combined dialogues of Part and CAD Document. The Part dialog above and the CAD Document dialog below.



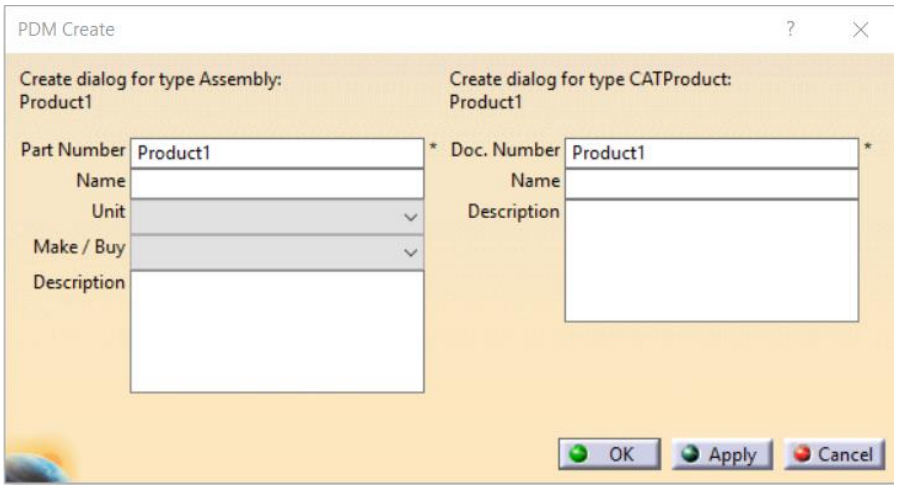
Picture 51: Combined "PDM Create" dialog, with the Part dialog above and the CAD Document dialog below

To show the "PDM Create" dialog in two columns, with the Part dialog on the left and the CAD Document dialog on the right, the schema setting "showCreateDialogsMultiColumn" has to be enabled:

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

Default value: “false”

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



Picture 52: Combined “PDM Create” dialog, with the Part dialog on the left and the CAD Document dialog on the right

“Create new file versions at update?” Button

In the Schema file it can be configured if the “Create new file versions at update?” button appears in the “Update to PDM” dialog.

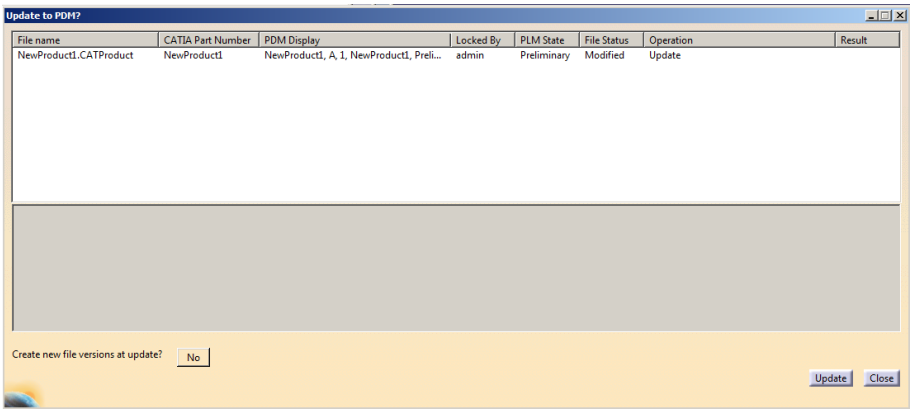
If the value of this setting is set to “false” the button does not appear in the “Update to PDM” dialog. The default value (overwriting the existing document) will always be active.

Example:

```
<showCreateVersionAtUpdate value="false" />
```

Default value: “true”

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



Picture 53: “Update to PDM” dialog with “Create new file versions at update?”

Default create Version Value

In the Schema file the default value of the “Create new file versions at update?” button can be configured.

If the value of this setting is set to “false” the default value of the button is “No”.

If the value of this setting is set to “true” the default value of the button is “Yes”.

Example:

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

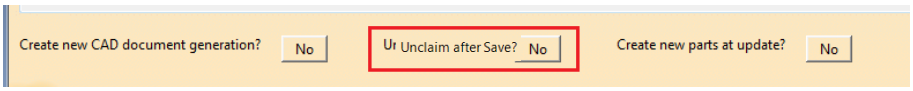
Default value: “false”

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

“Unclaim after Save?” Button

It is possible to automatically unclaim CAD Documents after they have been updated in the PDM update process.

The user has to click on the “Unclaim after Save?” button to change the value to “Yes”.



Picture 54: Optional “Unclaim after Save” button

In the Schema file the setting “showUnlockAfterUpdate” has to be set to “true” to show the button in the “Update to PDM” dialog:

Example:

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

Default value: “false”

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

Show create Parts

In the Schema file it can be configured if the “Create new parts at update?” button appears in the “Update to PDM” dialog.

If the value of this setting is set to “false” the button does not appear in the “Update to PDM” dialog.

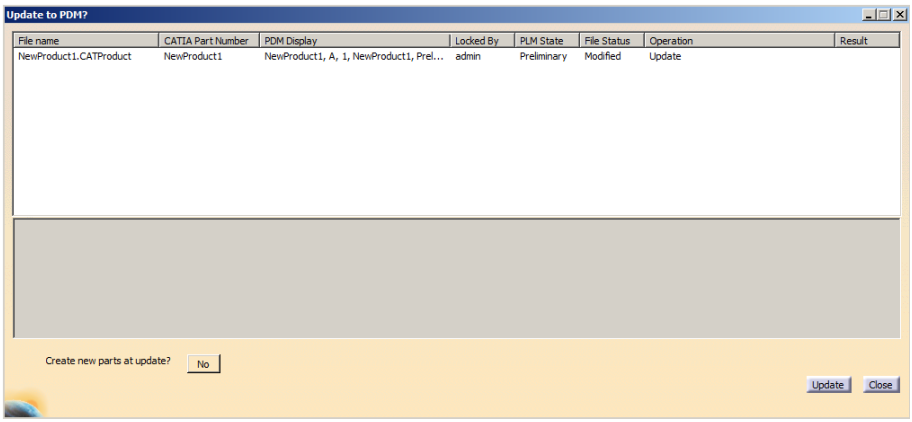
If the value of this setting is set to “true” the button appears in the “Update to PDM” dialog.

Example:

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

Default value: “false”

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



Picture 55: “Update to PDM” dialog with “Create new parts at update?”

Default create Parts Value

In the Schema file the default value of the “Create new parts at update?” button can be configured.

If the value of this setting is set to “false” the default value of the button is “No”.

If the value of this setting is set to “true” the default value of the button is “Yes”.

Example:

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

Default value: “false”

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

Validate Structure before Update

It is possible to have a CAD structure validated by a server method before the update process is started.

Configuration

There is a new toolbar icon which can be activated by commenting out the line ‘`removeToolbarIcons’/’<icon name="PreProcessCadStructure" />`’ in the Schema file:



```
<removeToolbarIcons>
   <icon name="LocalWorkspace" />
   <icon name="Register" />
   <!-- <icon name="PreProcessCadStructure" /> -->
   <icon name="Synchronize" />
   <icon name="NewPwbWindow" />
   <icon name="SetSessionConfig" />
</removeToolbarIcons>
```

Additionally, the server setting “CustomMethod_PreProcessCadStructure” has to be set to the name of a C# server method, for example “PwbCus_PreProcessCadStructure”:

CustomMethod_PreProcessCadStructure	PwbCus_PreProcessCadStructure

Picture 56: Sample CustomMethod_PreProcessCadStructure configuration

It is possible to have the Validate / Pre-process method be automatically called before update. In that case the server setting “PreProcessCadStructureBeforeUpdate” has to be set to “true”:

PreProcessCadStructureBeforeUpdate	true

Picture 57: Sample PreProcessCadStructureBeforeUpdate configuration

Usage

An example implementation of the validation or pre-process method for this, “Pwb_Sample_PreProcCadStructure”, exists.

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
```

Add newly created and updated Part or CAD Document Items to existing Items

It is possible to link the Part and CAD Document 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 Document 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:



Picture 58: 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 to PDM” 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 Document 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
```

A sample method for this, “Pwb_Sample_PostProcUpdateInfo”, exists.

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 Document 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 59: 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**>

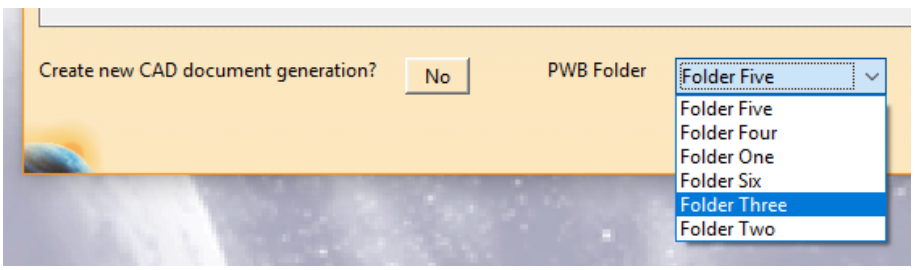
<**NewPartIds**>DD0C0CB5CDB74BE08463C21CA01D1D7F|B1AF4EBA36A447A8B820EEE811BB0890|541AD2AB2F7249AFA46D91D7B75D7AAC</**NewPartIds**>

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

A sample method for this, “Pwb_Sample_ PostProcUpdate”, which relates the root Part item of the updated structure to the selected folder item, exists.

Usage

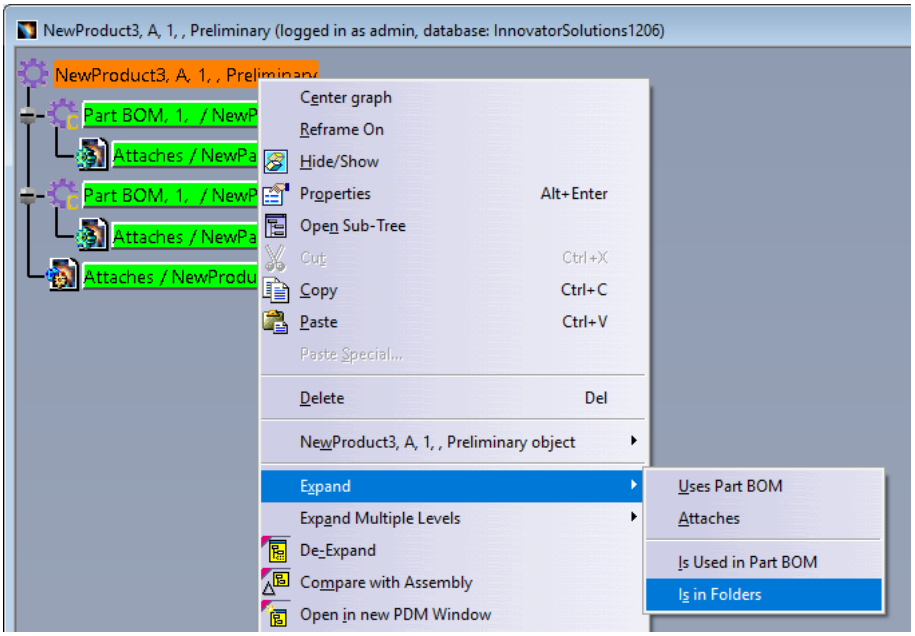
With the correct configuration the user can select an entry from a list of folder names in the “Update to PDM” dialog. The list is the one returned by the custom method defined by the setting “CustomMethod_PostProcUpdate”:



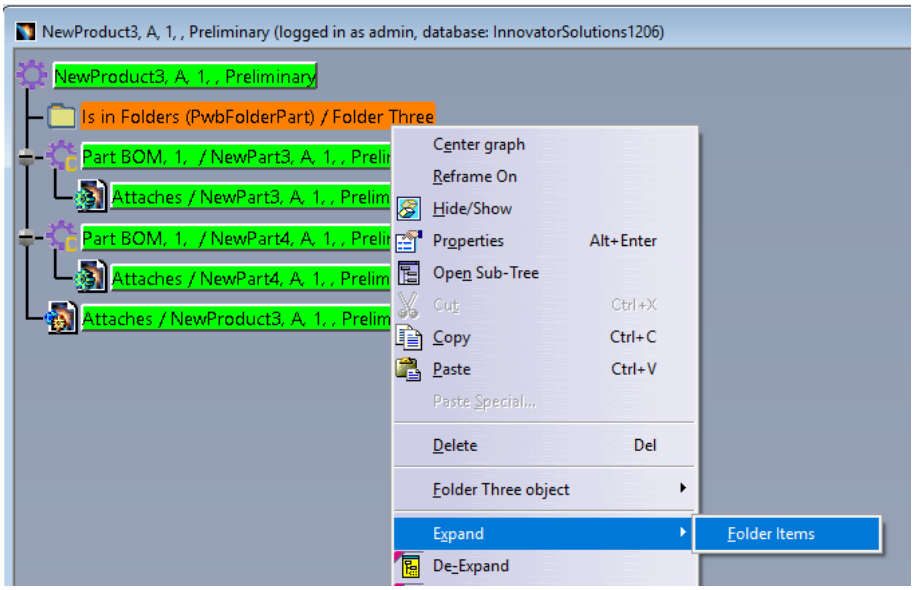
Picture 60: Folder list

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

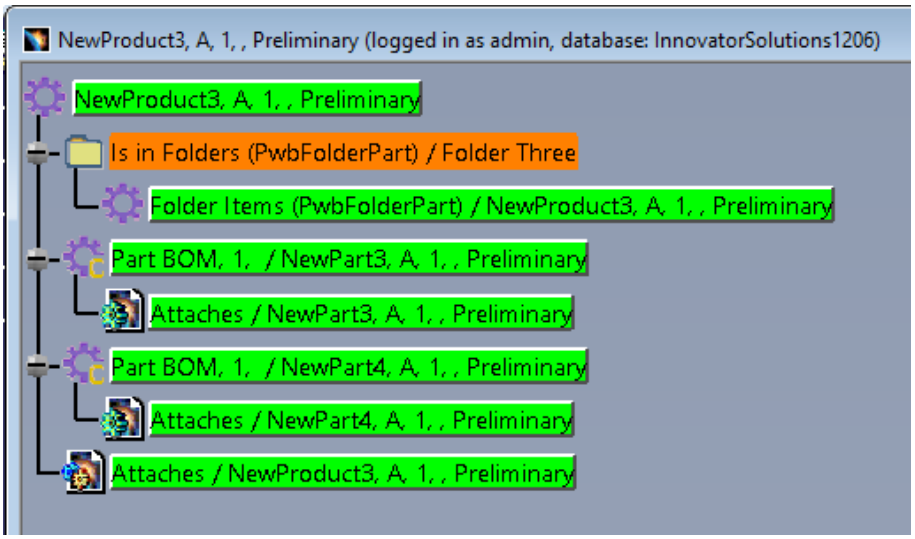
If the folder item and relationship is defined in the Schema file the user can expand the relations in the PDM structure window and find out which folder, if any, a Part item is related to, and which Part items are related to a folder:



Picture 61: Expanding "Is in Folders" in the PDM structure window



Picture 62: Expanding "Folder Items" in the PDM structure window



Picture 63: Expanded folder items in the PDM structure window

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.

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 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 “PDM Create” Dialog

Depending on the attribute mapping, many attributes in the “PDM 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 “PDM 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>
```

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" ... />

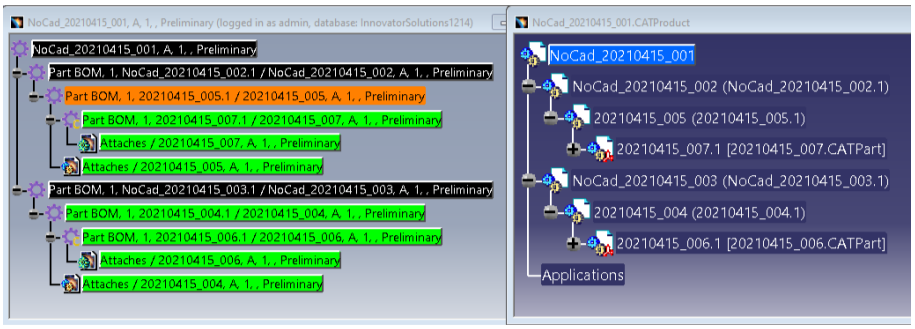
   <formAttribute name="is_standard" widgetType="SingleCheckBox" ... />

</form>
```

Deny create of CAD at top level Structure in BOM Part Structure Data Model

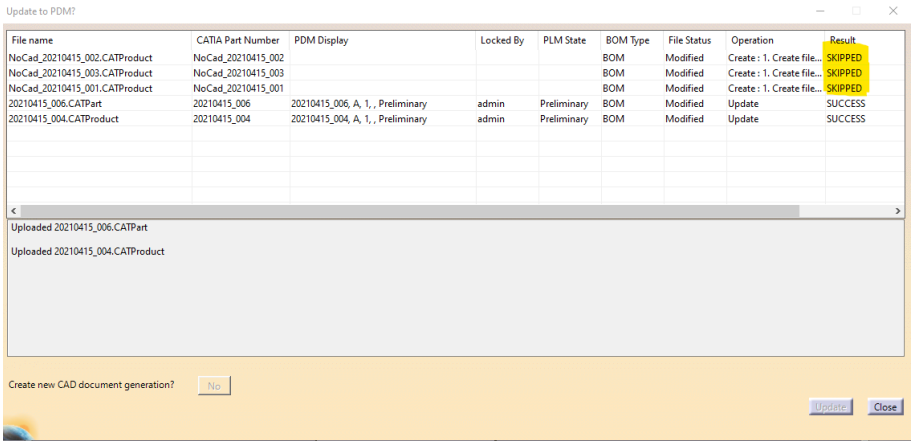
It is possible to create a top level Part structure in Aras Innovator without attached CAD Documents. The structure can be loaded to CATIA, the missing CATIA files are created “On the fly” in CATIA.

This functionality allows to prevent the update of the top level Part structure, including the save of the “On the fly” created CATIA files.



Picture 64: Non CAD top level structure with on the fly created CATProducts

During update, the “On the fly” created CATProducts are skipped.



Picture 65: Update Non-CAD top level structure -> Result SKIPPED

Configuration

Client:

Remove the following line in the Schema file:

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

It is the default behavior now and cannot be configured anymore.

Server:

A custom method has to be created to determine if an item can be updated in CATIA. The method name has to be set to the PWB Configuration setting:

CustomMethod_CheckCatiaAllowUpdate=<Method name>

Sample Custom method:

```
// Sample CustomMethod_CheckCatiaAllowUpdate
//
// This method is called before an Item is send to CATIA.
// If not, the Item can neigther updated nor claimed in CATIA.

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

var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this);

Item ItemToCheck = this.getPropertyItem("ItemToCheck");

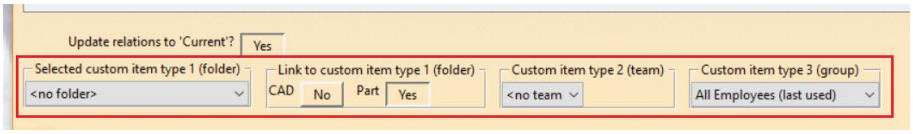
if (ItemToCheck.getType()!= "Part")
{
    // ignore CATIA environment settings
    result.setAttribute("denyUpdate", "false");
    return result;
}

string AttrValue = ItemToCheck.getProperty("item_number", "");
if (0 == AttrValue.IndexOf("NoCad", 0, 5))
{
    result.setAttribute("denyUpdate", "true");
    return result;
}

result.setAttribute("denyUpdate", "false");
return result;
```

Custom Row in "Update to PDM" Dialog

There is an optional row for customer-specific widgets in the "Update to PDM" dialog:



Picture 66: Customer-specific row in "Update to PDM" dialog

Configuration

The widgets in the custom row are configured in the Schema file.

Two widget types are possible: "Button" for check buttons (values checked or unchecked) and "ComboBox" for dropdown lists.

```
<form name="UpdDlgCustomFrame" info="visible=false|title=false" >
   <frame name="SelectedFolderFrame" info="visible=true|title=true" >
      <cusAttr name="SelectedFolder" widgetType="ComboBox"
               visibleLength="12" />
   </frame>
   <frame name="LinkToFolderFrame" info="visible=true|title=true" >
      <cusAttr name="LinkCad" displayName="NLS_LinkCad"
               widgetType="Button" />
      <cusAttr name="LinkPart" displayName="NLS_LinkPart"
               widgetType="Button" />
   </frame>
   <frame name="CustomTeamFrame" info="visible=true|title=true" >
      <cusAttr name="CustomTeam" widgetType="ComboBox"
               visibleLength="12" />
   </frame>
   <frame name="CustomGroupFrame" info="visible=true|title=true" >
      <cusAttr name="CustomGroup" widgetType="ComboBox"
               visibleLength="12" />
   </frame>
</form>
```

The NLS strings of the widgets are defined in the file "PWBSchemaDisplayNames_Aras_Aras.CATNls":

```
UpdDlgCustomFrame = "Custom Frame";
SelectedFolderFrame = "Selected custom item type 1 (folder)";
LinkToFolderFrame = " Link to custom item type 1 (folder)";
NLS_LinkCad = "CAD";
NLS_LinkPart = "Part";
CustomTeamFrame = " Custom item type 2 (team)";
CustomGroupFrame = " Custom item type 3 (group)";
```

The values of the lists are retrieved from the server method configured as "CustomMethod_PostProcUpdateInfo" every time the main "Update to PDM" dialog is created.

If a custom method corresponding to the setting "CustomMethod_CheckPostProcUpdate" exists, then this method is called before the update process starts. This method can be used to return an error in the case of invalid custom parameter combinations.

The selected custom widget values are sent to the method configured by the setting “CustomMethod_PostProcUpdate” which is called at the end of the update process.

A sample method for this, “Pwb_Sample_PostProcUpdateInfo”, exists.

Optional Refresh before Update

It is possible to always automatically refresh the modified CAD Documents before the PDM update process.

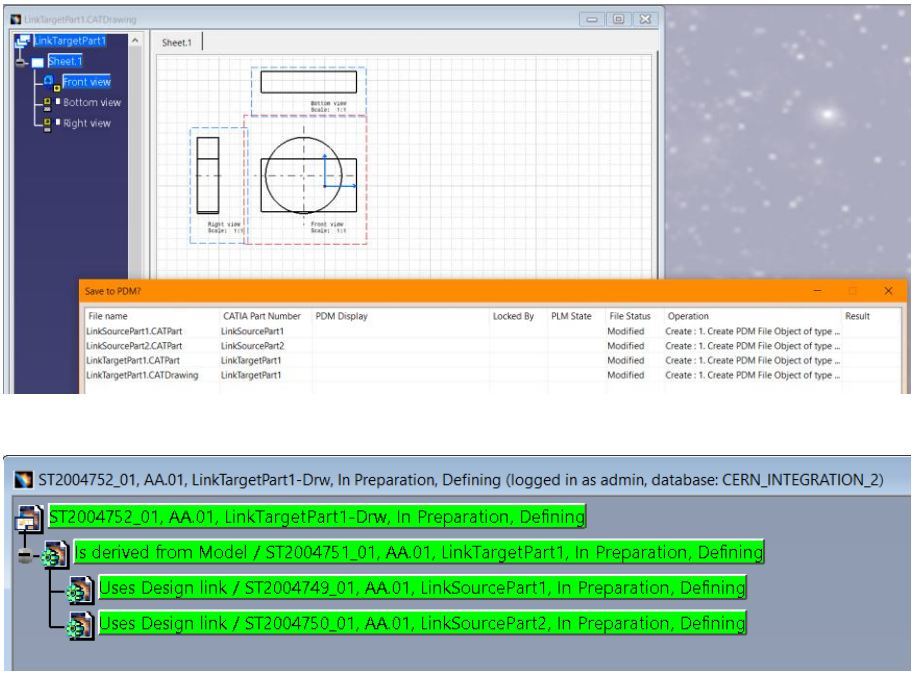
Configuration

Configure in the Schema file:

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

Recursive Save of linked Documents

It is possible to recursively save linked CATIA files at PDM update. This works across the supported CATIA link types.



➔

Picture 67: CATIA documents saved recursively following links

Configuration

In the Schema file the setting

```
<setting name="UpdateLinkedFiles" value="true"/>
```

has to be defined.

Also the settings

```
<setting name="UpdDrw-CheckLinked3DFiles" value="false"/>
<setting name="UpdPrt-CheckLinked3DFiles" value="false"/>
```

have to be changed accordingly, because the value “true” for these two settings conflicts with the “UpdateLinkedFiles” functionality.

Always save the "Path to Root" CATProducts of modified CATParts

There is an option to update the parent CATProducts of modified CATParts in a CATProduct structure. The behavior is as if the "path to root" of parent CATProduct nodes would have been manually modified by the user before clicking on the "PDM Save" button.

Configuration

This behavior can be switched on by defining the setting "AlwaysUpdatePathToRoot" with the value "true" in the Schema file:

```
<settings>
   ...
   <setting name="AlwaysUpdatePathToRoot" value="true" />
   ...
</settings>
```

Usage

The PDM Update / Save behavior is as usual, except that the CATProducts on the path to root are set to "modified" before the PDM save process.

Revise Configuration

Context Action "Revise"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Revise" usedIn="PdmWindow" />
```

Modifiable Attributes

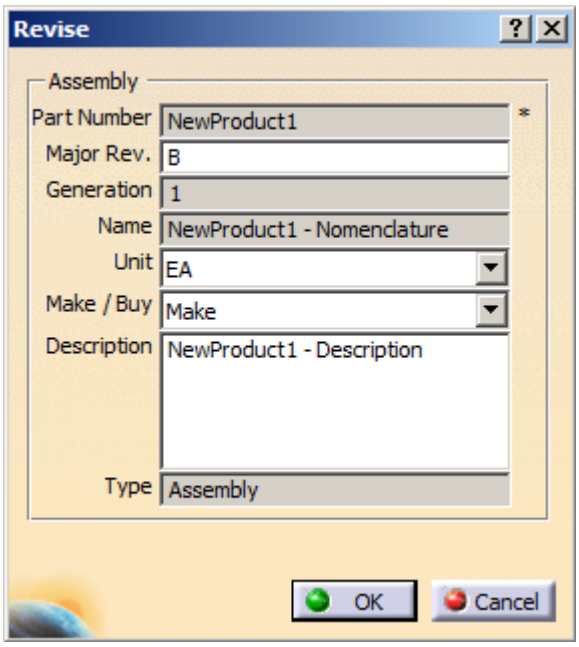
In the Schema file it can be configured which attributes can by additionally modified in the "Revise" dialog.

Example:

```
<attrsModifiableAtRevise value="major_rev|description" />
```

These attribute values can be combined with the pipe "|".

Optional.



Picture 68: "Revise" dialog with modifiable attributes "major_rev" and "description"

Expand Structure

In the Schema file it can be configured if the structure should be expanded before the revise.

Example:

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

Default value: "false"

Optional. Possible values: "true", or "false".

Revising as a different User ("ReviseAs")

By default, only administrators can directly promote the life cycle state of an item. The processes in some companies require that also regular designers should be able to promote items. For this it is possible to perform the "Revise" operation as a different user with the necessary permissions.

The PWB Configuration item setting "ReviseAs" has to be set to an identity with the necessary permissions, for instance "Aras PLM":

ReviseAs	Aras PLM

Picture 69: Sample ReviseAs configuration

Duplicate Configuration

Duplicate

By default, in the Duplicate process, all the attributes of the "PDM Create" dialog are filled with the corresponding values of the item to be duplicated. It is possible to configure that specific dialog attributes should not be pre-filled with that value.

Configuration

This behavior is defined by setting the XML attribute “pwbAttrInfo” of the dialog attribute to the value “ClearAtDuplicate” in the Schema file:

```
<form name="Register">
   ...
   <formAttribute name="name" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15"
required="true"
                  pwbAttrInfo="ClearAtDuplicate" />
   ...
</form>
```

Usage

For the defined attributes the values in the “PDM Create” dialog will be empty.

Immediately start Update

When using the “Duplicate” command, the “PDM Create” dialog of the duplicated item is shown immediately. If you use a “Custom Row in Update Dialog”, it is possible to deactivate the immediately start of the “PDM Create” dialog to allow the user to choose the correct settings in the custom row of the “Update to PDM” dialog.

Configuration

This behavior can be switched on by defining the setting “DuplicateCmdStartImmediately” with the value “false” in the Schema file:

```
<settings>
  …
  <setting name="DuplicateCmdStartImmediately" value="false" />
  …
</settings>
```

Duplicate Structure Configuration

It is possible to duplicate CATProduct structures, not only single CATPart or CATProduct documents.

Context Action “Duplicate Structure”

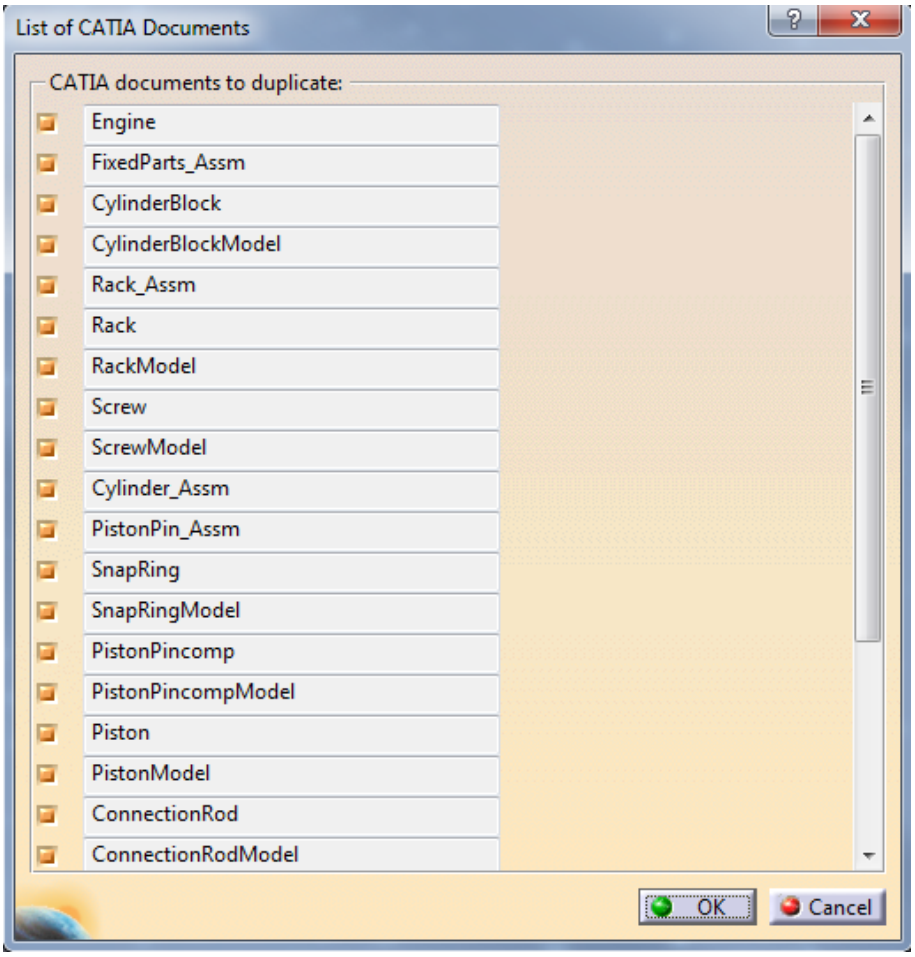
In the Schema file the context action can be enabled.

Example:

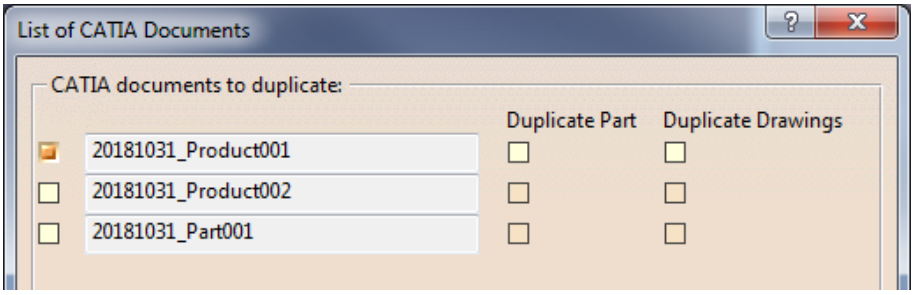
```
<contextAction name="DuplicateStructure"
usedIn="PdmWindow|QueryDialog"/>
```

In the Schema file one of the following two variants can be configured. It is not possible to use both variants in the same PDM Workbench environment.

Variant A (only available in CAD Document Structure Data Model)



Picture 70: Duplicate structure - Variant A - Preselected list of documents



Picture 71: Duplicate structure - Variant A – Duplicate Part and Drawings

Example:

```
<duplicateStructureDialog onlySelectRootNode="false" height="500"
                          width="500" maxAttrLength="50"
                          includeParts="false" includeDrawings="false"
/>
```

with

`onlySelectRootNode: "true":` only the root node is selected

`"false":` all nodes are selected

`height:` height of the dialog in pixel

`width:` width of the dialog in pixel

`maxAttrLength:` the length of the “Part Number” field in characters

`includeParts:` set to “true” to display column “Duplicate Part”

`includeDrawings:` set to “true” to display column “Duplicate Drawings”

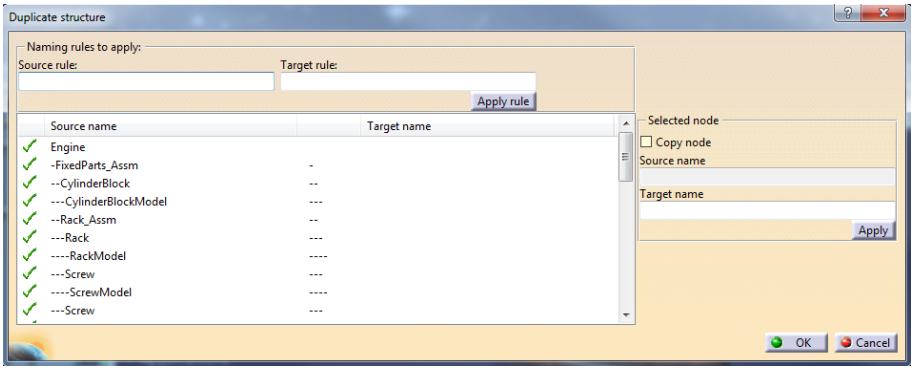
Optional.

```
<duplicateStructure useStructureDialog="false" disableSettings="true"/>
```

`useStructureDialog="false"` switches on the dialog for Variant A.

Optional.

Variant B



Picture 72: Duplicate structure - Variant B - Preselected list of documents

Example:

```
<duplicateStructure useStructureDialog="true" disableSettings="false"
/>
```

`useStructureDialog="true"` switches on the dialog for Variant B.

Optional.

```
<duplicateStructureRequest size="10000" />
```

Defines the number of CAT Documents to get the related Drawings for at one shot.

Default value: “10000”.

Optional.

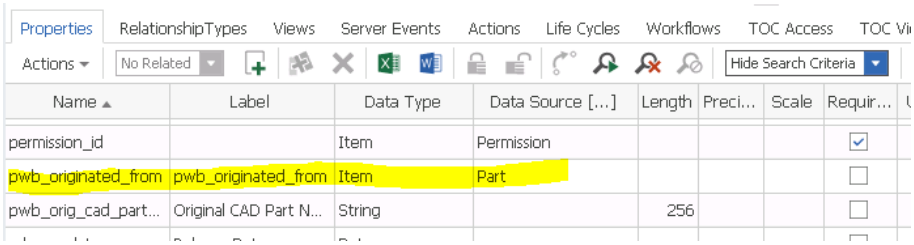
Duplicate Structure provides “originatedFrom” Property (in Variant B)

When using “Duplicate Structure” it is possible to add the original Item to a property in the new Item (CAD / Part).

Configuration

Add an item property like “pwb_originated_from” to the Item Type “CAD”.

If you use the BOM Part Structure Data Model you have also to add the property to the Item Type “Part”.



Picture 73: Item Type “CAD” – Add property “pwb_originated_from”

Create the server method “PwbCus_PreProcDlgAttrs_OriginatedFrom” to pre-process the “PDM Create” dialog attributes. This method uses the property “pwb_originated_from”.

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
	string PdmType = this.getProperty("Type");
	string PdmClassification = this.getProperty("Classification");

	PwbServerApiObj.Log(
		"Called method -> '" + PdmType + "' / '" + PdmClassification + "'" );

	IDictionary<string, string> PartInputDialogDict = null;
	if (PdmType == "Part")
	{
		// Part dialog attributes
		Item PartInputDialogItem = this.getPropertyItem("PartDialogAttrs");
		if (PartInputDialogItem == null)
		{
			return getInnovator().newError("" +
				"Could not retrieve part dialog attributes!");
		}
		else
		{
			PartInputDialogDict =
				PwbServerApiObj.DialogAttrsItemToDictionary(
					PartInputDialogItem);
			if (PartInputDialogDict == null)
			{
				return getInnovator().newError(
					"Could not convert part dialog attributes!");
			}
		}
	}

	IDictionary<string, string> CadDocInputDialogDict = null;
	if (PdmType == "CAD")
	{
		// CAD dialog attributes
		// The CAD document dialog is only needed when a CAD type needs to be
created
		Item CadDocInputDialogItem =
			this.getPropertyItem("CadDocDialogAttrs");
		if (CadDocInputDialogItem == null)
		{
			return getInnovator().newError(
				"Could not retrieve CAD document dialog attributes!");
		}

		CadDocInputDialogDict =
			PwbServerApiObj.DialogAttrsItemToDictionary(
				CadDocInputDialogItem);
		if (CadDocInputDialogDict == null)
		{
			return getInnovator().newError(
				"Could not convert CAD document dialog attributes!");
		}
	}

	// CATIA standard properties
	// 'CadPartNumber', 'CadRevision', 'CadNomenclature', 'CadDefinition',
	// 'CadDescriptionReference', 'CadFileName'
	IDictionary<string, string> CatiaStdPropsDict = null;
	Item CatiaStdPropsItem = this.getPropertyItem("CadStdProps");
	if (CatiaStdPropsItem == null)
	{
		return getInnovator().newError(
			"Could not retrieve CATIA standard properties!");
	}
	else
	{
		CatiaStdPropsDict =
			PwbServerApiObj.DialogAttrsItemToDictionary(
				CatiaStdPropsItem);
		if (CatiaStdPropsDict == null)
		{
			return getInnovator().newError(
				"Could not convert CATIA standard properties !");
		}
	}

	foreach (KeyValuePair<string, string> Attribute in CatiaStdPropsDict)
	{
		PwbServerApiObj.Log( "CATIA standard property '" +
				Attribute.Key + "'/'" + Attribute.Value + "'" );
```

```
        }

        // Resulting output dialog attributes
        IDictionary<string, string> OutputDialogDict = new Dictionary<string, string>();

        // First copying the dialog attributes
        foreach (KeyValuePair<string, string> Attribute in PartInputDialogDict)
        {
                PwbServerApiObj.Log("Part dialog attribute '" +
                                    Attribute.Key + "'/'" + Attribute.Value + "'" );
        }

        IDictionary<string, string> InputDialogDict = null;
        if ((PdmType == "CAD") || (PdmType == "Part"))
        {
                if (PdmType == "CAD")
                {
                        InputDialogDict = CadDocInputDialogDict;
                }
                else
                {
                        InputDialogDict = PartInputDialogDict;
                        if (CatiaStdPropsDict != null)
                        {
                                string OriginatedFromPartValue = null;
                                if (CatiaStdPropsDict.TryGetValue(
                                        "OriginatedFromPart",
                                         out OriginatedFromPartValue)
                                         &&
!String.IsNullOrEmpty(OriginatedFromPartValue))
                                {
                                        InputDialogDict.Add(
                                                "pwb_originated_from",
                                                OriginatedFromPartValue);
                                }
                        }
                }
        }

        Item OutputDialogItem =
                PwbServerApiObj.DialogAttrsDictionaryToItem(InputDialogDict);

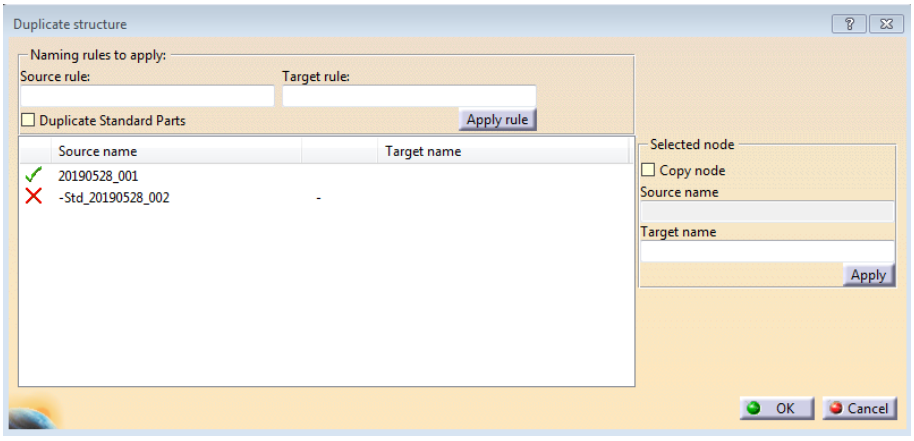
        return OutputDialogItem;
}
```

The name of the server method must be set in the PWB Configuration setting:

```
CustomMethod_PreProcCreDlgAttrs =
                        PwbCus_PreProcDlgAttrs_OriginatedFrom
```

Duplicate Structure Handling of Standard Parts (in Variant B)

When you duplicate a structure which uses standard parts, it is most likely that the standard parts stay unchanged in the new structure. If you use the default PDM Workbench mechanism to handle standard parts, you can exclude the standard parts from the renaming.



Picture 74: Duplicate Structure Handling of Standard Parts

The check box “Duplicate Standard Parts” controls the default behavior for standard parts. The user can change the behavior of a single node by selecting the line.

This functionality is available in CAD Document Structure Data Model and BOM Part Structure Data Model.

Configuration

The appearance and the default state of the “Duplicate Standard Parts” check box can be configured in the Schema file:

```
<duplicateStructure useStructureDialog="true"
                    disableSettings="false"
                    showDuplicateStdParts="true"
                    defaultDuplicateStdParts="false" />
```

with

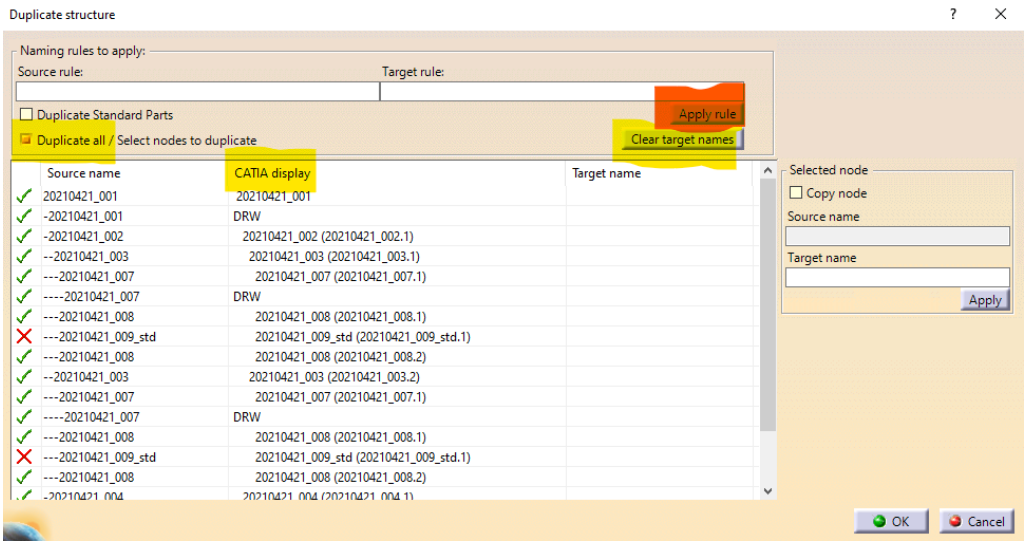
`showDuplicateStdParts = [true|false]` enables / disables the check box

`defaultDuplicateStdParts = [true|false]` sets the default state of the check box

The settings are independent, it is possible to hide the check box and disable the copy of standard parts.

Duplicate Structure Enhancements (in Variant B)

There are some new functionalities in the “Duplicate structure” dialog:



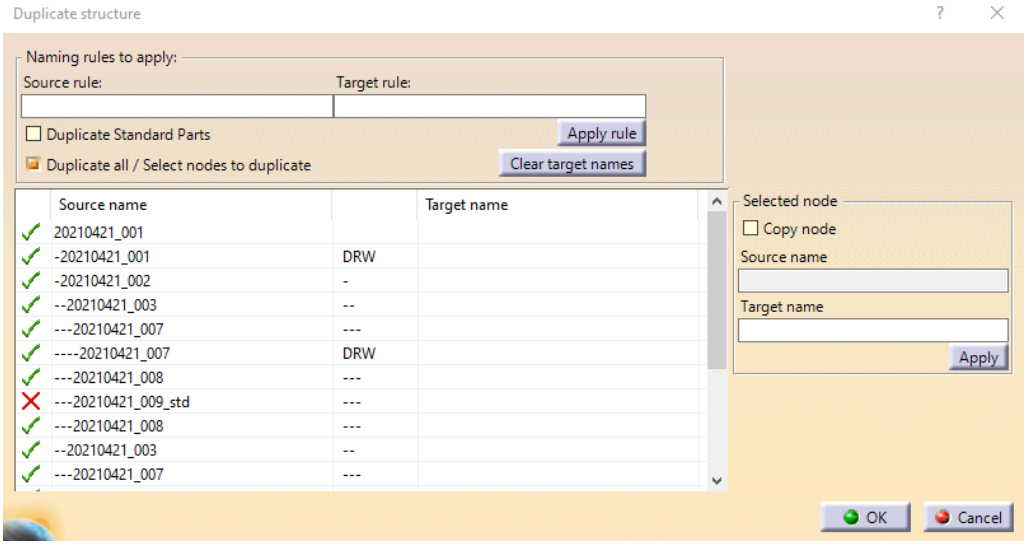
Picture 75: New function in the “Duplicate structure” dialog

1. Apply rule
   This function was changed. Now you can use the rule to create new target names multiple times. The target name is only modified if the rule hits the source name. If the source name is not hit by the rule, the target name will stay unchanged.
2. Clear target names
   This function clears all target names.
3. CATIA display
   This column shows the same text, like the nodes in the CATIA structure tree.

4. Duplicate all / Select nodes to duplicate
   By default all nodes except standard parts are selected to be duplicated. If you only want to duplicate some dedicated nodes and their path to root, you can uncheck the box. In this case, all nodes are excluded from the duplicate action. You can select the specific nodes and enable the “Copy node” manually.

Configuration

It is possible to disable the “CATIA display“ column. In this case the original column of the “Duplicate structure” dialog is activated.



Picture 76: “Duplicate structure” dialog, hide “CATIA display” column

To disable the “CATIA display” column you have to set “DuplicateStructureHideCatiaDisplay” to “true” in the settings section of the Schema file on the client:

```
<settings>
   ...
   <setting name="DuplicateStructureHideCatiaDisplay"
            value="true"/>
</settings>
```

CATIA V5 Window Configuration

“Properties” Dialog

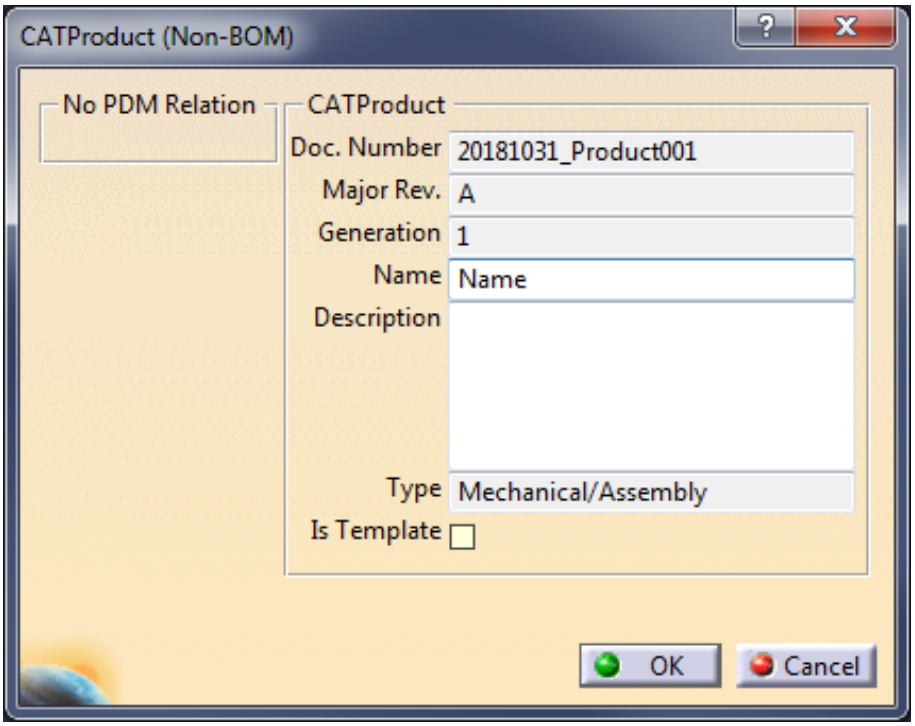
It is possible to configure the dialog to be displayed for the context action “PDM Workbench → PDM Properties”.

Example:

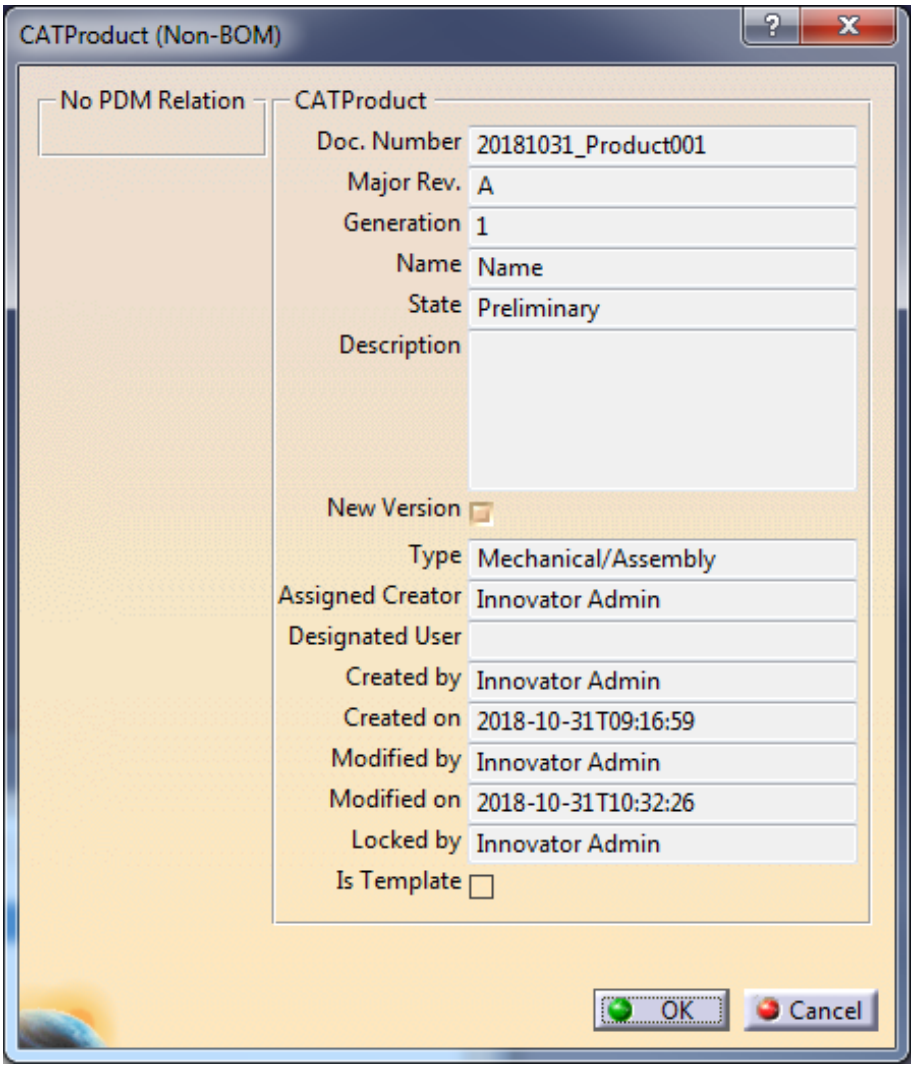
```
<catiaWindow pdmPropertiesDialog="Properties" />
```

Default value: “UpdateItem”.

Optional.



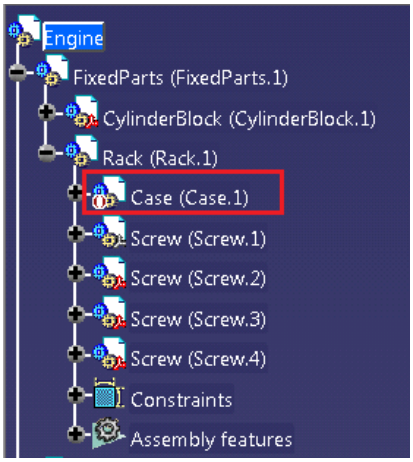
Picture 77: PDM Properties - UpdateItem



Picture 78: PDM Properties – Properties

Allow deactivated CATProduct and CATPart Instances

It is possible to import and update a structure which contains deactivated nodes (“Case.1” in the picture, as opposed to “Screw.1”, where only the representation is deactivated).



Picture 79: CATProduct structure with a deactivated node

Previously, deactivated nodes were treated as not existing. With this new functionality the nodes can be treated like regular activated nodes.

Also, the activation state of a CATPart or CATProduct instance can be passed to a custom method when the corresponding PDM relation is created, making it possible to different parameter values based on the activation state.

For switching on the management of deactivated nodes the value of the XML attribute “allowDeactivatedNodes” of the XML tag “catiaNodeBehaviorDefinitions” has to be set to “true”:

```
<catiaNodeBehaviorDefinitions ignoreLeafComponents="false"
                              allowDeactivatedNodes="true" >

    ...

</catiaNodeBehaviorDefinitions>
```

Optional.

For passing the activation state to a custom server method when a new CATIA instance is processed in the update process the server-side configuration setting “CustomMethod_PreProcRelCreAttrs” has to be set to the name of a server-side C# method, for example like this:

CustomMethod_PreProcRelCreAttrs	PwbCus_PreProcRelCreAttrs

Picture 80: Sample CustomMethod_PreProcRelCreAttrs configuration

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
        PwbServerApiObj.Log("PwbCus_PreProcRelCreAttrs");

        // Input information
        string Type = this.getProperty("Type");

        string LeftObjType = this.getProperty("LeftObjType");
        string LeftObjId = this.getProperty("LeftObjId");

        string RightObjType = this.getProperty("RightObjType");
        string RightObjId = this.getProperty("RightObjId");

        string InstanceIsActivatedInCad = this.getProperty("InstanceIsActivatedInCad");
```

```
            string MsgStr = "Type:'" + Type +
                            "', LeftObjType:'" + LeftObjType +
                            "', LeftObjId:'" + LeftObjId +
                            "', RightObjType:'" + RightObjType +
                            "', RightObjId:'" + RightObjId +
                            "', InstanceIsActivatedInCad:'" + InstanceIsActivatedInCad + "'";

            PwbServerApiObj.Log(MsgStr);

            Item InputPdmAttrsItem = this.getPropertyItem("InputPdmAttrs");
            if (InputPdmAttrsItem == null)
            {
                    throw new Exception(
                       "PwbCus_PreProcRelCreAttrs -> InputPdmAttrsItem == null !");
            }

            IDictionary<string, string> InputPdmAttrsDict =
               PwbServerApiObj.DialogAttrsItemToDictionary(InputPdmAttrsItem);

            // Resulting output PDM attributes
            IDictionary<string, string> OutputPdmAttrsDict =
                    new Dictionary<string, string>();

            foreach (KeyValuePair<string, string> Attribute in InputPdmAttrsDict)
            {
                    PwbServerApiObj.Log("Attribute.Key:'" + Attribute.Key +
                                        "', Attribute.Value:'" + Attribute.Value + "'");

                    OutputPdmAttrsDict.Add(Attribute.Key, Attribute.Value);
            }

            if (Type.Equals("CAD Structure"))
            {
                    if (InstanceIsActivatedInCad.Equals("False"))
                    {
                            // Add or modify attributes here
                    }
            }

            Item OutputPdmAttrsItem =
                    PwbServerApiObj.DialogAttrsDictionaryToItem(OutputPdmAttrsDict);

            return OutputPdmAttrsItem;
    }
```

To get all properties of the child Item (avoid query) you have to add the setting “PreProcRelInstCreAttrsGetAllChildAttrs” to the settings section of the Schema file:

```
<settings>

   <!-- <setting name="PreProcRelInstCreAttrsGetAllChildAttrs" value="true"/> --
>

</settings>
```

PDM Status Information in the CATIA Tree

A PDM status information can be displayed on product structure nodes within the CATIA tree. These nodes show an additional icon mask and additional text and tooltip text corresponding to some information from the PDM system.

Configuration

Edit the following dictionary file on the client, if necessary:

```
<PWB Installation
dir>\win_b64\code\dictionary\PWBUiCatiaMod.dico
```

Remove the comment sign “#” at the beginning of the two lines

```
#CATProduct CATInit libPWBUiCatiaModLib
```

```
#ASMPRODUCT_node CATINavigModify libPWBUiCatiaModLib
```

Create two server PWB configuration settings for the CadTreeIconInfo:

Name	Value

CustomMethod_CadTreeIconInfo	Pwb_Sample_CadTreeIconInfo
CadTreeIconInfoNotInPdm	I_MaskNew| - Not in PDM| - Not in PDM
CadTreeIconInfoUsePartInfo	true

The value of the setting “CustomMethod_CadTreeIconInfo” defines the real name of the Aras Innovator server method, that you use for your customization.

A sample method for this, “Pwb_Sample_CadTreeIconInfo”, exists. You can use this as the base for your own implementation.

The value of the setting “CadTreeIconInfoNotInPdm” defines the strings to be used, if a relevant CATIA node does not have a corresponding item in the PDM system (e.g. because that document was not yet stored in Aras Innovator).

The value is a multi-field string. The fields are separated by a vertical bar “|”.

Field 1: The icon name (without the .bmp suffix)

Field 2: The text to be appended to the regular tree node text

Field 3: The text to be appended to the regular short help (Tooltip)

Example: I_MaskNew| - Not in PDM| - Not in PDM

Use the setting CadTreeIconInfoUsePartInfo = true (default false), if the CADs have related Parts and you want to access some Part properties in the custom method.

The server method is called whenever an Aras Innovator CAD Document item information is retrieved by a PWB CATIA call to the Aras Innovator server. The CAD Document item information to be returned is provided to the server method and can be used by the server method to define the additional PDM info in the CATIA specification tree.

The method returns three strings: IconName, NodeText, HelpText (see example below).

The strings can be simple strings or multi-field strings (see “Released” case below).

Simple strings are faster processed on the client; they simply define what to show on the related CATIA node.

Multi-field strings lead to more processing on the client, but provide advanced capabilities:

Field 1: The string is used in regular cases

Field 2: The string is used, if the CATIA document in the current session is dirty (modified)

Field 3: The string is used, if the CATIA document in the current session is not dirty, but not synchronized with the Aras Innovator server information.

Strings over multiple lines (containing “\n”) are supported, but only for the HelpText wb(Tooltip) in simple strings and multi-field strings.

Additional Configuration when working with very large structures:

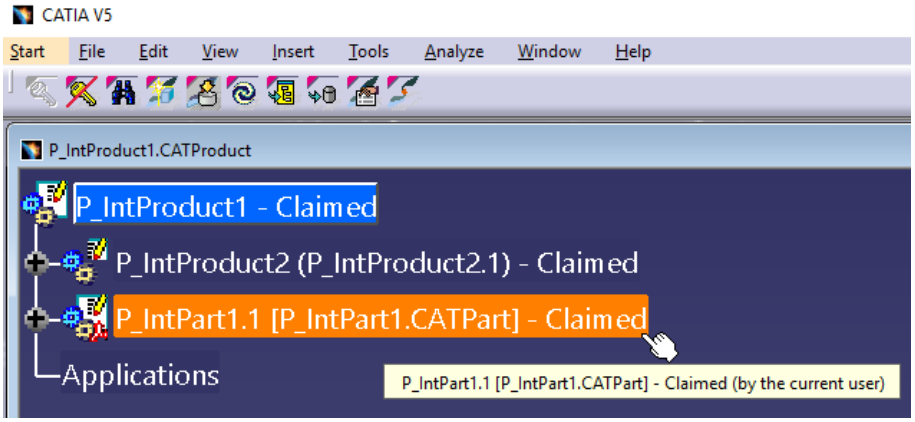
By default the icons / text in the CATIA Product Structure Navigation tree (PSN tree) is refreshed after every PDM Workbench action which could change any icon or text like claim or unclaim. To avoid these additional PSN tree refreshes you can disable this behavior with the following setting in the Schema file:

```
<settings>
   ...
   <setting name="RefreshPsnTreeAfterAction" value="false" />
   ...
</settings>
```

If “RefreshPsnTreeAfterAction” is set to “false” the PSN tree is only refreshed when native CATIA redraws it (like change of the workshop, …). This setting should only be used if there are performance issues about the refresh of the PSN tree.

Usage

If configured by the administrator, a PDM status information is displayed on product structure nodes within the CATIA tree:



Picture 81: PDM status information in the CATIA tree

These nodes show an additional icon mask in the upper right corner of the icon and additional text and tooltip text corresponding to some information from the PDM system.

PDM Structure Window Configuration

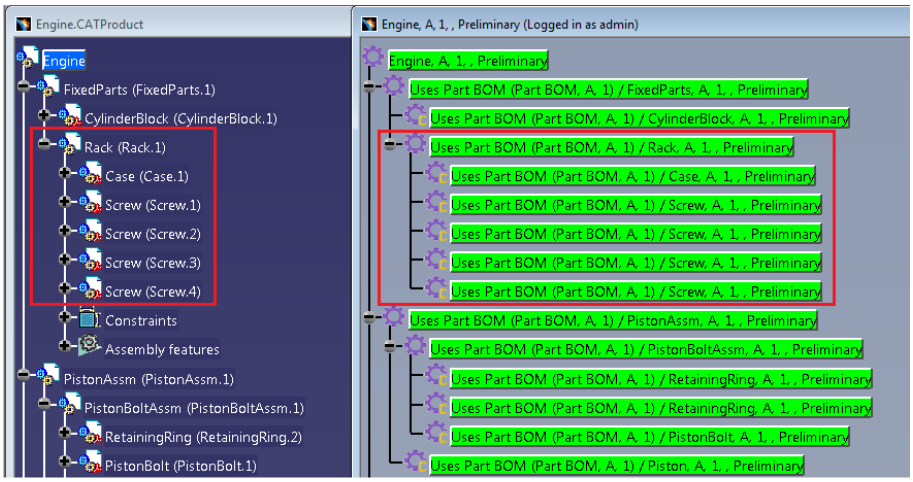
Displaying Part Structure Instances as separate Nodes

The display of the part structure in the PDM structure window can be changed such that every part instance is shown as a separate node.

The PWB Configuration item setting “UseSeparateRelationsForInstances” has to be set to “true” to switch on this functionality.

UseSeparateRelationsForInstances	true

Picture 82: Sample UseSeparateRelationsForInstances configuration



Picture 83: Every part instance is shown as separate node

Displaying Attributes in PDM Nodes in several Lines

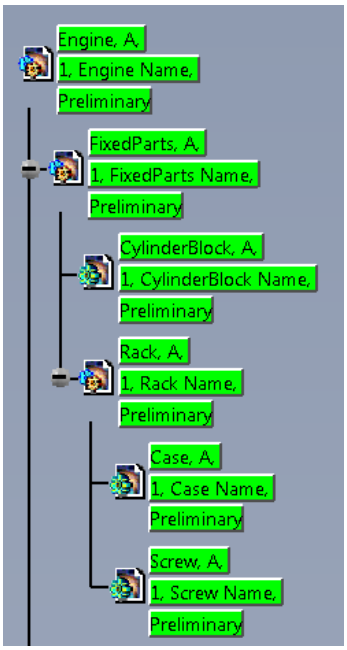
It is possible to display the PDM attribute values of nodes in the PDM structure window, like “Part Number” or “Part Name”, in several lines, as opposed to just one line.

It is also possible to define which PDM attribute values should be displayed on which line.

The Slash character (“/”) acts as a return character in the definition of the PDM attribute values to be displayed in the PDM structure window.

The following example shows two “/” characters, which means that the attribute values are displayed in three lines.

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



Picture 84: PDM Node Attributes displayed in several Lines

PWB Window Color

In the Schema file the background color of the PWB window can be configured. This tag contains the red, green, and blue values (0 - 255) of the color.

Example:

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

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

Optional.

Preselection of Nodes

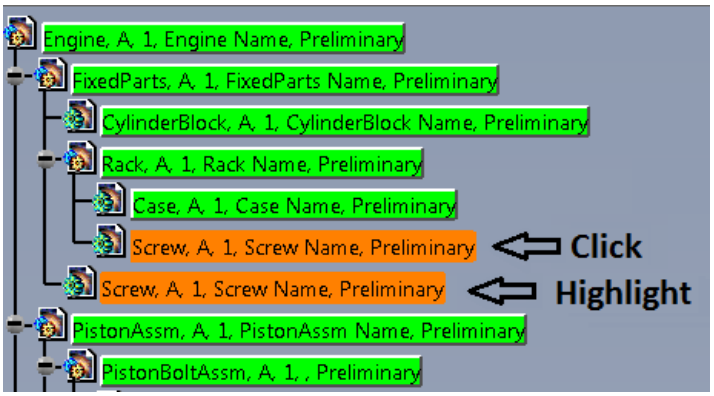
It is possible to switch off the automatic highlight of the PDM tree nodes which represent the same underlying PDM node.

In the Schema file you have to set the attribute `preselectSamePdmNode="false"` in the *pwbWindow* XML tag in order to change the behavior from the default (highlighting all related nodes) to not highlighting them.

Example:

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

In the default behavior related nodes are also highlighted when the user clicks on a PDM node.



Picture 85: Default highlight behavior

Context Action "Delete relation"

PDM relations can be deleted in the PDM structure window with a single context menu action, even if the PDM relations are not displayed in the structure.

For any item the context action "Delete Relation" can be added in the Schema file:

```
<contextAction name="DeleteRelation" usedIn="PdmWindow" />
```

Comparing PDM Structure Trees

It is possible to compare two structures, or two generations of the same structure, displaying the differences between these two structures.

The functionality can be configured with the "pwbCompareStructureWindow" tag and the "nodeColors" sub-tag in the Schema file:

```
<!-- possible colors for nodes in 'compare structure window':
"green", "red", "magenta", "cyan", "blue", "yellow", "black",
"white", "background" -->

<pwbCompareStructureWindow red="200" green="200" blue="150"
                           showSameChildNodes="false" >

   <nodeColors additionalNodes="green" missingNodes="red"
               sameNodesDifferentVersion="magenta"
               sameNodes="background" />

</pwbCompareStructureWindow>
```

The attributes "red", "green" and "blue" of the tag "pwbCompareStructureWindow" define the respective value, from 0 to 255, of the primary color. The three values define the color of the window background.

The attribute "showSameChildNodes" defines whether nodes which exist in both structures should be shown.

The attributes "additionalNodes", "missingNodes", "sameNodesDifferentVersion", and "sameNodes" of the tag "nodeColors" define the color of additional nodes, missing nodes, same nodes with a different generation, and the same nodes of one structure compared to the other structure, respectively. The values can be chosen from the fixed set "green", "red", "magenta", "cyan", "blue", "yellow", "black", "white", and "background".

Optional.

To use the functionality the context action "Compare with Assembly" has to be defined for "Part/Assembly" in the part BOM Part Structure Data Model and for "CAD/Mechanical/Assembly" in the CAD Document Structure Data Model.

Example:

```
<contextAction name="CompareStructure" usedIn="PdmWindow" />
```

Selecting Nodes in the PDM Structure Window

It is possible to select specific child nodes of a structure by defining logical AND or OR combinations of values of different attributes.

The dialog form “SelectChildNodes” has to be defined. It contains the attributes which can be used for selecting the nodes. This is an example definition of the dialog:

```
<form name="SelectChildNodes">
   <formAttribute name="item_number" widgetType="ComboBox" ... />
   <formAttribute name="major_rev" widgetType="ComboBox" ... />
   <formAttribute name="generation" widgetType="ComboBox" ... />
   <formAttribute name="name" widgetType="ComboBox" ... />
   <formAttribute name="state" widgetType="ComboBox" ... />
   <formAttribute name="description"
                  widgetType="MultiLineEditor" ... />
   <formAttribute name="created_on" widgetType="ComboBox" ... />
   <formAttribute name="modified_on" widgetType="ComboBox" ... />
   <formAttribute name="created_by_id" widgetType="ComboBox" ... />
   <formAttribute name="modified_by_id" widgetType="ComboBox" ... />
   <formAttribute name="locked_by_id" widgetType="ComboBox" ... />

   <formAttribute name="PwbSelectChildNodesCondition"
                  widgetType="RadioButtons" mode="update"
                  visibleLength="15" required="true"
                  entryAllowed="true" />
</form>
```

In addition to this definition the context action “Select Nodes” has to be defined for “Part/Assembly” or for “CAD/Mechanical/Assembly”.

Example:

```
<contextAction name="SelectNodes" usedIn="PdmWindow" />
```

Alternative Icons for PDM Structure Nodes

It is possible to define different icons for different PDM attribute values.

To switch on the functionality the value of the XML attribute “attrDepIcons” has to be set to “true”:

```
<pwbWindow preselectSamePdmNode="true" attrDepIcons="true">
   <pwbWindowColor red="143" green="155" blue="177" />
</pwbWindow>
```

In addition to this, for each PDM item that should have attribute-dependent icons the definition string in the “icon” XML tag has to be changed to contain the PDM attributes whose values the icon should be dependent of. The syntax is to write the PDM attribute name contained in the square brackets (“[“ and “]”).

This is an example.

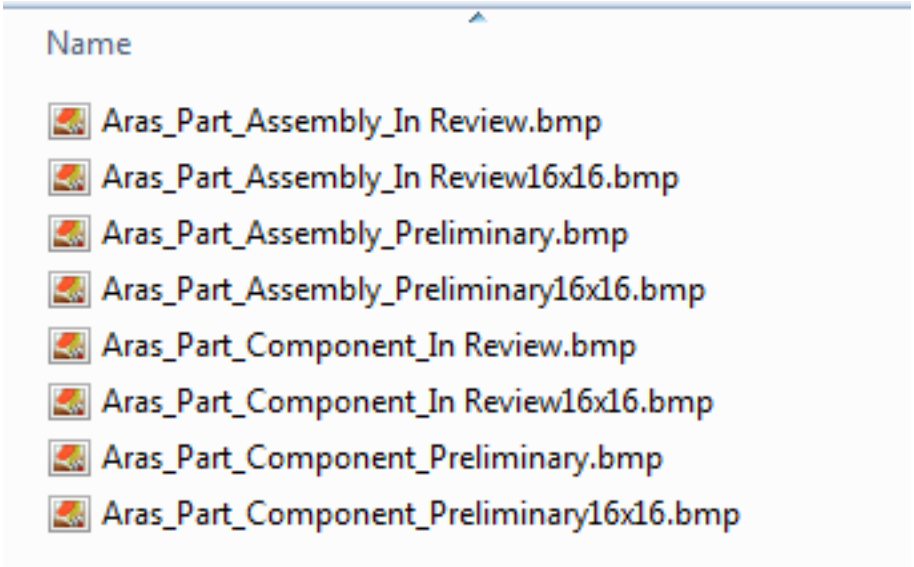
```
<object name="/Part/Assembly"
        displayName="NLS_Assembly" icon="Aras_Part" >
```

➔

```
<object name="/Part/Assembly"
        displayName="NLS_Assembly"
        icon="Aras_Part_[classification]_[state]" >
```

If this functionality is switched on, the icons with the corresponding names will have to exist in the directory `win_b64\resources\graphic\icons\normal`, otherwise the PDM nodes will be displayed without icons.

These are example icon names:



Picture 86: Example icon names

Expand Configuration

Context Action "Expand"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Expand" usedIn="PdmWindow|QueryDialog" />
```

Context Action "Expand Multiple Levels"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="MultipleExpand" usedIn="PdmWindow" />
```

Context Action "Custom Expand"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CustomExpand" usedIn="PdmWindow" />
```

The custom expand for the object class has to be defined:

```
<customExpand>
   <pdmTreeRef name="PartToCatiaFilePdmTree" />
</customExpand>
```

The PDM tree has to be defined:

```
<pdmTree name="PartToCatiaFilePdmTree"
         displayName="NLS_PartToCatiaFilePdmTree">

   <pdmTreeRel class="Part CAD" direction="LEFT_TO_RIGHT">

      <pdmTreeObj
class="/CAD/Mechanical/Assembly|/CAD/Mechanical/Part|/CAD/cgr|/C
AD/Mechanical/Drawing|/CAD/Mechanical/cgr|/CAD/Mechanical/CatiaV
4Model">

         <pdmTreeObjPartToFileInfo
partClass="Part|/Part/Assembly"
             fileClass="/CAD/Mechanical/Assembly"
             fileType="Main" />

         <pdmTreeObjPartToFileInfo
partClass="Part|/Part/Assembly"
             fileClass="/CAD/Mechanical/Drawing"
             fileType="Auxiliary" />


         <pdmTreeObjPartToFileInfo partClass="/Part/Component"
             fileClass="/CAD/Mechanical/Part"
             fileType="Main" />

         <pdmTreeObjPartToFileInfo partClass="/Part/Component"
             fileClass="/CAD/Mechanical/cgr"
             fileType="Main" />

         <pdmTreeObjPartToFileInfo partClass="/Part/Component"
             fileClass="/CAD/Mechanical/CatiaV4Model"
             fileType="Main" />


         <pdmTreeObjPartToFileInfo partClass="/Part/Component"
             fileClass="/CAD/Mechanical/Drawing"
             fileType="Auxiliary" />

      </pdmTreeObj>

   </pdmTreeRel>

</pdmTree>
```

Context Action "De-Expand"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="DeExpand" usedIn="PdmWindow" />

<deExpand legacyMode="false" onlyChildNodes="false" />
```

Default values: "false" and "false"

Optional. Possible values: "true", or "false".

Context Action "Expand Multiple Levels including Bom Children"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="MultipleExpandWithBomChildren" usedIn="PdmWindow"
/>
```

Explicit Display and Control of "AsSaved" and "Current" Expand Resolutions

If a CAD structure is expanded with the "Current" expand resolution the PDM structure window contains information about which generations of the CAD Documents are saved (which generation would be retrieved with the "AsSaved" expand resolution.

Also, in the update process the user can define whether the CAD Structure relations should be moved to the current generation of the child CAD Document or not.

This information is shown by default. It can be switched off by setting the server configuration "ShowCadStrcExpResChildDocGenInfo" to "false".

Check for 'CAD Structure' expand resolution.

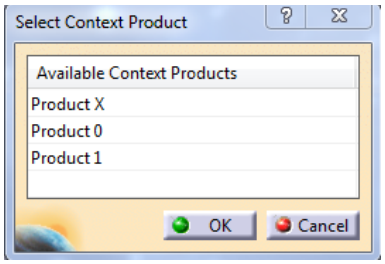
Before loading a structure into CATIA, PDM Workbench checks if the structure (or a sub structure) is already loaded in CATIA using a different expand mode. The check about the different expand mode was refined. The PDM Workbench no longer uses the expand mode to determine if it is possible to load the structure. Now PDM Workbench uses the versions of the expanded children to determine if it is possible to load the structure.

Manage Context Products

A user can only work in one Context Product at one time. This Context Product is used to store newly created files to the correct vault.

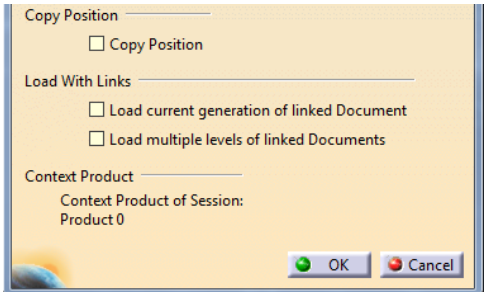
If the functionality is enabled, there must be at least one valid Context Product for every user.

The Context Product is set during login process in CATIA. If there are multiple valid Context Products, the user must select one. If there is no Context Product available for the user, the user cannot login.



Picture 87: Select Context Product

The currently used Context Product can be seen in the "PDM Workbench Options" dialog.



Picture 88: Currently used Context Product

Configuration

The possible Context Products for a user have to be determined in a custom method. The name of the custom method has to be set to the PWB Configuration item setting “CustomMethod_ContextProduct”. Setting the name of the custom method to the PWB Configuration item setting also enables the rest of the functionality.

CustomMethod_ContextProduct	PwbCus_ContextProductMethod

Picture 89: Sample CustomMethod_ContextProduct configuration

Sample Method:

```
Innovator InnovatorObj = this.getInnovator();
var userID = InnovatorObj.getUserID();


Item result = InnovatorObj.newItem("result");
result.setAttribute("ContextProduct", "Product X");
result.setAttribute("ContextProductId", "12345678901234567890");


var rand = new Random();
int count = rand.Next(5);
for (int i = 0 ; i < count ; i++)
{
   Item x_result = InnovatorObj.newItem("result");
   x_result.setAttribute("ContextProduct", "Product " + i.ToString());
   x_result.setAttribute("ContextProductId",
      (1234567890123456789 *(i+1)).ToString());
   result.appendItem(x_result);
}
return result;
```

The currently used `ContextProduct` and the corresponding `ContextProductId` is available in every custom method that provides the cadProperties, like “CustomMethod_PreProcCreDlgAttrs”.

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 “PDM Workbench 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.

Naming Configuration - Numbering

CadDocNumberAttr

The PWB Configuration item setting “CadDocNumberAttr” contains the name of the “CAD Document Number” attribute of the Item Type “CAD Document”.

CadDocNumberAttr	item_number

Picture 90: Sample CadDocNumberAttr configuration

Default value: “item_number”

Optional.

PartNumberAttr

The PWB Configuration item setting “PartNumberAttr” contains the name of the “Part Number” attribute of the Item Type “Part”.

PartNumberAttr	item_number

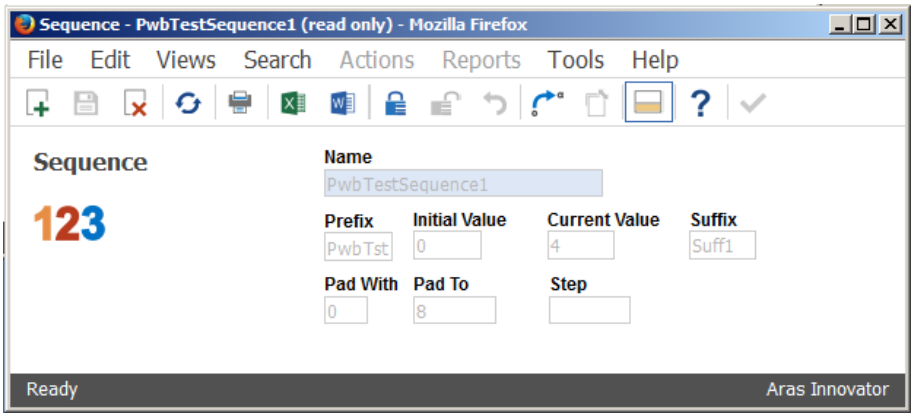
Picture 91: Sample PartNumberAttr configuration

Default value: “item_number”

Optional.

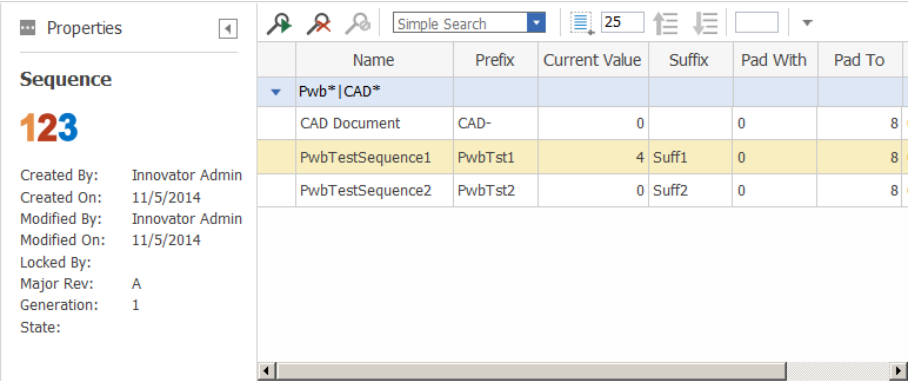
Autoname Support using Aras Innovator Sequence Items

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



Picture 92: Sample Sequence item

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



Picture 93: 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="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"
                     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 “false” in the active PWB Configuration item.

Optional.

Autoname Functionality can use a Server Method

The autoname functionality can use a server method 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>
```

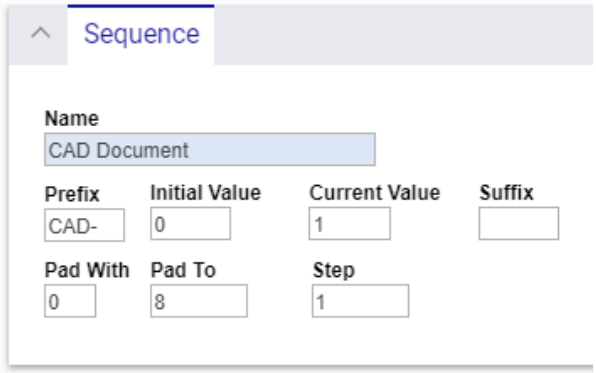
```
   <!-- "LocalWorkspace", "Register", "Update", "Synchronize",
"Refresh",
        "SetSessionConfig", "NewPwbWindow", "About"
     -->
   <!-- <icon name="LocalWorkspace" /> -->
   <icon name="Register" />
   <!-- <icon name="Update" /> -->
   <icon name="Synchronize" />
   <!-- <icon name="Refresh" /> -->
   <!-- <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 CATIA attributes of the CATIA files to be imported to PDM, or values from PDM Workbench dialogs.

In this example the Aras Innovator Sequence “CAD Document” is used for generating the number.



Picture 94: Sequence item “CAD Document”

Please check “Pwb_Sample_AutonameMethod” for a sample on which you can build your own custom autoname functionality.

Autoname Functionality in Create Dialog

When the autoname functionality is used the Part Number for Parts and the Doc. Number for CAD Documents have to be removed from the “Register” dialogs in the Schema file.

Otherwise, the user will be confused when he has filled in the values in the dialogs and his entries are overwritten by the autoname functionality.

Initial Version Strings

InitialVersionCadDoc

The PWB Configuration item setting “InitialVersionCadDoc” is the initial version string for the Item Type “CAD Document”.

InitialVersionCadDoc	A

Picture 95: Sample InitialVersionCadDoc configuration

Optional.

InitialVersionPart

The PWB Configuration item setting “InitialVersionPart” is the initial version string for the Item Type “Part”.

InitialVersionPart	A

Picture 96: Sample InitialVersionPart configuration

Optional.

Promote States

Context Action “Promote”

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Promote" usedIn="PdmWindow" />
```

PromoteSourceStates

The PWB Configuration item setting “PromoteSourceStates” is a list of the promote source states, separated by “|”.

PromoteSourceStates	Preliminary|In Review

Picture 97: Sample PromoteSourceStates configuration

Default value: “Preliminary|In Review”

Optional.

PromoteTargetStates

The PWB Configuration item setting “PromoteTargetStates” is a list of the promote target states, separated by “|”.

PromoteTargetStates	In Review|Released

Picture 98: Sample PromoteTargetStates configuration

Default value: “In Review|Released”

Optional.

PromotePartSourceStates

The PWB Configuration item setting “PromotePartSourceStates” is a list of the promote source states for the Item Type “Part”, separated by “|”.

PromotePartSourceStates	Preliminary|In Review 2

Picture 99: Sample PromotePartSourceStates configuration

This value will overwrite the default value of the PWB Configuration item setting “PromoteSourceStates”.

The settings “PromotePartSourceStates” and “PromotePartTargetStates” have to be defined together to set the Life Cycle States for the Item Type “Part”.

Optional.

PromotePartTargetStates

The PWB Configuration item setting “PromotePartTargetStates” is a list of the promote target states for the Item Type “Part”, separated by “|”.

PromotePartTargetStates	In Review 2|Released 2

Picture 100: Sample PromotePartTargetStates configuration

This value will overwrite the default value of the PWB Configuration item setting “PromoteTargetStates”.

The settings “PromotePartSourceStates” and “PromotePartTargetStates” have to be defined together to set the Life Cycle States for the Item Type “Part”.

Optional.

PromoteCADSourceStates

The PWB Configuration item setting “PromoteCADSourceStates” is a list of the promote source states for the Item Type “CAD”, separated by “|”.

PromoteCADSourceStates	Preliminary|In Review 3

Picture 101: Sample PromoteCADSourceStates configuration

This value will overwrite the default value of the PWB Configuration item setting “PromoteSourceStates”.

The settings “PromoteCADSourceStates” and “PromoteCADTargetStates” have to be defined together to set the Life Cycle States for the Item Type “CAD”.

Optional.

PromoteCADTargetStates

The PWB Configuration item setting “PromoteCADTargetStates” is a list of the promote target states for the Item Type “CAD”, separated by “|”.

PromoteCADTargetStates	In Review 3|Released 3

Picture 102: Sample PromoteCADTargetStates configuration

This value will overwrite the default value of the PWB Configuration item setting “PromoteTargetStates”.

The settings “PromoteCADSourceStates” and “PromoteCADTargetStates” have to be defined together to set the Life Cycle States for the Item Type “CAD”.

Optional.

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

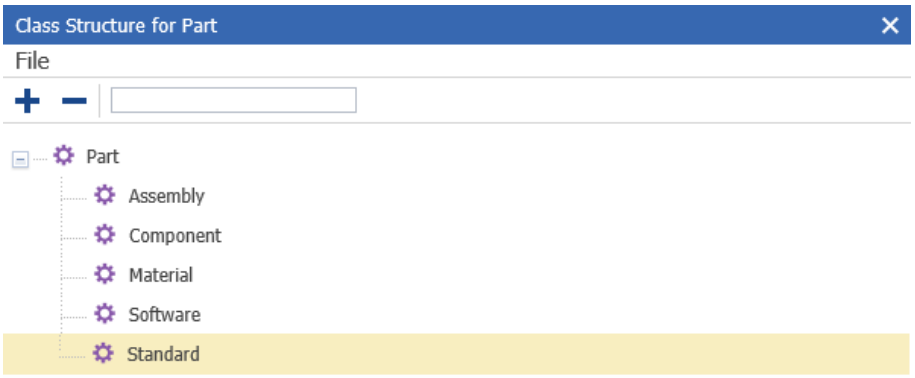
Picture 103: Sample StandardPartAdmin configuration

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

Standard Part Functionality for BOM Part Structure Data Model

In the BOM Part Structure Data Model, it is possible to define Part items and their corresponding CAD Document items as standard parts. Standard parts are supposed to be parts which are used in a wide variety of different contexts and which are generally not modified by the designer, only used in the product structures that the designer works on.

An administrator has to add the subclass “Standard” to the “Part” ItemType of Aras Innovator:



Picture 104: Subclass “Standard”

This subclass “Standard” must be used for all Part items which should be considered as a standard part. It is NOT correct to use a “Component” Part item, attach a CAD Document item and just mark that CAD Document item as “is_standard”.

The CATParts which are related to standard parts can either be downloaded like the CATIA files which are related to other parts, or they can reside in a local directory which is accessible to CATIA V5.

This setting in the Schema file defines that the standard part CATParts are downloaded to the PWB_XMAP directory:

```
<standardPartFiles download="true" />
```

Optional.

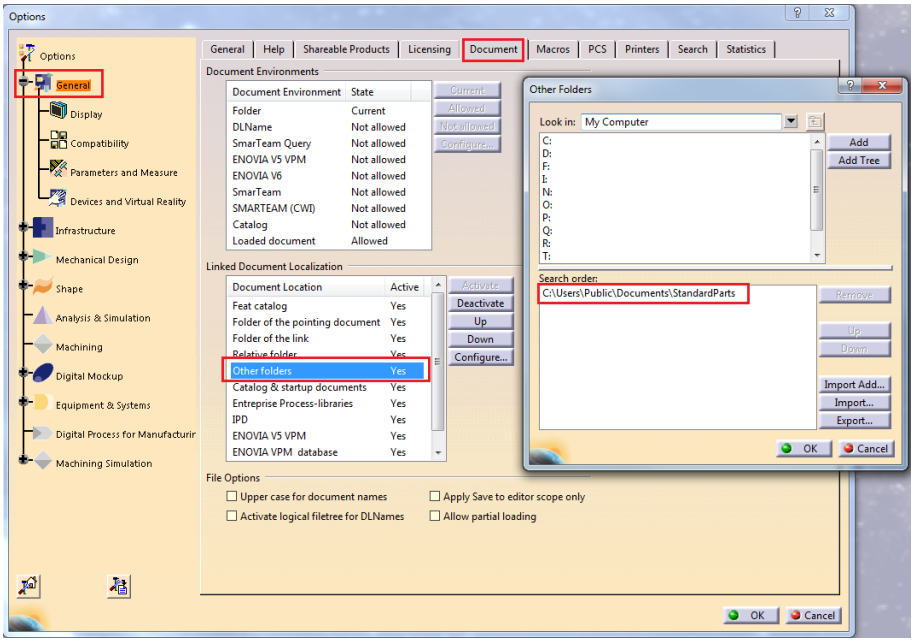
This setting defines that they are not downloaded, but taken from a local directory:

```
<standardPartFiles download="false"
                   localDir="C:\Users\Public\Documents\StandardParts"
/>
```

Optional.

For this setting three conditions need to be fulfilled:

1. The defined directory is accessible in the CATIA V5 environment.
2. The standard part CATPart files reside in that directory.
3. CATIA V5 needs to be set up to be able to use files from that directory.



Picture 105: Making CATParts in a local folder accessible

Another setting makes it possible to import structures containing standard part CATParts to PDM even if those standard parts have not been loaded in the PDM Workbench session.

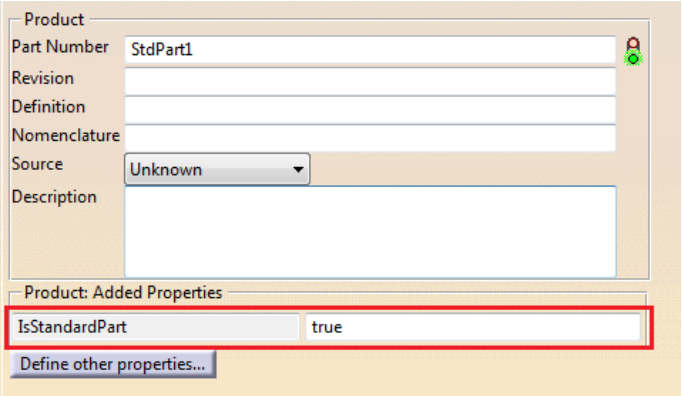
Normally CATIA files which are not loaded from PDM are treated as new items, and new CAD Documents and parts are created for them in the update process. If CATParts contain a certain definition as a user-defined attribute they are treated as standard parts, and the update process queries for existing standard Part items in the database instead of trying to create a new Part item. If such a standard Part item is found it is related to the parent part.

In the Schema file it is possible to define the name of the user-defined attribute which defines a standard part CATPart file, for instance “IsStandardPart”:

```
<standardPartUserDefPropAttribute name="IsStandardPart" />
```

Optional.

A user-defined attribute with the same name has to be defined in the CATPart file. If the value is any of the strings “true”, “TRUE”, or “1”, then the CATPart is considered to be a standard part CATPart.



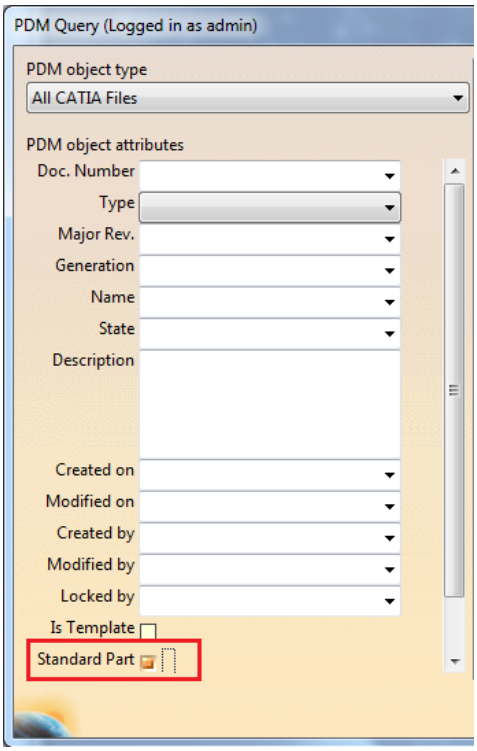
Picture 106: Defining a CATPart as a standard part

In that case the update process does not try to create a new part which corresponds to the CATIA document, but it queries for an existing standard part in PDM instead.

Standard Part Functionality for CAD Document Structure Data Model

The standard part functionality also works with the CAD Document Structure Data Model.

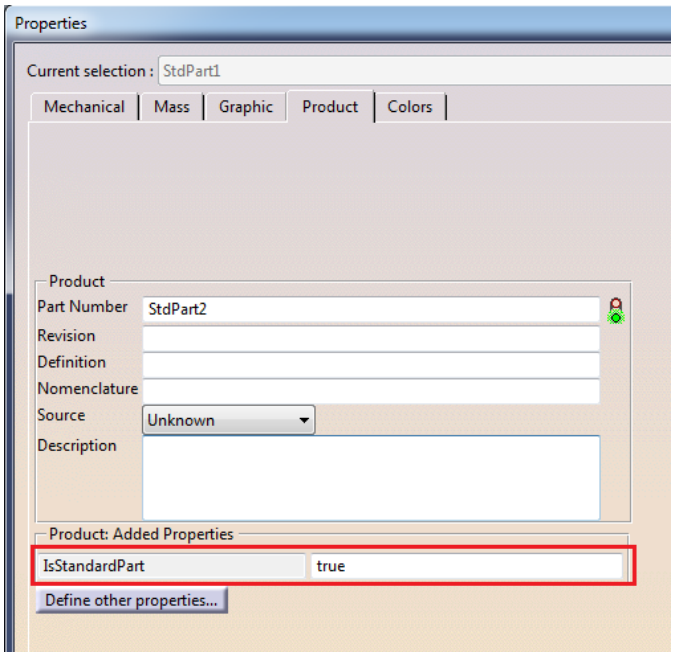
There is a check box which defines whether a CAD Document is defined as a standard Part item:



Picture 107: "Standard Part" check box for CAD Documents

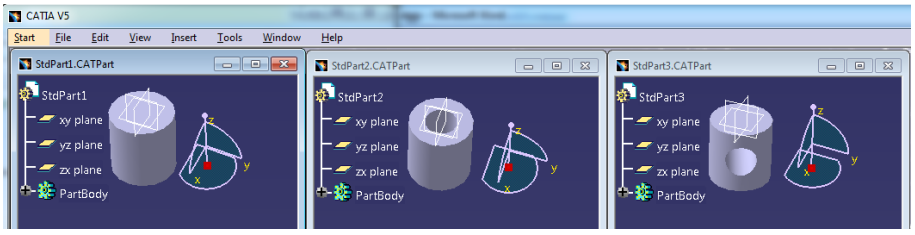
As in the BOM Part Structure Data Model standard Part items are treated differently than regular CAD Document items. Standard part CATParts (CATProducts are not allowed) are defined by two things:

1. The PDM attribute "is_standard" ("Standard Part") is set to "1".
2. In the CATPart there is a user-defined CATIA property with the name "IsStandardPart" and the value of "true".



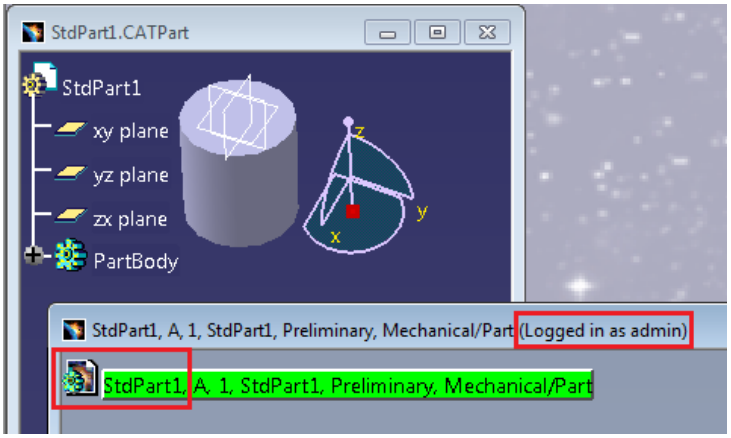
Picture 108: "IsStandardPart" user-defined property

In this example there are three CATParts which are defined as standard parts: "StdPart1.CATPart", "StdPart2.CATPart", and "StdPart3.CATPart":



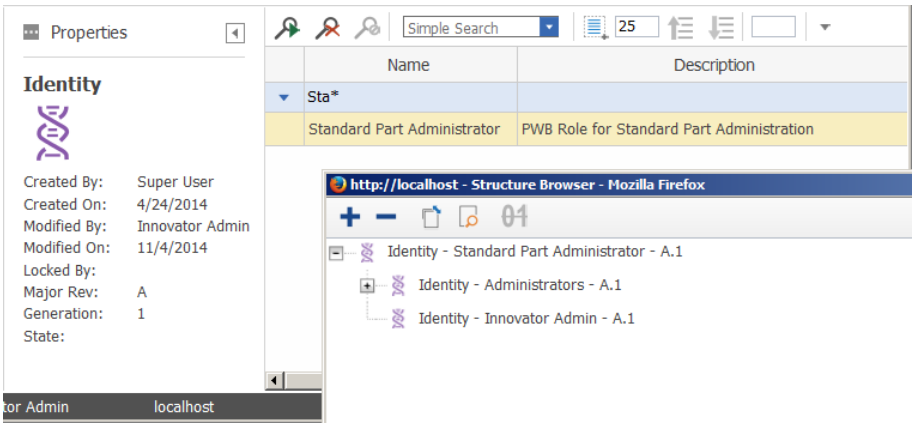
Picture 109: Three example standard part CATParts

Standard Part items can only be created by a user who is defined as standard part administrator.



Picture 110: User is logged in as administrator

By default, "Innovator Admin" is a standard part administrator, that is, a member of the "Standard Part Administrator" identity:



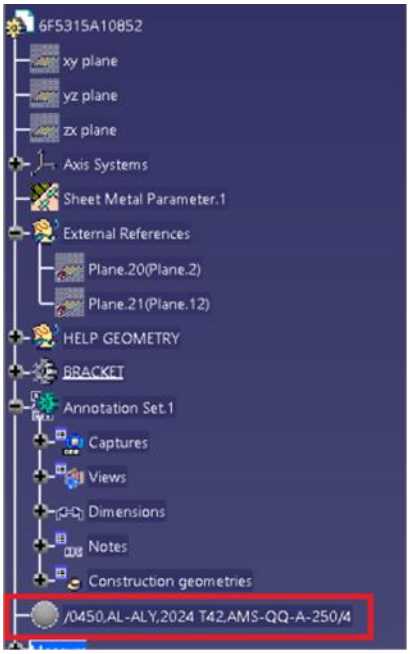
Picture 111: The "Standard Part Administrator" identity

Material

This functionality reads used materials of a CATPart and provides a custom method to handle this information in Aras Innovator.

There are multiple possibilities in CATIA where materials can be specified, only the following locations are checked according to the following rules:

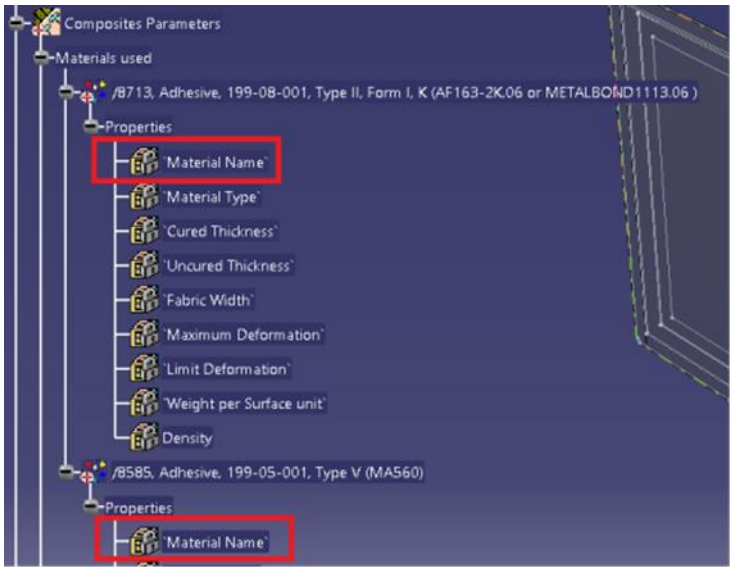
1) Look for Material at CATPart.



Picture 112: Material located at CATPart

If there is a Material we take this material and stop.

2) If there is no Material at CATPart level, look for materials in "Composites Parameters".



Picture 113: Material located at Composite Parameters

Read out all Material Names from the “Composites Parameters”.

3) If there is no Material at CATPart and “Composites Parameters”



Picture 114: Material located at Bodies

Read out the Material information at the available Bodies.

Configuration

Create a custom server method to handle the material information and set the value of the PWB Configuration item setting “CustomMethod_SaveUsedMaterials” to your method name:

CustomMethod_SaveUsedMaterials	PWBCus_SaveUsedMaterials

Picture 115: Sample CustomMethod_SaveUsedMaterials configuration

Template custom method “PWBCus_SaveUsedMaterials”:

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
        string CAD_Id = this.getID();
        string CAD_Class = this.getType();
        PwbServerApiObj.Log(
                "PWBCus_SaveUsedMaterials -> CAD_Id:'" +
                CAD_Id + "', CAD_Class:'" + CAD_Class + "'");

        string Part_Id = this.getAttribute("Part_OBID", "");
        string Part_Class = this.getAttribute("Part_Class", "");

        PwbServerApiObj.Log(
                "PWBCus_SaveUsedMaterials -> Part_Id:'" +
                Part_Id + "', Part_Class:'" + Part_Class + "'");

        int MaterialCount = 0;
        int.TryParse(this.getAttribute("materialCount"),
           System.Globalization.NumberStyles.Integer,
           System.Globalization.CultureInfo.InvariantCulture,
           out MaterialCount);

        for (int i = 0; i < MaterialCount; i++)
        {
                Item material = this.getPropertyItem("material_" + i.ToString());
                if (material == null)
                {
                        continue;
                }
                string MaterialName = material.getAttribute("name", "");
                string IsVisible = material.getAttribute("visibility", "");
                string Location = material.getAttribute("location", "");
                PwbServerApiObj.Log(
                        "PWBCus_SaveUsedMaterials -> material:'" +
                        MaterialName +
                        "', IsVisible:'" + IsVisible +
                        "', Location:'" + Location + "'");
        }

        Item result = getInnovator().newItem("result");
        PwbServerApiObj.AddMessageToUser("materials handled",
                                         PwbServerAddin.Base.MessageType.Information);


        return PwbServerApiObj.GetResultItem();
}
```

Management of CATIA Templates

TemplateFileAdmin

The PWB Configuration item setting “TemplateFileAdmin” defines the “Template File Administrator” identity.

Mandatory. Can be removed if the following server events are removed:

PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files

Picture 116: Template file administrator

CAD Documents with the attribute “is_template” having the value “1” and their related File items have to be imported to PDM. This can be done with a batch process, or with the PDM Workbench integration itself.

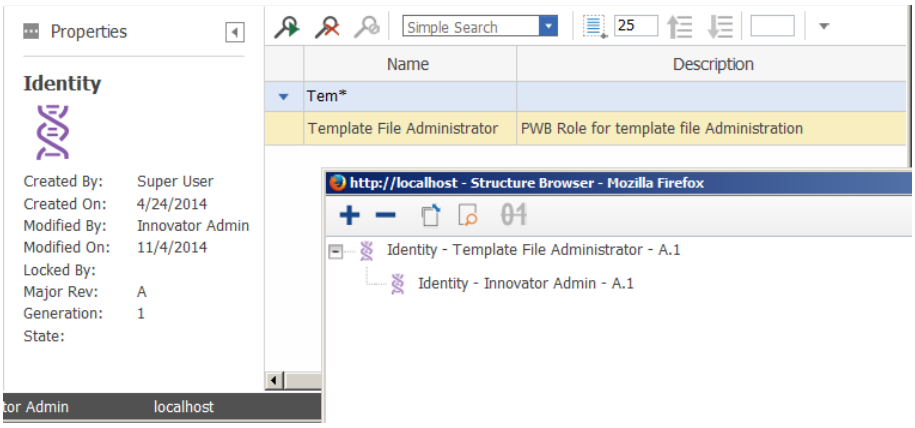
For this, the user has to be in a special template file administrator group.

This group can be configured as follows:

The PWB Configuration item setting “TemplateFileAdmin” has to contain the name of an identity with template file administrator access rights, for instance “Template File Administrator”:

TemplateFileAdmin	Template File Administrator

Picture 117: Sample TemplateFileAdmin configuration



Picture 118: The “Template File Administrator” identity

If this identity contains the user which is currently logged in, then the user is a template file administrator and has the rights to create and modify template file documents.

For regular users existing template file documents are read-only (black text background).

One way of creating template file documents in Aras Innovator with the PDM Workbench is this:

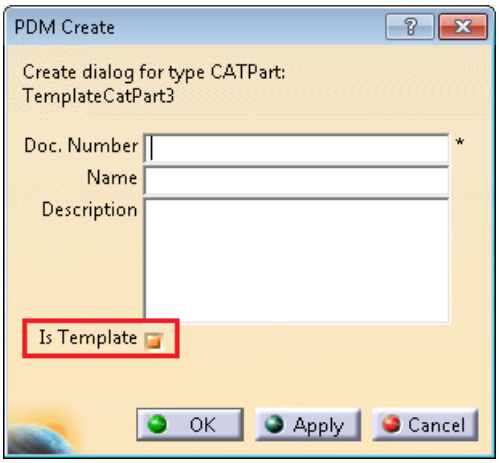
1. Make sure the following settings are set in the active PWB Configuration item, at least temporarily:

UseBomPartStructure	false
ShowCreateDialogsDuringUpdate	true

Picture 119: PWB Configuration variables for template creation

2. Log in to Aras Innovator with PDM Workbench as a template file administrator.
3. Open a CATIA file which is supposed to be used as a PDM template document in CATIA V5.
4. Click on the Update icon in the PDM Workbench toolbar to create the CAD Document and to upload the file.

Since the two settings are set to the values described in step 1 only a CAD Document will be created, and a “PDM Create” dialog is shown to the user:

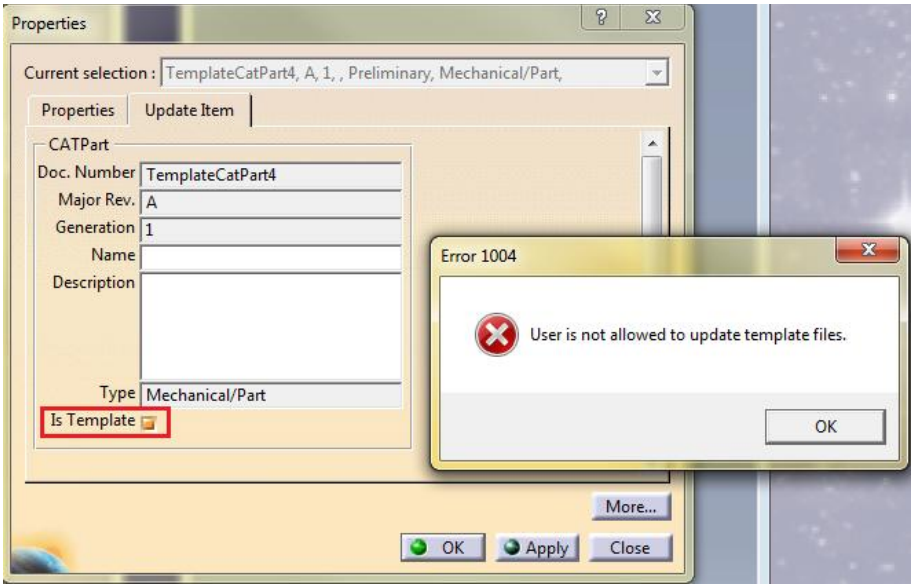


Picture 120: "PDM Create" dialog containing "Is Template" checkbox

The document number has to be entered, and it is important that the "Is Template" attribute is checked.

Another possibility would be to first create the template file as a regular file, and to change the "is_template" attribute afterwards. In this case the "ShowCreateDialogsDuringUpdate" setting does not have to be set to "true". Even the "UseBomPartStructure" setting can be set to "true", but in this case a Part item is created, too. This Part item may have to be deleted later, because it should probably not be in the bill of materials.

Only a template file administrator can change the "is_template" attribute value of a CAD Document item. A regular user will get this error message:



Picture 121: Template file update error message

For a regular user template file CAD Document items are read-only, and they also cannot be claimed by a regular user.

Client Configuration

In the Schema file the configuration *templateFiles* has to exist and it has to contain the attribute `loadFrom="PDM"`:

```
<templateFiles loadFrom="PDM"
               catPrt="PwbCatPrtTmplFileNames"
               catPrd="PwbCatPrdTmplFileNames"
               catDrw="PwbCatDrwTmplFileNames" />
```

The values of the XML attibutes *catPrt*, *catPrd*, and *catDrw* have to contain the names of data source definitions of the type “ValueList” which contain the file names of the CATPart, CATProduct, and CATDrawing template files:

```
<dataSource name="PwbCatPrtTmplFileNames" type="ValueList">
   <value name="TemplateCatPart1.CATPart" />
   <value name="TemplateCatPart2.CATPart" />
</dataSource>

<dataSource name="PwbCatPrdTmplFileNames" type="ValueList">
   <value name="TemplateCatProduct1.CATProduct" />
   <value name="TemplateCatProduct2.CATProduct" />
</dataSource>

<dataSource name="PwbCatDrwTmplFileNames" type="ValueList">
   <value name="TemplateCatDrawing1.CATDrawing" />
   <value name="TemplateCatDrawing2.CATDrawing" />
</dataSource>
```

The first file in each of these lists is the default template, and that template file is used when the template file functionality is switched on, and if a part structure with no related CAD Documents is loaded to CATIA.

The old template file functionality, where locally accessible files are used, is turned on by a definition like this in the Schema file:

```
<templateFiles loadFrom="C:\Users\Public\PDM-
Workbench\Templates"
               catPrt="PwbCatPrtTmplFileNames"
               catPrd="PwbCatPrdTmplFileNames"
               catDrw="PwbCatDrwTmplFileNames" />
```

In this definition the absolute file path which is defined as the value of the *loadFrom* XML attribute has to contain the CATIA files which are defined in the data sources which are referenced in the XML attributes *catPrt*, *catPrd*, and *catDrw*.

Optional.

If `loadFrom="PDM"` is set, then CAD Documents with the same document number (without the file extension) and the PDM attribute “is_template” having the value “1” have to be defined in Aras Innovator, and they have to be related to File items with the same file name (with the file extension).

Additionally, the “PDM Create” dialog definitions for the PDM types for CATPart, CATProduct, and CATDrawing files need to contain the form attribute for the template.

Here an example is given for the CATPart type:

```
<form name="Create">
   <formAttribute name="CatPrtTemplate" widgetType="ComboBox"
                  mode="update" visibleLength="30" required="false" />
</form>
```

The referenced attribute with the name “CatPrtTemplate” needs to contain two specific definitions: A data source which contains the names of the template files (as described above), and a specific information that this attribute contains information about CATPart template files (shown in red below).

```
<attribute name="CatPrtTemplate" displayName="NLS_CatPrtTemplate"
           dataSource="PwbCatPrtTmplFileNames"
           pwbAttrInfo="CatPrtTemplateListAttr" />
```

When the template files for one of the three types should not be used, e.g. no template support for CATDrawings, then the value “catDrw” has to be removed from the “templateFiles” tag and the form attribute for the template has to be removed from the “PDM Create” dialog.

Template File Support for “Create Part” with Templates depending on the Part Type

Originally the “Create” functionality only creates the “Part” business item in the database when a part type is selected, without a corresponding CAD Document item.

An extension of the “Create” functionality allows to create “Part” items with their corresponding CAD Document, and the native file of this CAD Document can be based on a list of template files which is defined specifically for this part type.

These Part items can also be created in the context of a parent assembly.

The “Create” functionality will create both the CAD Document and the corresponding part, with the part having the “default” classification, that is, the first entry in the part definition list in the Schema file with the fitting *canHaveSubParts* value:

```
<partClasses baseClassName="Part">
   <partClassName name="/Part/Design/Assembly/Mechanical"
                  canHaveSubParts="true" />
   <partClassName name="/Part/Design/Harness"
                  canHaveSubParts="true" />
   <partClassName name="/Part/Design/Component"
                  canHaveSubParts="false" />
   <partClassName name="/Part/Standard"
                  canHaveSubParts="false"/>
</partClasses>
```

Configuration

For the new functionality two new entries have to be defined in the global *settings* definition:

`CreatePartsWithCadFile`: “true” defines that when a BOM Part item is created a corresponding CAD Document item with a native CATIA file is also created. This CATIA file can be based on a template file.

`ContextCreateParts`: “true” defines that the “PDM Create in Context” functionality includes part types too, not only the CATProduct and CATPart types.

```
<settings>
   <setting name="CreatePartsWithCadFile" value="true" />
   <setting name="ContextCreateParts" value="true" />
</settings>


<object name="Part" displayName="NLS_Part" icon="Aras_Part">
   <description>
```

```
   ...
```

Also, if specific template files should be associated with a part type, an object-specific setting with the name “TemplateFileNameDataSource” and the value of a string list data source should be defined. The list should contain the names of the template CATIA files which correspond to this part type:

```
<object name="/Part/Design/Assembly/Mechanical"
        displayName="NLS_Assembly" icon="Aras_Part" >
   <settings>
      <setting name="TemplateFileNameDataSource"
               value="TmplFileNames-Assembly-Mechanical" />
   </settings>
   ...
</object>


<object name="/Part/Design/Harness"
        displayName="NLS_Harness" icon="Harness" >
   <settings>
      <setting name="TemplateFileNameDataSource"
               value="TmplFileNames-Harness" />
   </settings>
...
</object>


<object name="/Part/Design/Component"
        displayName="NLS_Component" icon="Aras_Component" >
   <settings>
      <setting name="TemplateFileNameDataSource"
               value="TmplFileNames-Component" />
   </settings>
   ...
</object>
```

The data sources should also be defined in the Schema file:

```
<!-- All CATProduct template files -->
<dataSource name="PwbCatPrdTmplFileNames" type="ValueList">
   <value name="TemplateCatProduct1.CATProduct" />
   <value name="TemplateCatProduct2.CATProduct" />
   <value name="Assembly-Mechanical-1.CATProduct" />
   <value name="Assembly-Mechanical-2.CATProduct" />
   <value name="Harness-1.CATProduct" />
   <value name="Harness-2.CATProduct" />
</dataSource>


<!-- All CATPart template files -->
<dataSource name="PwbCatPrtTmplFileNames" type="ValueList">
   <value name="TemplateCatPart1.CATPart" />
   <value name="TemplateCatPart2.CATPart" />
   <value name="Component-1.CATPart" />
```

```
   <value name="Component-2.CATPart" />
</dataSource>

<!-- Template files specific to part types -->
<dataSource name="TmplFileNames-Assembly-Mechanical" type="ValueList">
   <value name="Assembly-Mechanical-1.CATProduct" />
   <value name="Assembly-Mechanical-2.CATProduct" />
</dataSource>

<dataSource name="TmplFileNames-Harness" type="ValueList">
   <value name="Harness-1.CATProduct" />
   <value name="Harness-2.CATProduct" />
</dataSource>

<dataSource name="TmplFileNames-Component" type="ValueList">
   <value name="Component-1.CATPart" />
   <value name="Component-2.CATPart" />
</dataSource>
```

Thumbnail Configuration

Create Thumbnails from Types

The PWB Configuration item setting “CreateThumbnailsFromTypes” is a list of file extensions for which the thumbnails should be created by update, separated by “|”.

CreateThumbnailsFromTypes	.CATPart|.CATDrawing

Picture 122: Sample CreateThumbnailsFromTypes configuration

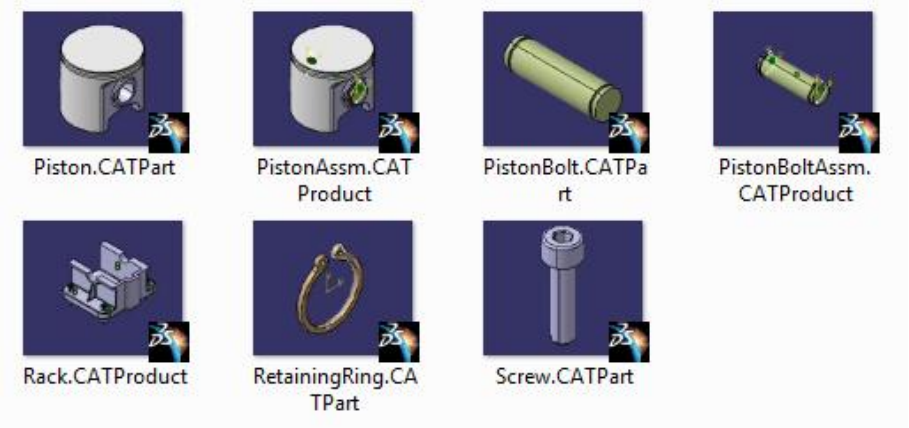
Default value: “.CATPart|.CATDrawing”

Optional.

Create Thumbnails by CATIA

It is possible to let CATIA generate the thumbnail pictures internally, instead of using the pictures used by Windows.

This is an example of thumbnails of CATIA files in the Windows Explorer:



Picture 123: Windows Thumbnails

In some cases, due to some configuration of the Windows registry, the thumbnails in the Windows Explorer are not shown correctly. In this case there will be no thumbnails associated to the CAD Documents in PDM either.

With this setting the thumbnail pictures are created inside CATIA, so they are independent of the Windows setting.

In addition, with this setting it is possible to have thumbnail pictures for CGR and CATIA V4 files too, which would not be possible otherwise.

Configuration in the Schema file.

Example:

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

Default value: “false”

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

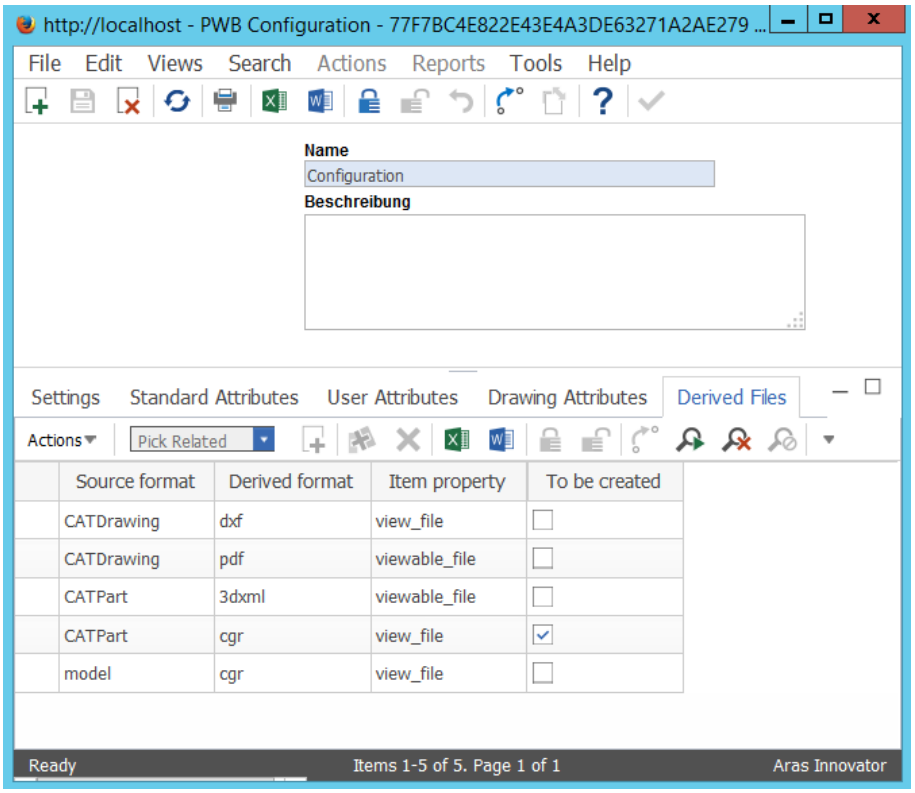
Derived Files Configuration

Derived Files Tabulator

There is a “Derived Files” tab in the PWB Configuration item. The customer can define there the derived file formats to be created per source format and its storage property.

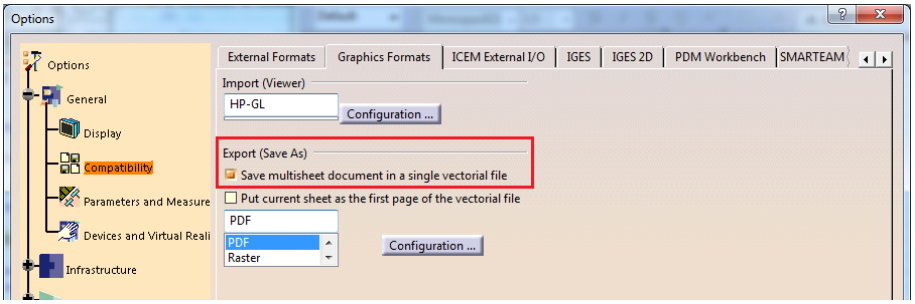
The storage property must be a File item property, which is defined on the CAD Document item, like “viewable_file” or “view_file”.

The PDM Workbench provides the capability to create and store derived files when a user is uploading new or modified CAD files to Aras Innovator. These derived files are created in a CATIA SaveAs operation of the current PDM Workbench user’s CATIA session. Take care, that each defined SaveAs operation results in a single file with the same name as the source file but a different suffix. If CATIA creates multiple files from a single CATIA file during a SaveAs operation, then this type is not supported here.



Picture 124: "Derived Files" tab

Example for single generated PDF file: Only one PDF for a CATDrawing can be stored in Aras Innovator as viewable_file (derived file). Therefore, all sheets of a CATDrawing must be stored in a single PDF. The following CATIA setting enables this option:



Picture 125: Activate multisheet option in CATIA V5
Tools→Options→General→Compatibility→Graphics Formats

Please note that the old configuration settings "UpdateDerivedPdfFile" and "UpdateDerived3DxmlFile" are not supported anymore.

Additional CAD Document Types

Design Tables

The PWB Configuration item setting "UseDesignTables" configures the use of Design Tables with the PDM Workbench. The use of Design Tables in Aras Innovator can be enabled or disabled in PDM Workbench.

Please note that when Design Tables are disabled from PDM Workbench, they can still be used in CATIA, but they are not synchronized and uploaded to Aras Innovator.

UseDesignTables	true

Picture 126: Sample UseDesignTables configuration

Default value: “false”

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

When design table files are created in PDM they can be renamed according to the CATIA file that they are related to. It is also possible to relate one design table document to multiple CATPart documents.

Configuration

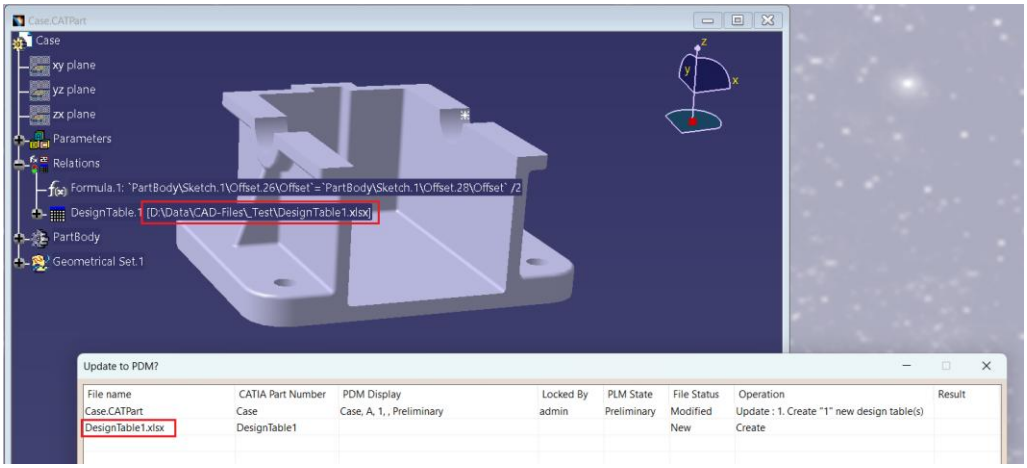
By default, the first design table document will be named like the CAD Document it is related to, and additional design table documents will be named like the CAD Document plus “_” plus an index (1, 2, 3, etc.).

It is possible to customize this behavior by setting a custom method for the server setting “CustomMethod_PreProcCreDlgAttrs”. A sample method for this (“Pwb_Sample_PreProcDlgAttrs”) exists.

Usage

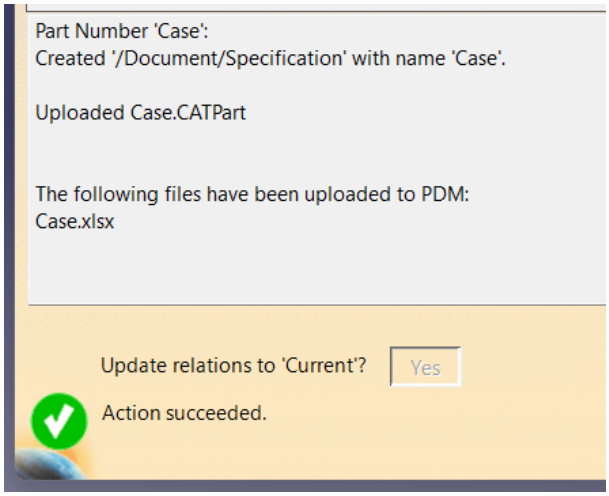
In this example two design tables are related to the file “Case.CATPart”.

The first design table file is originally named “DesignTable1.xslx”:



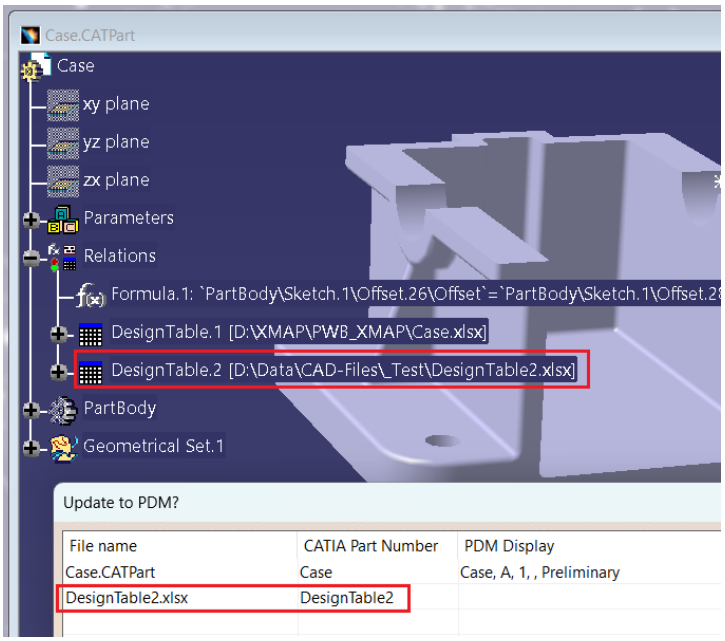
Picture 127: Adding first design table

PDM update renames the design table file to “Case.xlsx”:



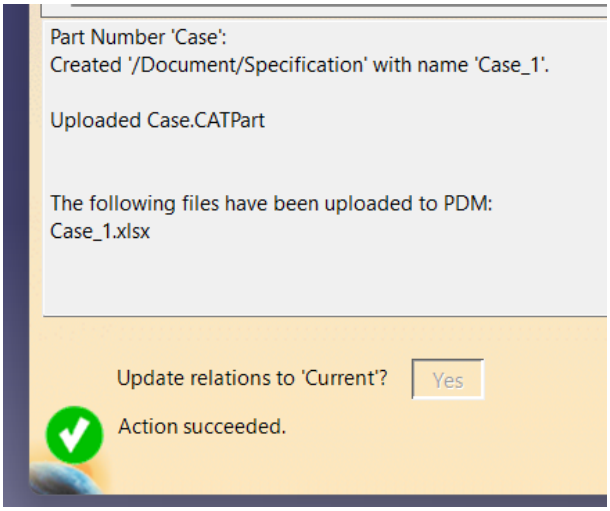
Picture 128: Renamed first design table file.

A second design table file (“DesignTable2.xlsx”) is related:



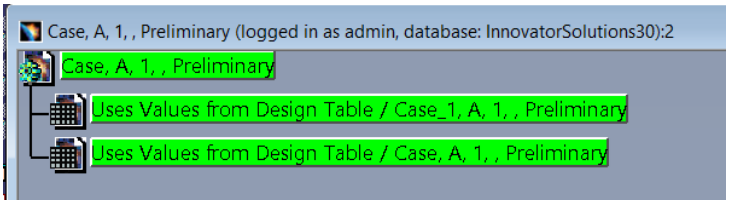
Picture 129: Adding second design table

PDM update renames the design table file to “Case_1.xslx”:

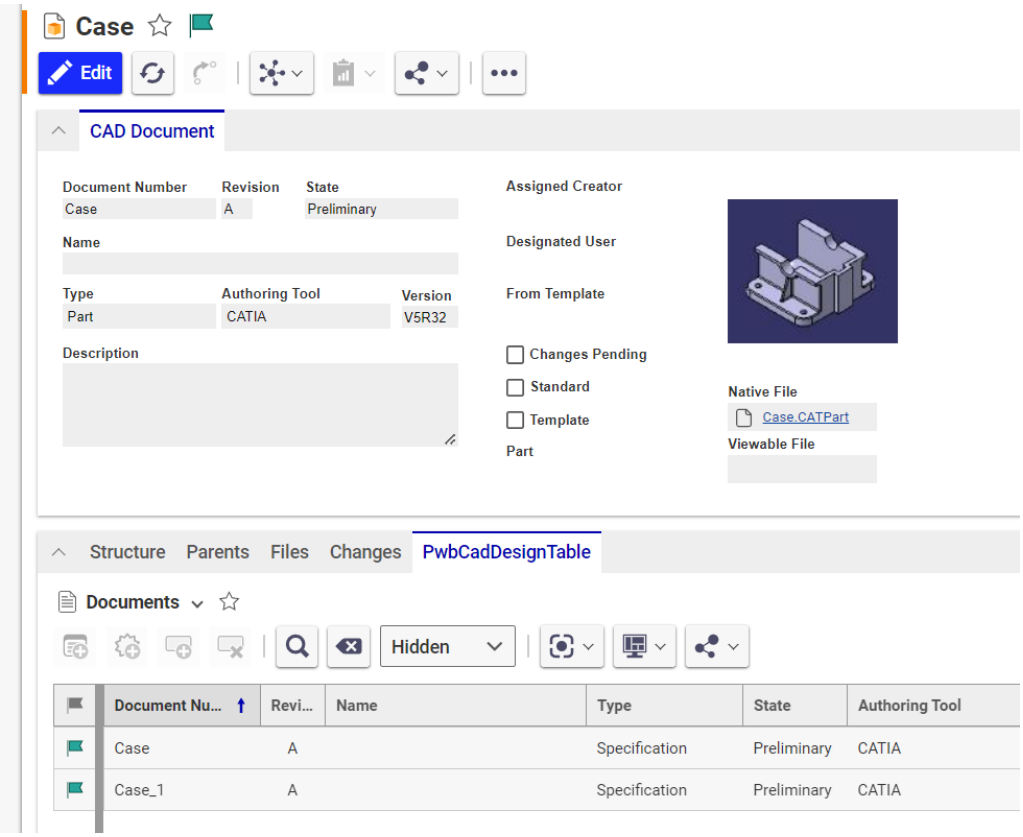


Picture 130: Renamed second design table file.

Both design table documents are related in PDM:



Picture 131: Display of related design tables in the PWB structure window



Picture 132: Display of related design tables in the Aras Innovator web client

Archives

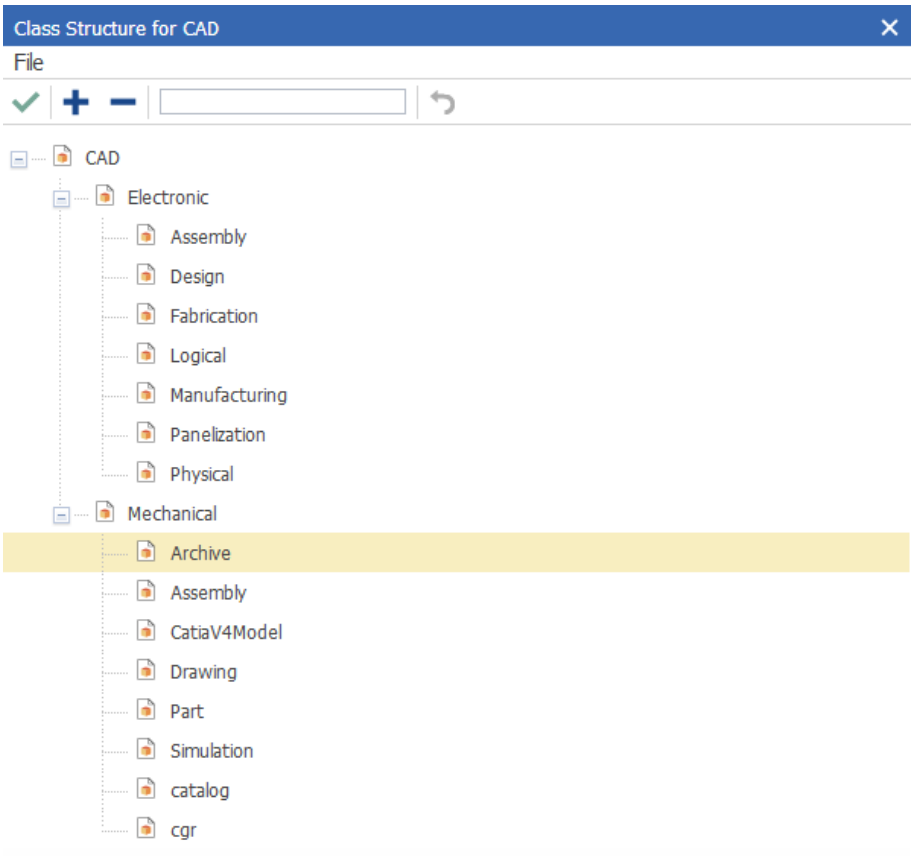
It is possible to compress a complete CATProduct sub-structure into one Zip file and to manage this compressed file in PDM. This makes it possible to hide a complicated CATProduct structure in one CAD Document if it is not necessary to manage the structure information in PDM.

The PWB Configuration item setting “UseArchives” has to be set to “true”:

UseArchives	true

Picture 133: Sample UseArchives configuration

If archives are used a new classification for the Item Type “CAD” needs to be defined, for instance “Mechanical/Archive”.



Picture 134: Add Mechanical/Archive to CAD

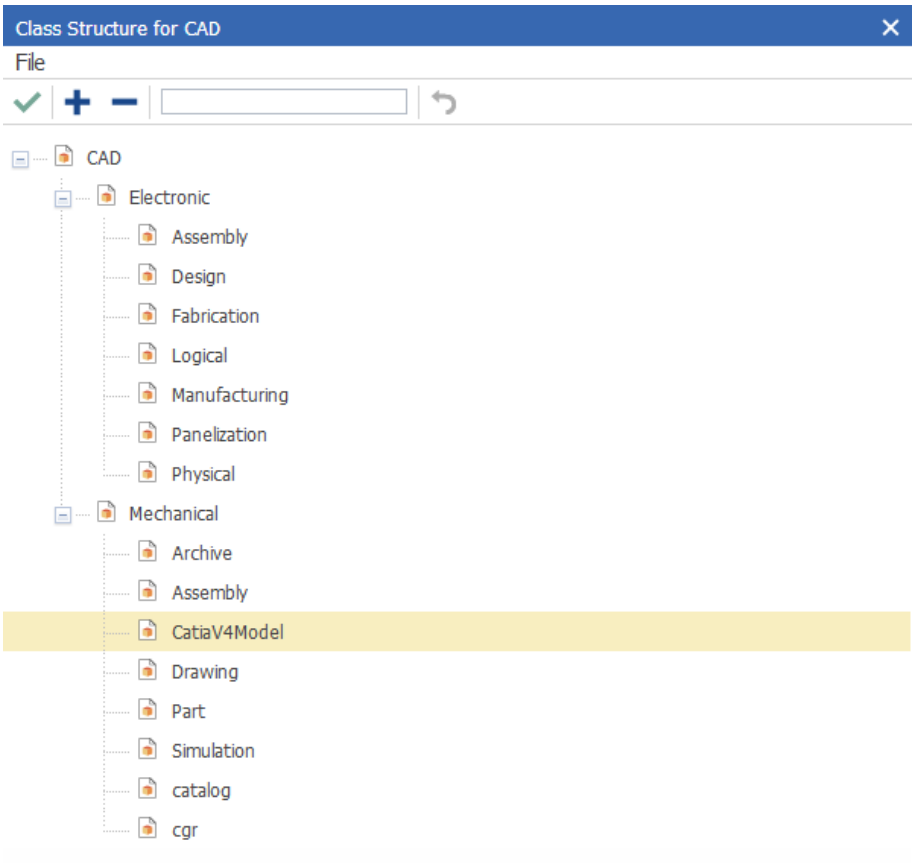
In the Schema file the same classification has to be defined.

Example:

```
<object name="/CAD/Mechanical/Archive"
        displayName="NLS_Archive"
        icon="Aras_Archive">
```

CATIA V4 Models

If structures which contain CATIA V4 models are used a new classification for the Item Type “CAD” needs to be defined, for instance “Mechanical/CatiaV4Model”.



Picture 135: Add Mechanical/CatiaV4Model to CAD

In the Schema file the same classification has to be defined.

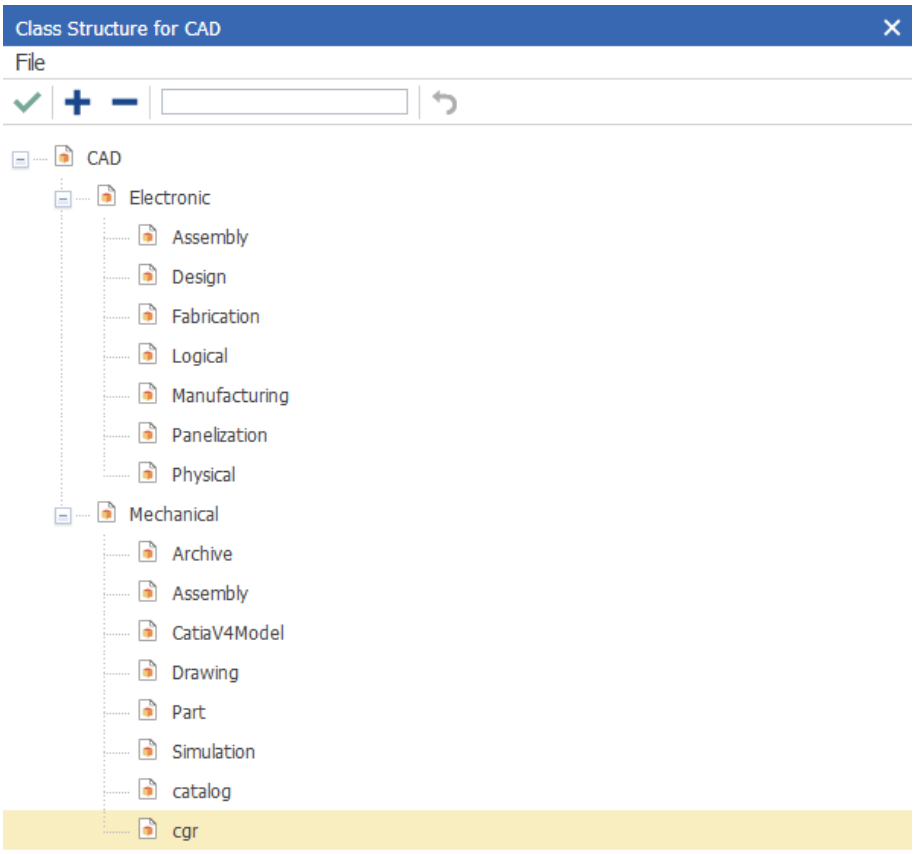
Example:

```
<object name="/CAD/Mechanical/CatiaV4Model"
        displayName="NLS_model"
        icon="model">
```

CGRs as native Files

CGR files are fully supported as native files, including assigning them a new CATIA Part Number, PDM attribute mapping, “Insert PDM Node” functionality, and replacing the representation with a different file and updating the new representation.

A new classification for the Item Type “CAD” needs to be defined, for instance “Mechanical/cgr”.



Picture 136: Add Mechanical/cgr to CAD

In the Schema file the same classification has to be defined.

Example:

```
<object name="/CAD/Mechanical/cgr"
        displayName="NLS_cgr"
        icon="cgr">
```

CATIA Catalogs

CATIA Catalogs for CATParts are supported. The catalogs can be created and updated by a “Standard Part Administrator”. The Catalog functionality supports CATParts used as “Standard Parts”, as “Templates” or CATParts holding a “Power Copy”.

The Catalog functionality adds the catalog keywords PWB_CAD and for Bom CATParts PWB_PART. The values of these keywords must not be changed by a user.

During opening a catalog in the CATIA Catalog Editor or Catalog Browser using the PDM Workbench, placeholder files for the referenced CATParts in Aras Innovator are automatically created in the PDM Workbench exchange directory. The actual geometry of a referenced CATPart will be fetched from Aras Innovator as soon as the geometry is needed by the native CATIA Catalog functionality.

In the Schema file the context actions can be enabled.

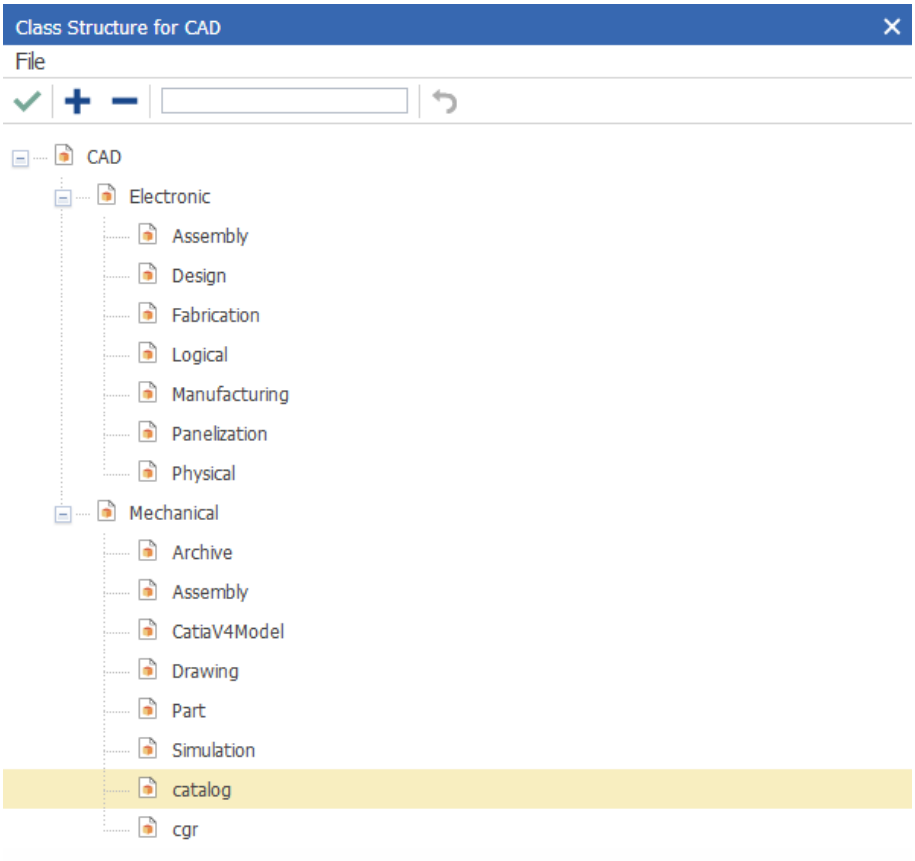
Example:

```
<contextAction name="OpenCatalog" usedIn="QueryDialog"
               defaultAction="true" />
```

The element `defaultAction="true"` defines this context action as action to be performed by double-clicking on an object of this type.

```
<contextAction name="OpenCatalogForEdit" usedIn="PdmWindow|QueryDialog"
/>
```

A new classification for the Item Type “CAD” needs to be defined, for instance “Mechanical/catalog”.



Picture 137: Add Mechanical/catalog to CAD

In the Schema file the same classification has to be defined.

Example:

```
<object name="/CAD/Mechanical/catalog"
        displayName="NLS_catalog"
        icon="catalog">
```

For this functionality the comment from the last line of the file

```
…\win_b64\code\dictionary\PWBModel.dico
```

in the connector client installation has to be removed:

```
# CATUExitIO    CATIUExitIO    libPWBModelLib
```

➔

```
CATUExitIO    CATIUExitIO    libPWBModelLib
```

Support floating Content in Catalog

By default, the catalog points to a fixed Aras Innovator version of the contained Aras Innovator CAD (Part). If this functionality is enabled, you will get the latest version of the corresponding CAD (Part).

Configuration

In the PWB Configuration you have to add the setting “UseFloatingCatalogContent”:

UseFloatingCatalogContent	true

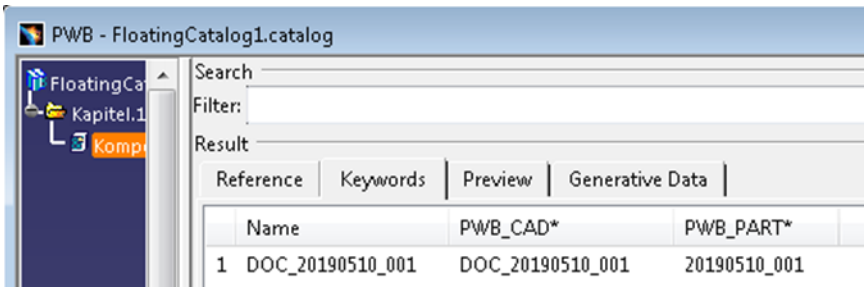
Picture 138: Sample UseFloatingCatalogContent configuration

Default value: “false”

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

Configurable Catalog Keywords

By default the identifying attributes (id or item_number when using floating catalog content) are stored in the values of PWB_CAD and PWB_PART (for BOM Part).



Picture 139: Configurable Catalog Keywords

It is possible to configure different catalog keywords to store these values. In case of `useBomPartStructure = true`, if only BOM CATParts are used in the catalog, it is also possible to remove the keyword for the CAD.

Configuration

The changes are done in the PWB Configuration.

To change the PWB_PART keyword you have to add the setting “CatalogPartIdentifier”:

CatalogPartIdentifier	<Your Keyword>

Picture 140: Sample CatalogPartIdentifier configuration

Default value: “PWB_PART”

Optional.

To change the PWB_CAD keyword you have to add the setting “CatalogCadIdentifier”:

CatalogCadIdentifier	<Your Keyword>

Picture 141: Sample CatalogCadIdentifier configuration

Default value: “PWB_CAD”

Optional.

If “CatalogCadIdentifier” is set to an empty value, the keyword is not used (only for useBomPartStructure = true).

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 item setting “CustomMethod_ProcessCatalogContent” must point to the method name of your custom method:



Picture 142: 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 the CAD Document Structure Data Model, 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
*/
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
   string CAD_Id = this.getID();
   string CAD_Type = this.getType();
   string CAD_Classification = this.getProperty("classification");

   PwbServerApiObj.Log(
      "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;
      }

      PwbServerApiObj.Log(
         "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;
      }

      PwbServerApiObj.Log(
         "PwbCus_ProcessCatalogContent -> referencedCad:' " +
         referencedCad);
   }
```

```
    Item result = getInnovator().newItem("result");
    result.setAttribute("message", "Catalog Content handled ");

    return result;
}
```

CATProcess

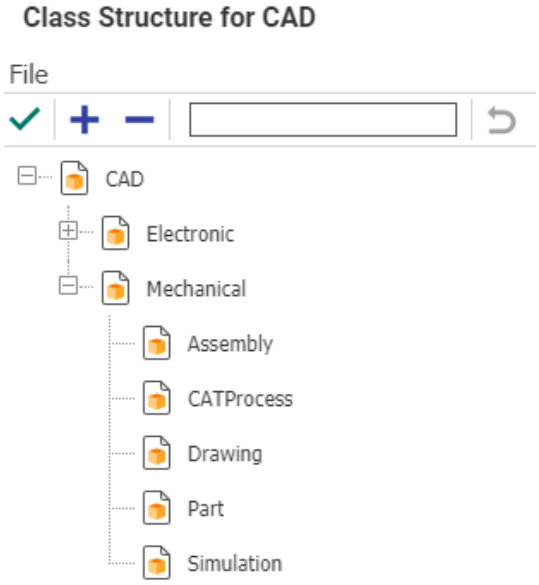
CATProcess can be updated and loaded from Aras Innovator. If the CATProcess uses external references in the CATProcess Product List or in the CATProcess Resources List, PDM Workbench creates a relation to the refenced items during update and downloads the referenced items during load.

Configuration

To use CATProcess documents with the PDM Workbench, the Aras Innovator classifications of CAD and CAD Structure must be extended. As the Schema file already holds the necessary object and relation definition, it is necessary to use the same names or also to change the CATProcess definition in the Schema file.

Item Type “CAD”:

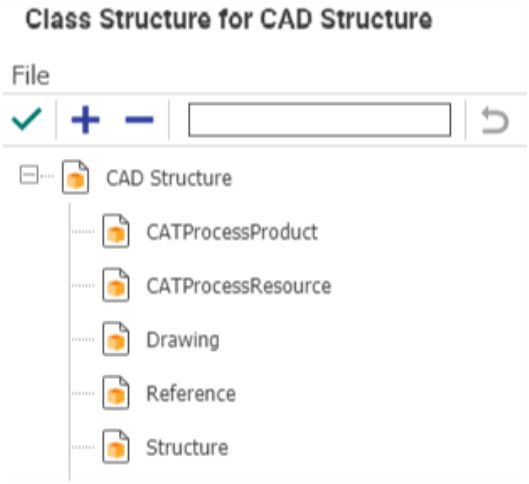
Add the subclass CATProcess to /CAD/Mechanical:



Picture 143: Subclass /CAD/Mechanical/CATProcess

Item Type “CAD Structure”

Add the subclasses CATProcessProduct and CATProcessResource to CADStructure:



Picture 144: Subclasses /CAD Structure/CATProcessProduct and /CAD Structure/CATProcessResource

To use a different relation for the CATProcess ProductList you must add the setting “CadProcessProductRelType” to the PWB Configuration.

Default value: “/CAD Structure/CATProcessProduct”.

If “CadProcessProductRelType” is set to an empty value, the CATProcess ProductList will be ignored by PDM Workbench.

If you use a different value must also change the value in the Schema file.

To use a different relation for the CATProcess ResourcesList you must add the setting “CadProcessResourcesRelType” to the PWB Configuration.

Default value: “/CAD Structure/CATProcessProduct”.

If “CadProcessResourcesRelType” is set to an empty value, the CATProcess ResourcesList will be ignored by PDM Workbench. This can be useful if you manage the tools in the CATIA environment.

If you use a different value must also change the value in the Schema file.

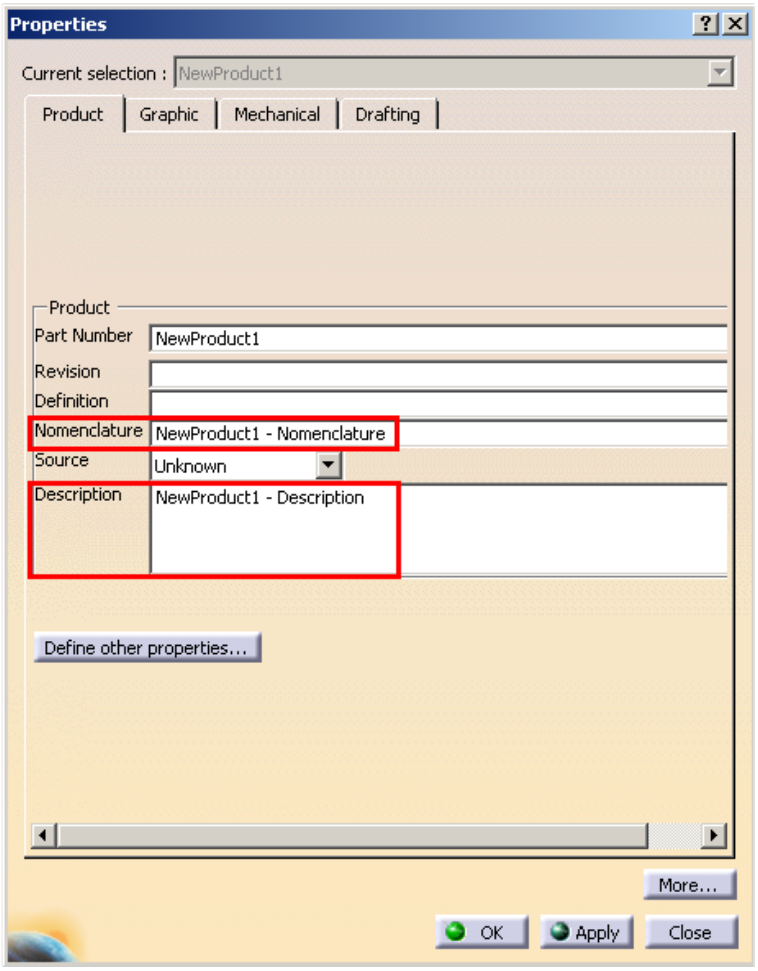
Attribute Mapping Configuration

Standard Attribute Mapping

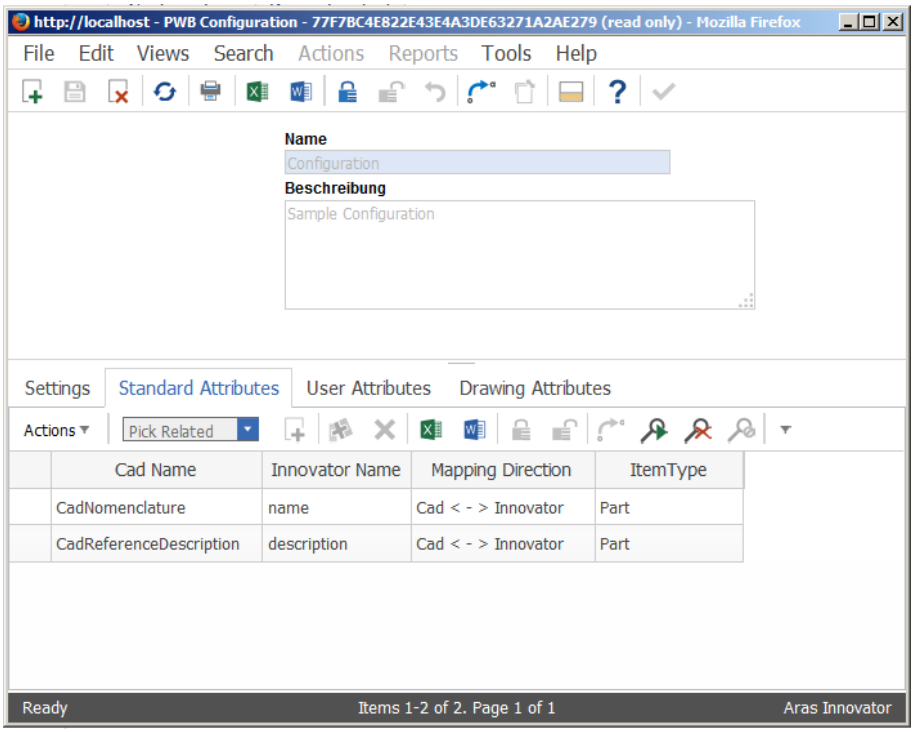
Here is a description of how to configure the attribute mapping:

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

In the following example the standard CATIA attributes “Nomenclature” and “Description” are mapped to the PDM attributes “name” and “description” of the Aras Innovator part object (see *Picture 145: Standard attributes in the “Properties” dialog* and *Picture 146: Configuration of standard attributes in Aras Innovator*).

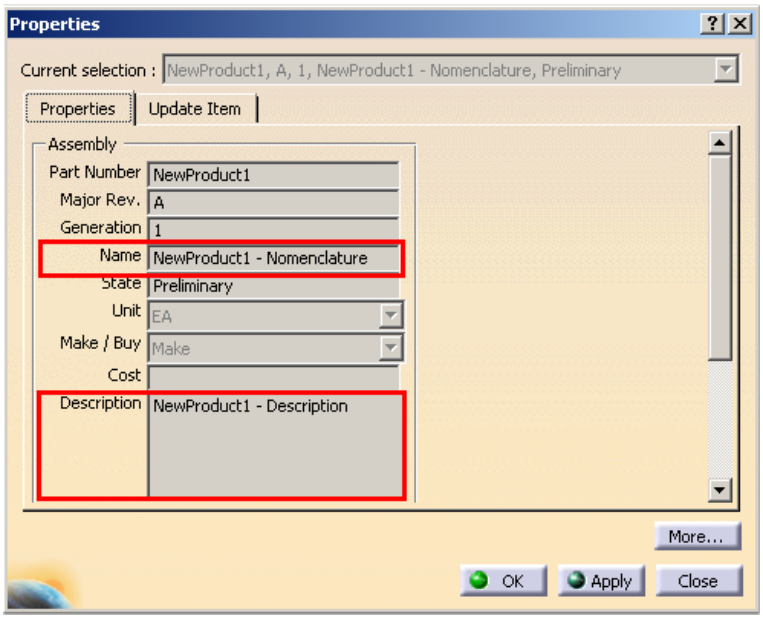


Picture 145: Standard attributes in the "Properties" dialog

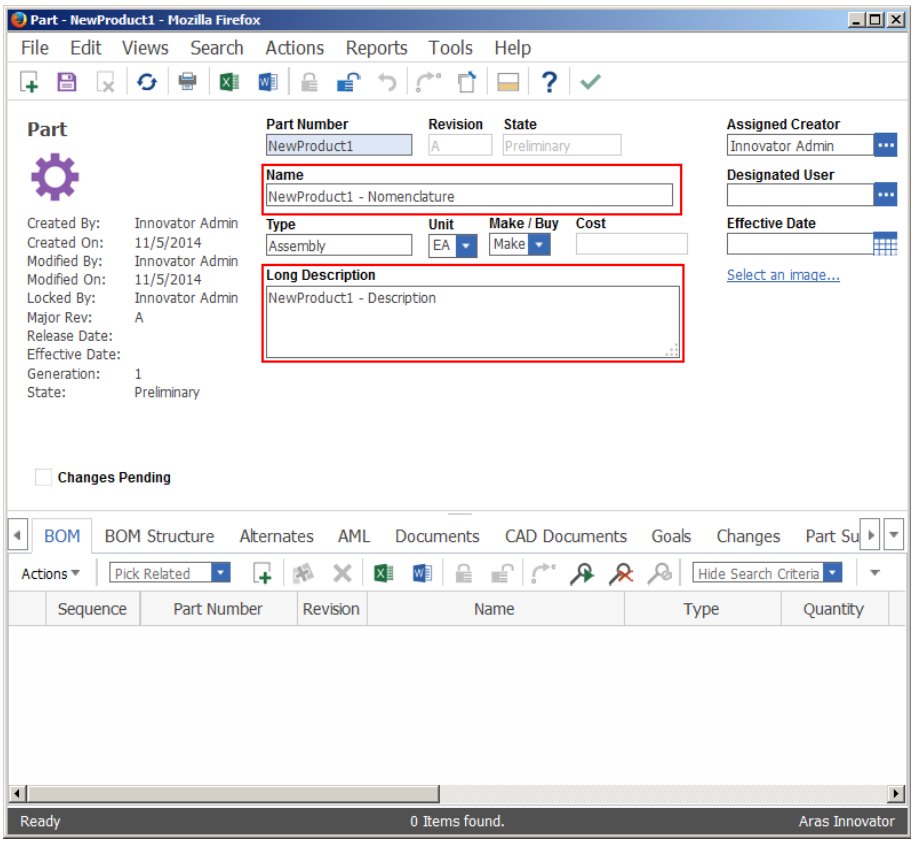


Picture 146: Configuration of standard attributes in Aras Innovator

After creating the part with Update the defined CATIA attribute values have been written to the PDM part object (see *Picture 147: Standard attributes in the "Properties" dialog of the PDM node* and *Picture 148: Standard attributes in Aras Innovator window*).



Picture 147: Standard attributes in the "Properties" dialog of the PDM node

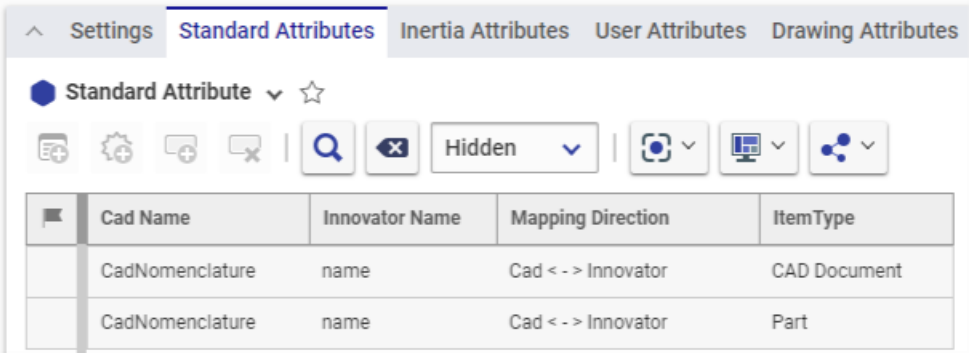


Picture 148: Standard attributes in Aras Innovator window

Allow Mapping of Part and CAD Property to the same CATIA Standard Attribute

By default, you can either map a Part or a CAD property to a CATIA standard attribute like Nomenclature, Definition, Revision, and Reference Description. This causes a problem if you work in BOM Part Structure Data Model with additional Non-Bom CATIA files. For instance, if you map the value of the Part property “name” to the CATIA attribute “Nomenclature” you cannot map any CAD property to the same CATIA attribute. This means the CATIA attribute “Nomenclature” cannot be controlled by any Aras Innovator property for a Non-BOM CATIA file.

This functionality allows to define a property mapping for both types:



Picture 149: Mapping of CATIA attribute Nomenclature from CAD and Part

The mapping of the Part property is handled with a higher priority. This means only if there is no Part, the mapping of the CAD will be used.

The mapping direction from CATIA to Aras Innovator is not affected.

Configuration

This behavior is only active when using the following PWB settings:

`UseBomPartStructure=true`

`UseNonBom3DCadDocs=true`

To disable the functionality, you have to set `DisableUniqueStdAttributeCheck=false` in the PWB configuration.

Possibility to use CATIA User-Defined Properties in Autoname and PreProcDlgAttrs Custom Methods

It is possible to use user-defined properties in the autoname method and the customization method defined by the setting ‘CustomMethod_PreProcCreDlgAttrs’.

Here is how to retrieve them in the autoname method (Pwb_Sample_AutonameMethod):

```
// Standard CATIA Properties
Item CadStdPropsItem = this.getPropertyItem("CadStdProps");
IDictionary<string, string> CadStdPropsDict = null;
if (CadStdPropsItem != null)
{
        CadStdPropsDict =
                PwbServerApiObj.DialogAttrsItemToDictionary(
                        CadStdPropsItem);
}

// NEW: User-defined CATIA Properties
Item CadUserDefPropsItem = this.getPropertyItem("CadUserDefProps");
IDictionary<string, string> CadUserDefPropsDict = null;
if (CadUserDefPropsItem != null)
{
        CadUserDefPropsDict =
                PwbServerApiObj.DialogAttrsItemToDictionary(
```

```
                        CadUserDefPropsItem);
}
```

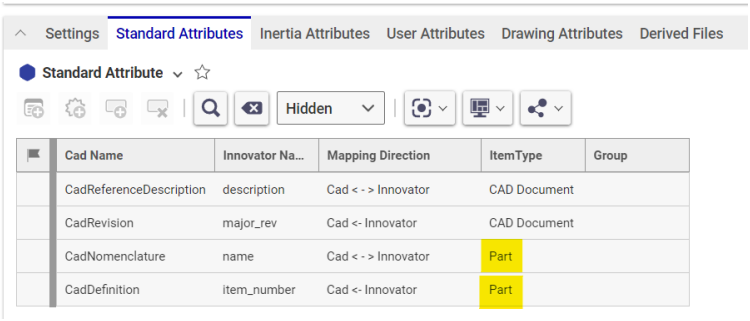
Here is how to retrieve them in Pwb_Sample_PreProcDlgAttrs:

```
// CATIA user-defined properties
IDictionary<string, string> CatiaUserDefPropsDict = null;
Item CatiaUserDefPropsItem =
        this.getPropertyItem("CadUserDefProps");
if (CatiaUserDefPropsItem != null)
{
        CatiaUserDefPropsDict =
                PwbServerApiObj.DialogAttrsItemToDictionary(
                        CatiaUserDefPropsItem);
}
if (CatiaUserDefPropsDict != null)
{
        foreach (KeyValuePair<string, string> Attribute in CatiaUserDefPropsDict)
        {
                PwbServerApiObj.Log(
                        "CATIA user-defined property '" + Attribute.Key +
                        "'/'" + Attribute.Value + "'");
        }
}
```

Query / Expand Performance CAD Structure mode with Part attribute mapping

When using attribute mapping from a related Aras Part to the file in CATIA, use a set based expand to get the related Parts during query.

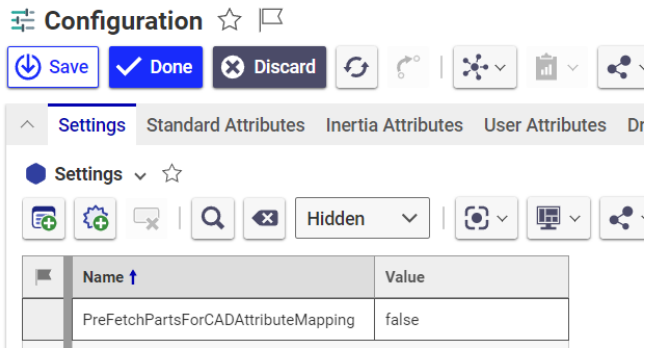
The feature is only relevant if you are in CAD Structure mode (UseBomPartStructure=false) and you have mappings from Part to Cad in “Standard Attributes”, “User Attributes” or “Drawing Attributes”:



Configuration

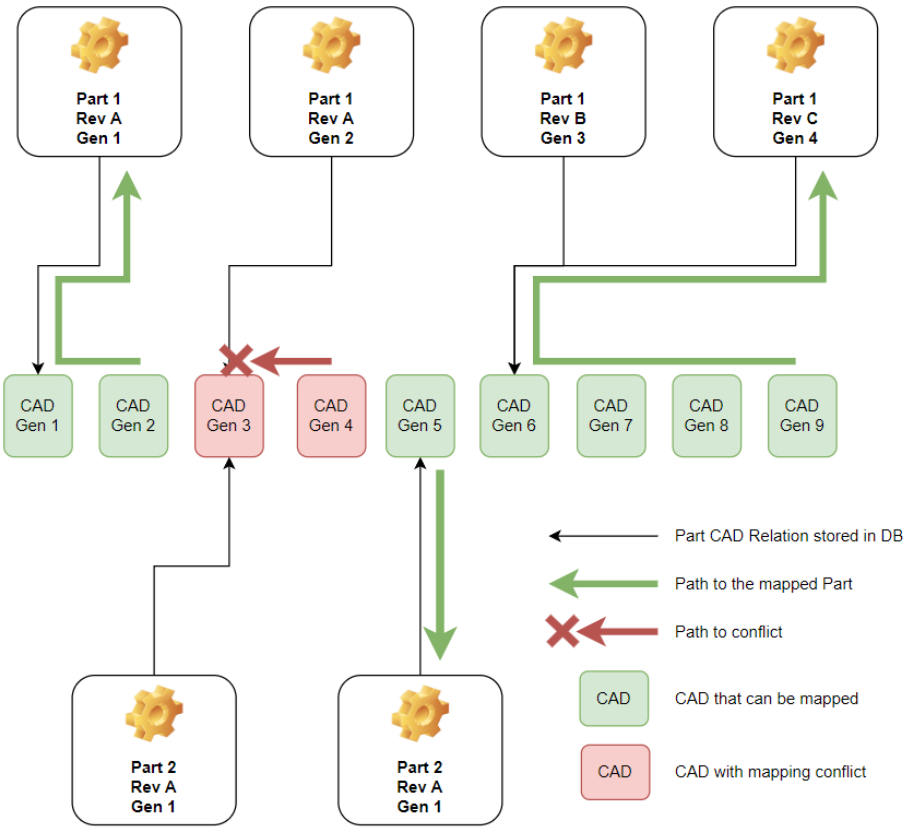
This behaviour is enabled by default. It can be disabled with the PWB configuration setting:

PreFetchPartsForCADAttributeMapping = false (default value: true)



Behavior

When activated, the set based expand uses the following behaviour to get the related Part for a specific generation of each CAD:



If a generation of a CAD is attached to two different Parts, there is no attribute mapping available for this generation of the CAD.

If a generation of a CAD is attached to multiple generations of the same Part, the most recent generation of the Part is used.

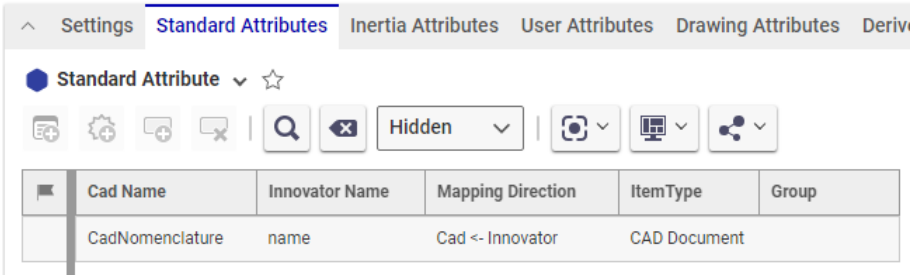
If a generation of a CAD is not directly attached to a Part, use the Part of the last generation before the inspected one which is attached to the Part.

Not setting mapped "CATIA Standard Attribute" Values in the "PDM Create" Dialog

By default all attributes that are mapped to a CATIA Standard attribute (CadNomenclature, CadRevision, CadDefinition, CadReferenceDescription) are prefilled in the "PDM Create" dialog with the current CATIA value. To disable the mapping set `pwbAttrInfo="DoNotUseCatiaValue"` to the formAttribute.

Example:

The Aras Innovator attribute "name" is mapped to the CATIA Standard attribute "CadNomenclature" for the Item Type "CAD Document":

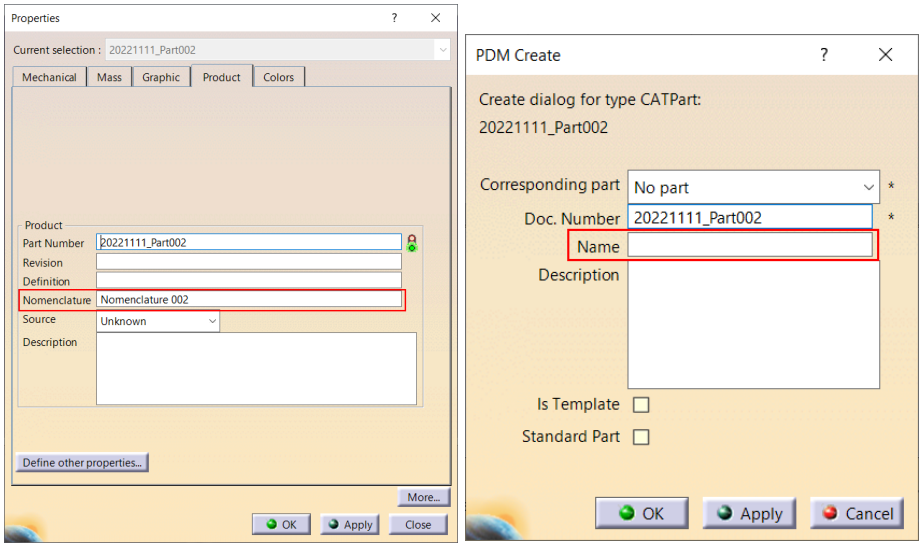


Picture 150: PWB Configuration – Standard Attribute Mapping

In the Schema file is defined, that the Aras Innovator attribute "name" should not be mapped to CATIA V5:

```
<formAttribute name="name" widgetType="SingleLineEditor"
               mode="update" visibleLength="15"
               required="false"
               pwbAttrInfo="DoNotUseCatiaValue" />
```

As result you can see that the defined Nomenclature of CATIA V5 is not pre-filled in the "PDM Create" dialog:



Picture 151: CATIA V5 Properties of the CATPart and the "PDM Create" dialog

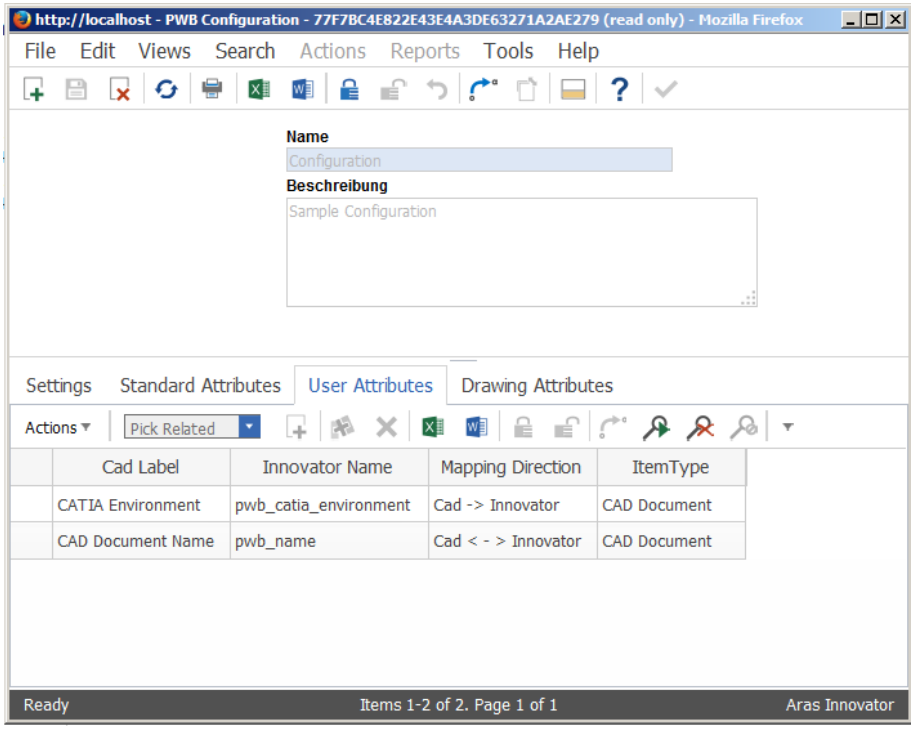
Prefill CATIA Revision Attribute in "Register" / "PDM Create" Dialog

If the Standard Attribute CadRevision is mapped to an Aras Innovator attribute like "major_rev" in the PWB Configuration object, and the "major_rev" attribute is shown in the "PDM Create" dialog, the dialog will be prefilled with the value of the CATIA Revision attribute. If a "data source" in the Schema file restricts the possible values, only a valid value will be mapped.

```
<formAttribute name="major_rev" widgetType="ComboBox" mode="update"
               visibleLength="15"   required="true" entryAllowed="false"
               dataSource="RevisionDataSource"
               pwbAttrInfo="DoNotUseCatiaValue" />
```

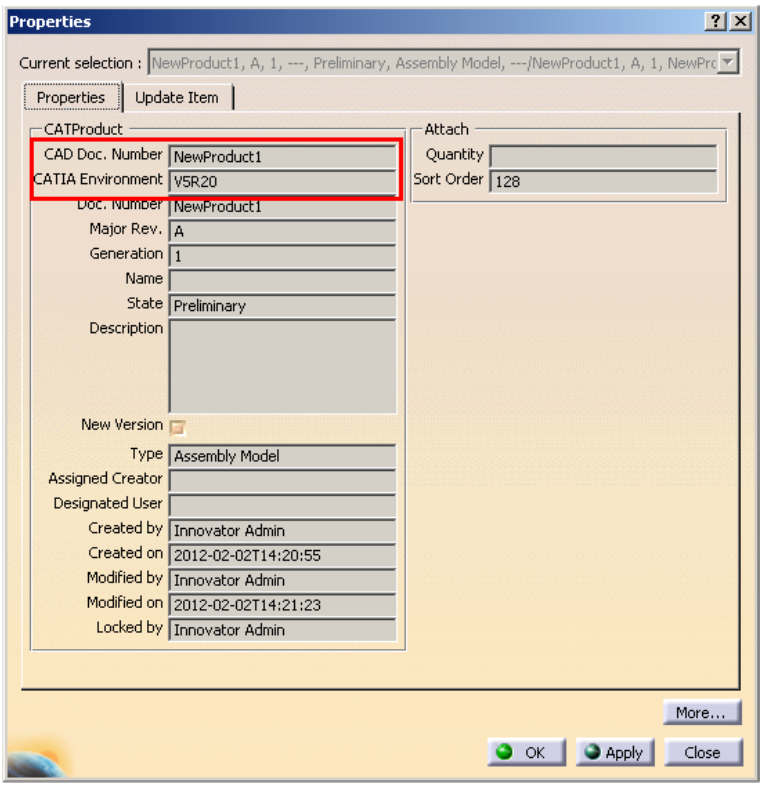
User Attribute Mapping

User-defined CATIA properties can also be mapped (see *Picture 152: Configuration of user-defined attributes in Aras Innovator*).

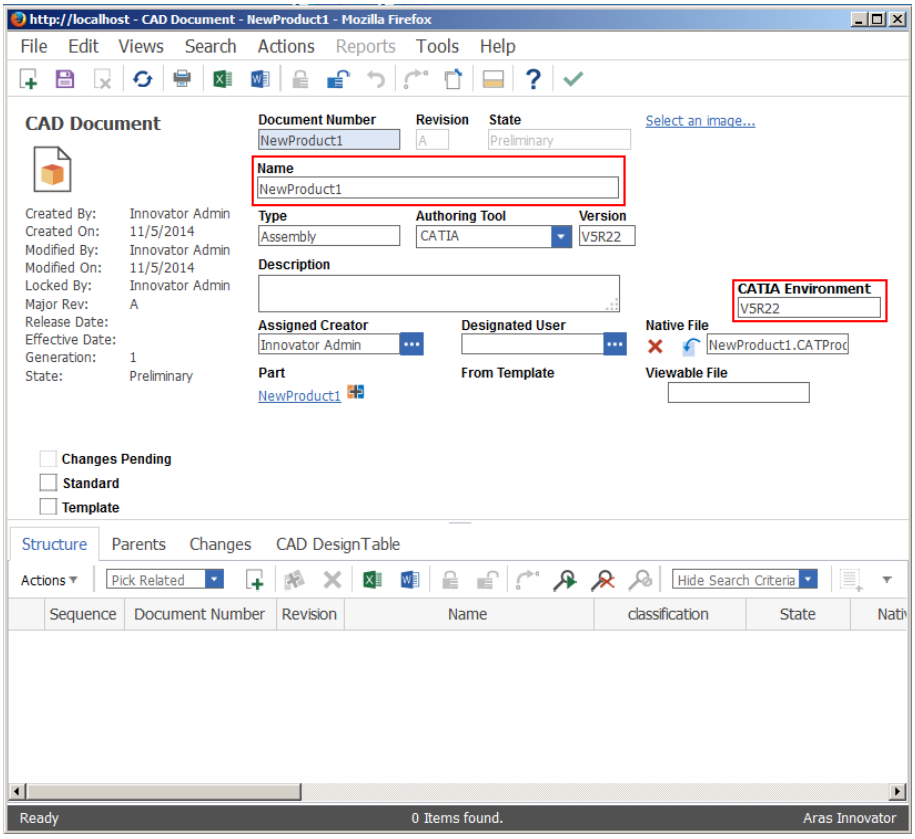


Picture 152: Configuration of user-defined attributes in Aras Innovator

While the structure is imported the values are written to the defined attributes of the Aras Innovator CAD Document object (see and *Picture 153: User-defined attributes in the "Properties" dialog of the PDM node* and *Picture 154: User-defined attributes in Aras Innovator window*).

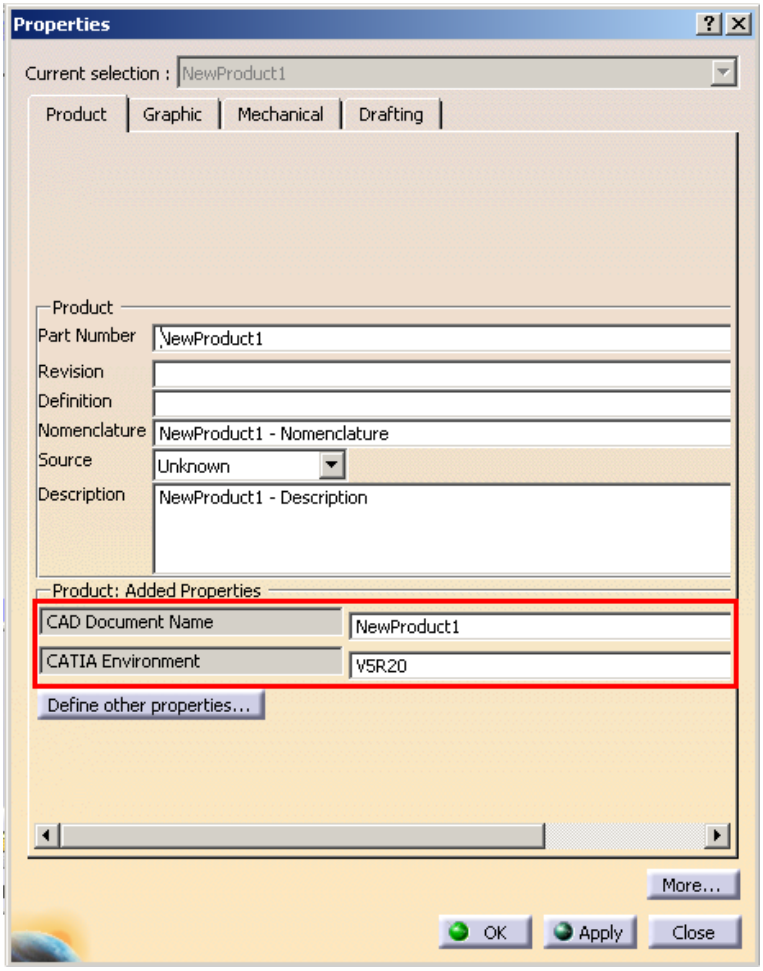


Picture 153: User-defined attributes in the "Properties" dialog of the PDM node



Picture 154: User-defined attributes in Aras Innovator window

After the import or after loading the structure, it can be shown that the values are written from the PDM attributes into the CATIA files (see *Picture 155: User-defined attributes in the "Properties" dialog*).



Picture 155: User-defined attributes in the “Properties” dialog

CATIA User Defined Attributes

Several user defined properties can be filled by CATIA. These properties have to be defined in the Schema file. During the update process in CATIA the property values are filled.

Example:

```
<updateCatiaUserDefinedProperties>

   <userDefinedProperty catiaInfo="CatiaLevel"
                        propertyDisplayName="CATIA Environment" />

   <userDefinedProperty catiaInfo="Mass"
                        propertyDisplayName="The Mass" />

   <userDefinedProperty catiaInfo="Volume"
                        propertyDisplayName="The Volume" />

   <userDefinedProperty catiaInfo="Density"
                        propertyDisplayName="The Density" />

   <userDefinedProperty catiaInfo="Area"
                        propertyDisplayName="The Area" />

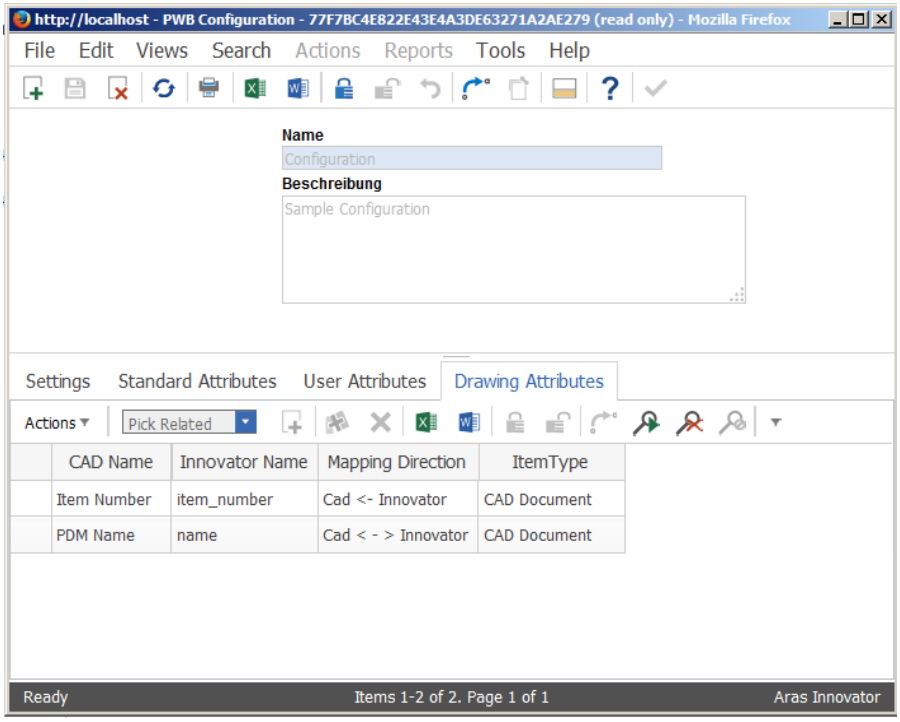
</updateCatiaUserDefinedProperties>
```

Optional.

When the values should to be sent to Aras Innovator they have to be mapped before. For details see the section above (see *Picture 152: Configuration of user-defined attributes in Aras Innovator*).

Drawing Attribute Mapping

The mapping attribute definition for drawings is done in the PWB Configuration.

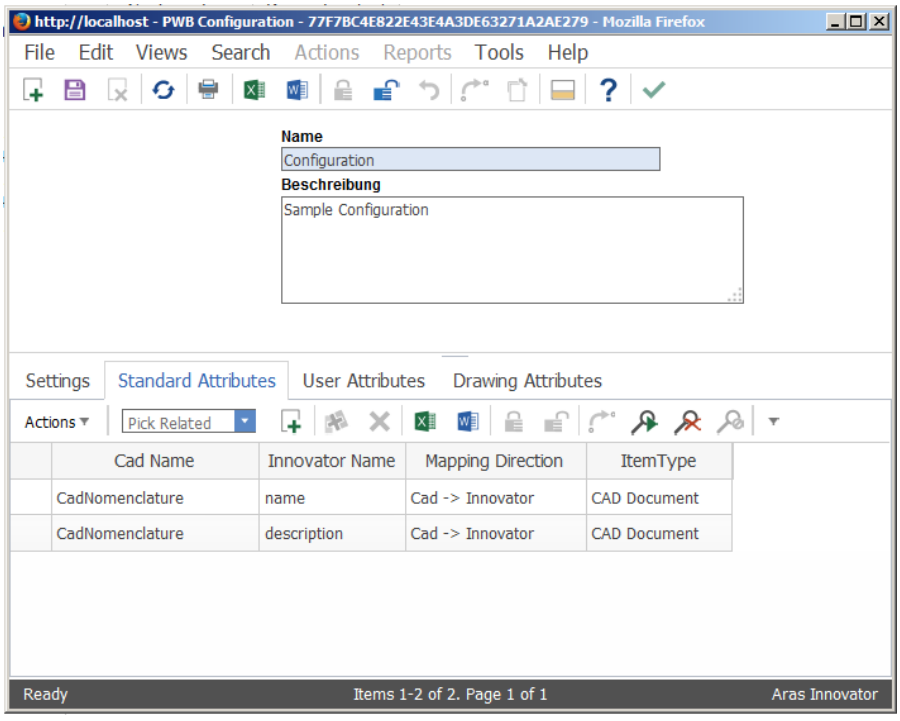


Picture 156: Configuration of drawing attributes in Aras Innovator

Extended Attribute Mapping Functionality

It is possible to define multiple assignments for the same CATIA property in the CAD to PDM direction.

The attribute mapping in the PWB Configuration item can contain multiple PDM attributes for the same CATIA attribute.



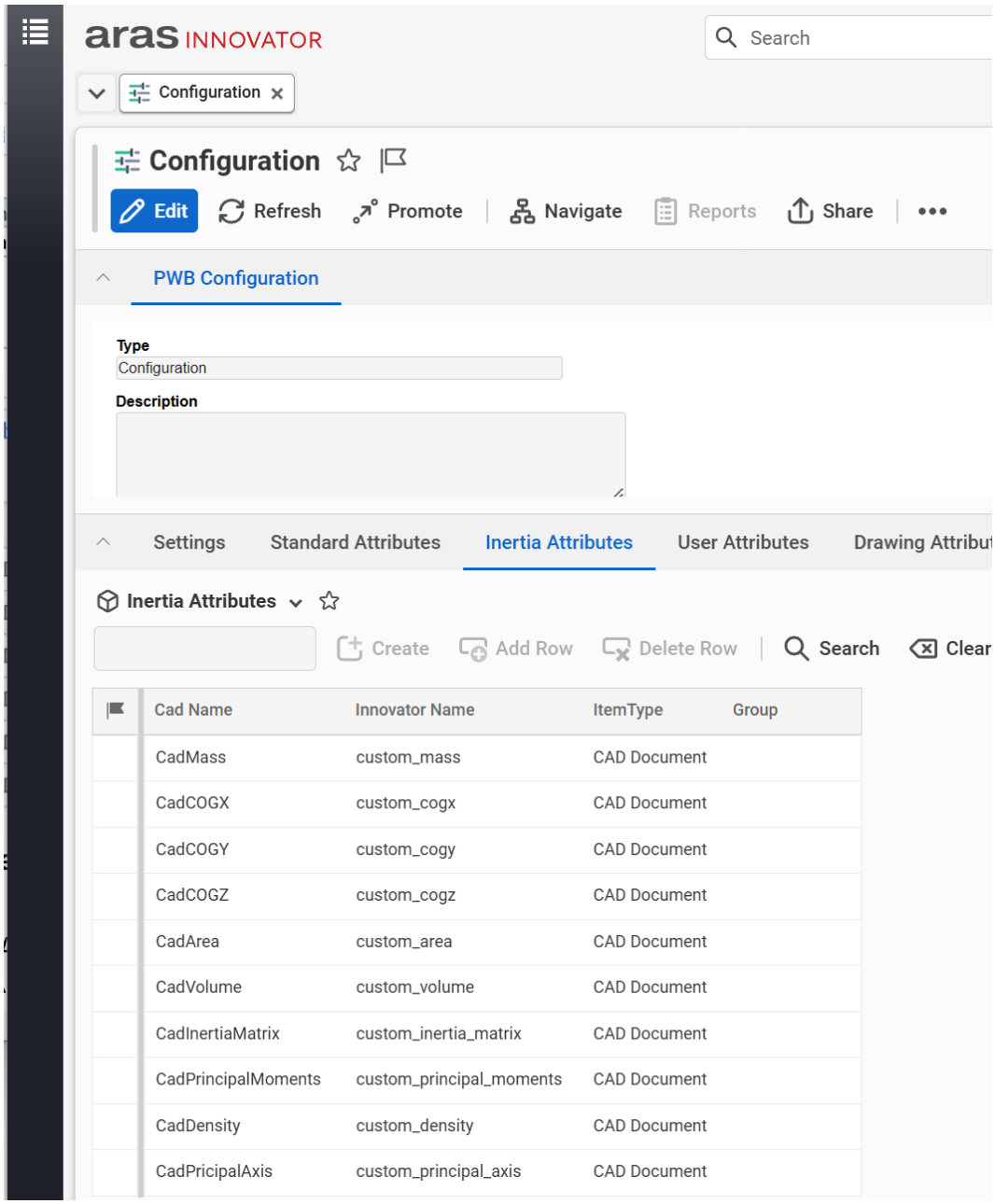
Picture 157: Example PWB Configuration attribute mapping

Inertia Attribute Mapping

You can configure the server to let the CAD client calculate and provide some inertia attribute values (like the center of gravity) when the CAD file is uploaded to the PDM system. These values can be mapped to properties of your CAD or Part items in Aras Innovator.

A CATIA V5 DMU Space Analysis 2 (SPA) license is required to calculate inertia values within CATIA. If you do not have this license available in your CATIA license configuration, you should not configure the inertia attribute mapping in the PWB configuration of Aras Innovator. The PWB Update process will not try to calculate inertia values then.

If you have the license available, you can define which inertia attribute should be mapped to which Aras Innovator property during the PWB Update process. This is similar to the mapping of CATIA standard properties like nomenclature, but always in the direction CAD to PDM only.



Picture 158: Sample inertia attributes mapping

The inertia values can be displayed in CATIA in different units, but they are always calculated, transferred to and stored in Aras Innovator in SI units:

Mass	kg	Kilogram
Wet Area	m^2	Square meter
Volume	m^3	Cubic meter
Center of Gravity	m	Meter
Inertia Matrix	kg m^2	Kilogram square meter (9 space separated values)
Principal Moments	kg m^2	Kilogram square meter (3 space separated values)
Density	kg / m^3	Kilogram per cubic meter
Principal Axis	-	(9 space separated values)

If you want to display the values differently in Aras Innovator, you have to perform your own GUI customization there.

Retrieval and Mapping of Inertia Attributes for non-BOM in BOM CATParts

PWB configuration settings `UseBomPartStructure=true` and `UseNonBom3DCadDocs=true`:

The inertia data are now also read from non-BOM in BOM CATParts.

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.

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” in the Schema file:

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

Add Temp Configuration

Context Action “Add Temp”

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="AddTemp" usedIn="PdmWindow" />
```

AddTemp Prefix

In the Schema file the prefix for the rename of the Part Numbers and File Names for the “Add Temp” and “Open File Temporary” command can be defined.

Example:

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

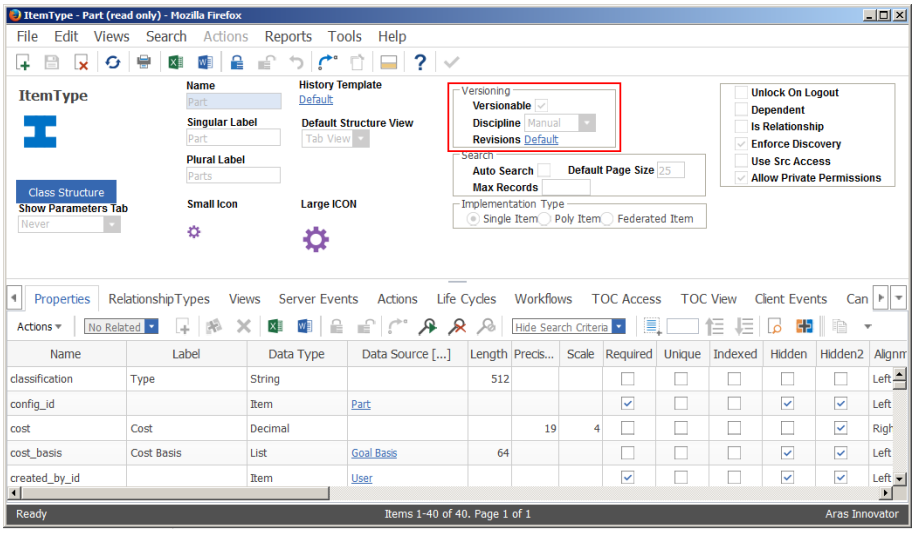
If the add temp prefix value is defined by the environment variable “PWB_ADDTEMP_PREFIX”, then this one takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

Default value: “TMP”

Optional.

Versioning Configuration

Please set the versioning discipline for the Item Types “Part” and “CAD” (and “Document”, if you want to manage Design Tables) to “Manual”. The new generation of a Part and CAD Document will not be created automatically by Aras Innovator in case of an update (see *Picture 159: Item Type “Part”*).



Picture 159: Item Type “Part”

Context Action “Create new generation”

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CreateNewVersion" usedIn="PdmWindow" />
```

Context Action “Delete newest generation”

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="DeleteNewestVersion" usedIn="PdmWindow" />
```

Context Action “Unlink and delete newest generation”

This function is a combination of “delete Part CAD relation” and “delete newest CAD generation”.

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="UnlinkAndDeleteNewestVersion"
               usedIn="PdmWindow" />
```

Only one new Generation of a CAD Document per "Claim" Action

A functionality can be configured where only one new generation of a CAD Document will be created for a claimed document. 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.

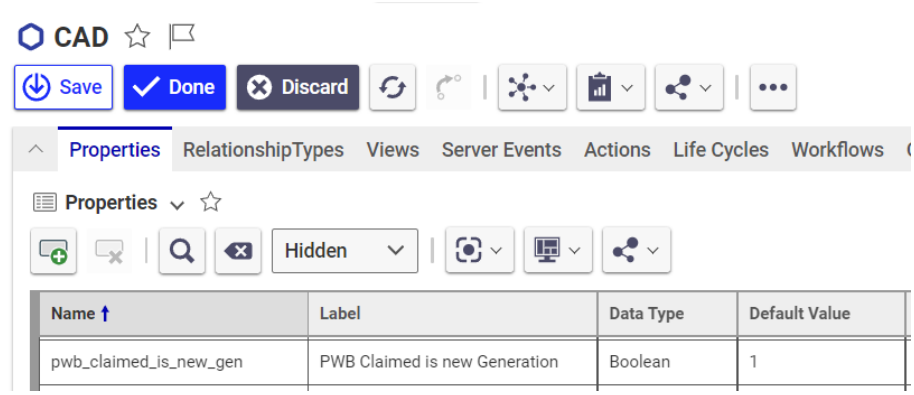
CAD Document generations which are read-only, for example because they are released or frozen can be claimed if they are current. In this case the new generation will be created by the claim process, and the first update will not create another new generation.

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.

Configuration

This functionality is switched on if the CAD Document type has a Boolean attribute which corresponds to the server setting "ClaimedIsNewGenAttr".

For example:

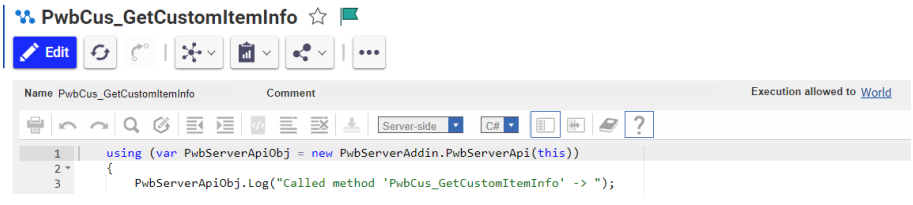


➔

ClaimedIsNewGenAttr	pwb_claimed_is_new_gen

Picture 160: Switching on the "One Generation per Claim" functionality

If the default behavior 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 setting "CustomMethod_GetCustomItemInfo":



➔

CustomMethod_GetCustomItemInfo	PwbCus_GetCustomItemInfo

Picture 161: Sample CustomMethod_GetCustomItemInfo configuration

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());
```

Load Configuration

Context Action "Load"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="LoadStructure" usedIn="PdmWindow|QueryDialog"
               defaultAction="true" />
```

The element `defaultAction="true"` defines this context action as action to be performed by double-clicking on an object of this type.

Context Action "Load (Current)"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="LoadStructureCurrent" usedIn="QueryDialog" />
```

Context Action "Load in Context"

It is possible to only load selected nodes of a structure to CATIA.

In the Schema file the `contextAction` tag with the name "LoadInContext" has to be defined for the needed structure nodes, for example for "/CAD/Mechanical/Part" and "/CAD/Mechanical/Assembly".

Example:

```
<contextAction name="LoadInContext" usedIn="PdmWindow" />
```

Open Configuration

Context Action "Open in new PDM Window"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="OpenInNewPdmWindow"
               usedIn="PdmWindow|QueryDialog" />
```

Context Action "Open File"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Open" usedIn="PdmWindow|QueryDialog"
               defaultAction="true" />
```

The element `defaultAction="true"` defines this context action as action to be performed by double-clicking on an object of this type.

Context Action "Open File with Link"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name=" OpenWithLinks" usedIn="PdmWindow"
               defaultAction="true" />
```

The element `defaultAction="true"` defines this context action as action to be performed by double-clicking on an object of this type.

Context Action "Open File temporarily"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="TempOpen" usedIn="PdmWindow|QueryDialog" />
```

CATDrawing: Loading referenced Data as "Current"

It is possible to load the 3D data (CATParts or CATProduct structures) which are referenced by a CATDrawing as "Current" instead of "As Saved", which is the default.

In the Schema file this functionality is enabled by defining these "contextAction" definitions for the "/CAD/Mechanical/Drawing" object definition.

Example:

```
<contextAction name="OpenDrawingWithRelated3D"
               usedIn="PdmWindow|QueryDialog" />
<contextAction name="OpenDrawingWithRelated3DCurrent"
               usedIn="PdmWindow|QueryDialog" />
```

Context Action "Open related drawings"

This action is available in the CATIA V5 window. It allows to open related CATDrawings for the selected object.

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="OpenRelatedDrawings" />
```

Show Neighborhood Configuration

Bounding Box Definition

In the Schema file the names of the bounding box attributes can be defined.

Example:

```
<boundingBoxAttributes>
   <boundingBoxXMinAttr name="x_min" />
   <boundingBoxXMaxAttr name="x_max" />
   <boundingBoxYMinAttr name="y_min" />
   <boundingBoxYMaxAttr name="y_max" />
   <boundingBoxZMinAttr name="z_min" />
   <boundingBoxZMaxAttr name="z_max" />
</boundingBoxAttributes>
```

When all bounding box attributes are defined, the bounding box functionality is enabled, otherwise the functionality is disabled.

Optional.

Context Action "Show Neighborhood"

In the Schema file the context action can be enabled.

Example:

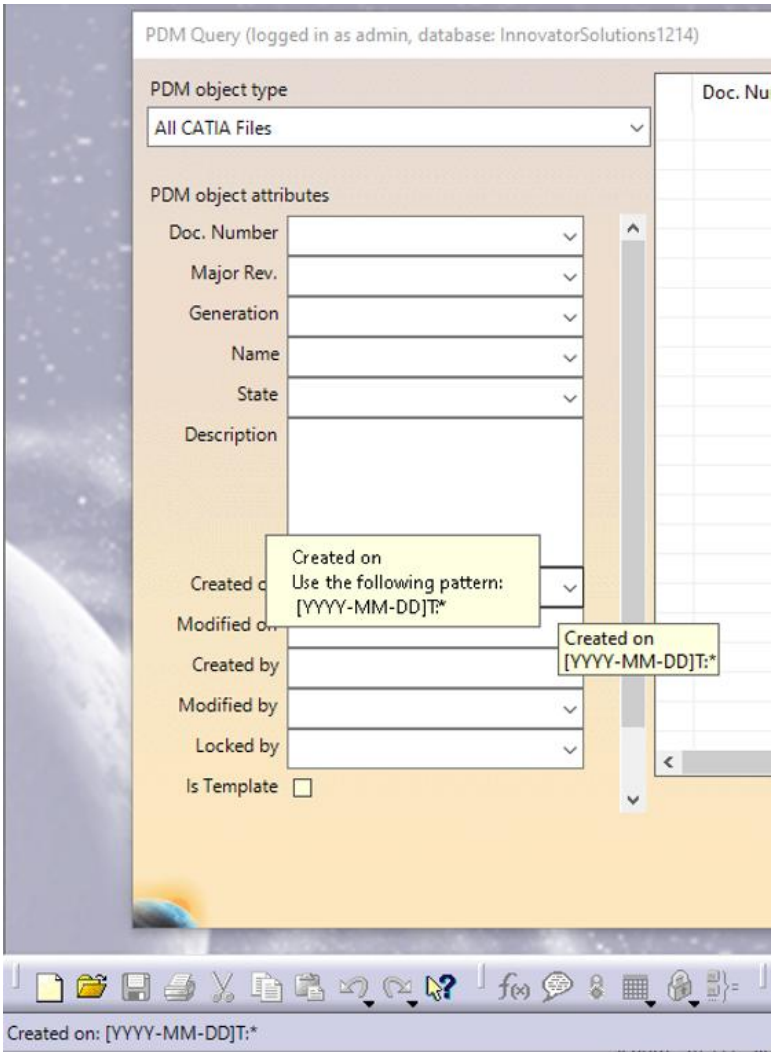
```
<contextAction name="ShowNeighbor" usedIn="PdmWindow|QueryDialog" />
```

Use Tooltip for Attributes in PDM Workbench Dialogs

If you use query, create, update item or any other PDM Workbench functionality where you must fill in some attributes, you often have to follow special patterns for the attributes.

It is possible to configure a help text for every attribute. If needed, you can use different help texts for each kind of dialog.

The help can be configured as tooltip, as text in the status bar and as help which is displayed when using “What’s This” or “?” of the dialog:



Picture 162: Display help for PWB attributes

Configuration

If you want to use the same help texts for an attribute in all kinds of dialogs, you can just add the help texts to the file PWBSchemaDisplayNames_Aras_Aras.CATNls (To create a multi-line help it is possible to use “\n” or a linebreak):

```
NLS_created_on = "Created on";
NLS_created_on.Help = "Created on: [YYYY-MM-DD]T:*";
NLS_created_on.ShortHelp = "Created on\n[YYYY-MM-DD]T:*";
NLS_created_on.LongHelp = "Created on
Use the following pattern:
[YYYY-MM-DD]T:*";
```

<NLS_attribute>.Help … This text is displayed in the status bar.

<NLS_attribute>.ShortHelp … This text is displayed as tooltip.

<NLS_attribute>.LongHelp … This text is displayed when using “What’s This”.

To use a different help text for an attribute in the context of a special dialog you need to define a special displayName for the attribute in the context of the dialog in the Schema file:

```
<object name="[Name]" displayName=”[NLS_name]" icon="[icon]">
   ...
   <form name="[Form name]">
      ...
      <formAttribute name="attribute name"
         displayName="NLS_name for attribute in form" ... />
      ...
   </form>
   ...
</object>
```

Then add the new displayName with its Help, ShortHelp, and LongHelp to the file “PWBSchemaDisplayNames_Aras_Aras.CATNls”.

Example:

In the Schema file:

```
<object name="/CAD/Mechanical/Assembly"
        displayName="NLS_CATProduct" icon="CATProduct">
   ...
   <form name="Register">
      <formAttribute name="item_number"
                     displayName="NLS_item_number_CAD_Register"
                     widgetType="SingleLineEditor" mode="update"
                     visibleLength="15" required="true" />
      ...
   </form>
</object>
```

In the file “PWBSchemaDisplayNames_Aras_Aras.CATNls”:

```
NLS_item_number_CAD_Register = "Doc. Number";
NLS_item_number_CAD_Register.Help = "Doc. Number: XXX.YYY-ZZZ";
NLS_item_number_CAD_Register.ShortHelp = "Doc. Number:
Pattern: XXX.YYY-ZZZ";
NLS_item_number_CAD_Register.LongHelp = "Doc. Number:
Pattern: XXX.YYY-ZZZ
Where
XXX ... Main Project
YYY ... SUB Project
ZZZ ... individual number";
```

Link Support

Link Management

When designing, the user can create many types of links between CATIA features or between a CATIA feature and a foreign object. Some of these links are presented to the user in the “Edit Links” command, others are not shown in the CATIA UI.

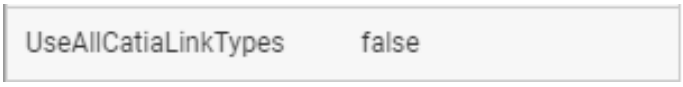
The information about the CATIA links, which are pointing from an object in one document to an object in another document (“external link”), can be retrieved and transferred to Aras Innovator.

Previous PDM Workbench releases supported the transfer of CATIA links between a CATDrawing and a CATProduct or CATPart (“Drawing” link in Aras Innovator) and between a CATPart and a CATPart (“Reference” link in Aras Innovator). Furthermore, regular product structure links and design table links were supported. This is still the default behaviour of the PDM Workbench.

The PDM Workbench provides the capability to detect and transfer all CATIA external link types as they are provided and grouped by CATIA. This has to be configured.

Configuration

Create the server PWB Configuration item setting to use basic link management (default).



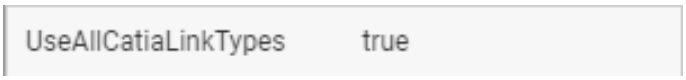
Picture 163: Sample UseAllCatiaLinkTypes configuration - “false”

The default value is “false”.

Only the following subclasses of the Aras Innovator ItemType “CAD Structure” are used:

- Reference
- Drawing

Create the server PWB Configuration item setting to use all CATLinkTypes:



Picture 164: Sample UseAllCatiaLinkTypes configuration - “true”

The following subclasses of the Aras Innovator ItemType “CAD Structure” are created:

- Contextual
- Reference (Original CATIA link type “Design”)
- Drawing (Original CATIA link type “Downstream”)
- IsComposedOf
- Product
- Reference
- Result
- RuleBase

For compatibility reasons links between CATDrawings and their original 3D geometry files are still created as “Drawing”, and non-context links between two CATPart files are by default still created as “Reference”.

With the following two server settings (value “true”) the original internal CATIA link names of “Downstream” for drawing links and “Design” for CATPart reference links will be created instead:

UseOrigCatiaLinkTypeName-Drawing	true
UseOrigCatiaLinkTypeName-Reference	true

Picture 165: Sample UseOrigCatiaLinkTypeName-Drawing and UseOrigCatiaLinkTypeName-Reference configuration

The default values are “false”.

Possibility to filter Relation Definitions in Schema File depending on Configuration Settings

It is possible to filter *relation* definitions in the Schema file depending on client environment variables, settings in the Schema file (<setting> … </setting>), or PWB Configuration settings on the server.

This is used for not showing certain classifications of CATIA link relations depending on server settings.

Example:

```
<!-- possible values for 'onlyIfDefined':
        "client-envvar", "client-setting", "server-setting" -->

<relation name="/CAD Structure/Reference"
          displayName="NLS_CADStructure_Reference"
          icon="Aras_Relation" createAllowed="true"
          expandAllowed="true" info="AsSaved"
          onlyIfDefined="server-setting:UseOrigCatiaLinkTypeName-Reference=false" >

   ...

</relation>
```

Basic Drawing Link Support

The drawing link functionality has to be switched on by defining the Schema file setting `<updateCatiaLinksInPdm value="true" />`.

Optional.

In addition to that, an additional form attribute has to be defined in the “Register” form of the CATDrawing object definition. This attribute has to have the attribute `displayOnly="true"` to indicate that it is not an actual PDM attribute, only a display attribute in the dialog:

```
<formAttribute name="LinkedMainGeometryFile" widgetType="ComboBox"
               mode="update" visibleLength="15" required="false"
               listViewRelevant="false" displayOnly="true" />
```

The corresponding attribute definition has to contain the XML attribute `pwbAttrInfo="MainDrwGeometryFile"`, and it has to refer to a data source which contains some special definitions:

```
<attribute name="LinkedMainGeometryFile"
           displayName="NLS_LinkedMainGeometryFile"
           dataSource="LinkedGeometryFiles"
           pwbAttrInfo="MainDrwGeometryFile" />
```

The data source definition which is referred has to contain the definition `additionalValues="GeometryFilesLinkedInSession"`:

```
<dataSource name="LinkedGeometryFiles" type="ValueList"
            additionalValues="GeometryFilesLinkedInSession" />
```

If the configuration is set up correctly then the user will see an additional list widget in the “Register” dialog for CATDrawings which enables him to select one of the linked 3D geometry files (see *PDM Workbench User Manual*).

Download related 3D File

In the Schema file it can be configured if with the loading of the drawing file the reference 3D file should also be downloaded and opened in CATIA.

Example:

```
<downloadDrawingRelated3DFile value="true" />
```

Default value: “false”

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

Option to block the Update if linked File is not saved

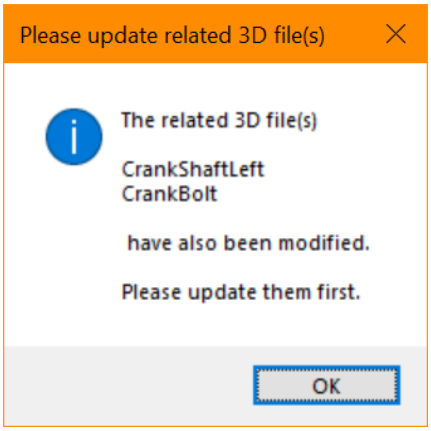
Configuration

In the Schema file the setting “UpdDrw-CheckLinked3DFiles” has to be set to the value “true”. The same applies to the setting “UpdPrt-CheckLinked3DFiles”.

```
<settings>
   <setting name="UpdDrw-CheckLinked3DFiles" value="true"/>
   <setting name="UpdPrt-CheckLinked3DFiles" value="true"/>
</settings>
```

Usage

When the user tries to update a CATDrawing which has links to CATParts that are loaded in the CATIA session, or a CATPart which has reference links to other CATParts, then the information to update those CATParts first is presented:



Picture 166: Information about CATParts to be updated

Duplicate related Drawings

When a CATPart or a CATProduct is duplicated with the PDM Workbench, the related CATDrawing will also be duplicated if the Part Number of the CATPart/CATProduct is the prefix of the CATDrawing, e.g. “Part1Drw” is related to “Part1” and has the correct prefix.

In the Schema file the length of the prefix to be compared can be defined.

Example:

```
<duplicateRelatedDrawings charsToCompare="25" />
```

Default value: “0”

Optional.

Basic Multi-Model Link Support

In the Schema file the multi-model link functionality can be switched on, similar to the drawing link functionality.

Example:

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

Default value: “false”

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

Representation Types

It is possible to define an attribute on the Item Type “CAD Document” which defines the so-called “rep type”, that is, the representation type. Only CAD Documents with the default rep types will be loaded to the CATIA session.

This functionality only applies to the BOM Part Structure Data Model.

Part CAD Filter

The PWB Configuration item settings “RepTypeAttribute”, “Default3DRepType”, and “Default2DRepType” have to be set.

The PWB Configuration item setting “RepTypeAttribute” has to be set to the internal attribute name of a list attribute which is defined on the type “CAD”. The list should contain all the values that are possible for the rep type attribute.

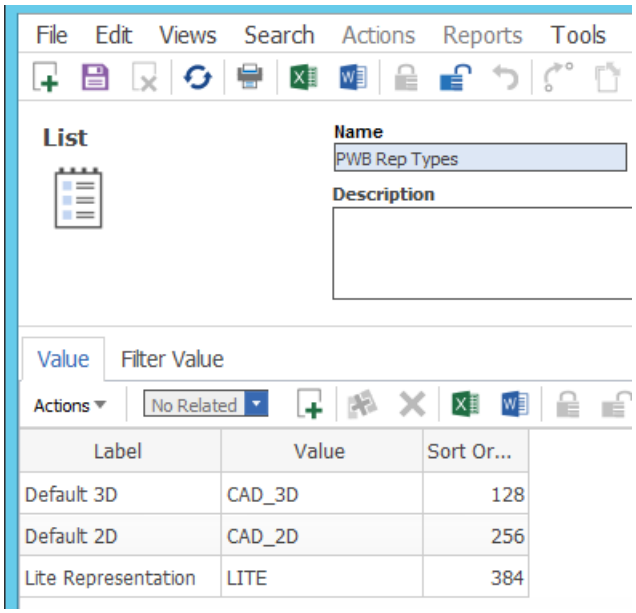
pwb_rep_type	pwb_rep_type	List	PWB Rep Types

Picture 167: Example rep type attribute

RepTypeAttribute	pwb_rep_type
Default2DRepType	CAD_2D
Default3DRepType	CAD_3D

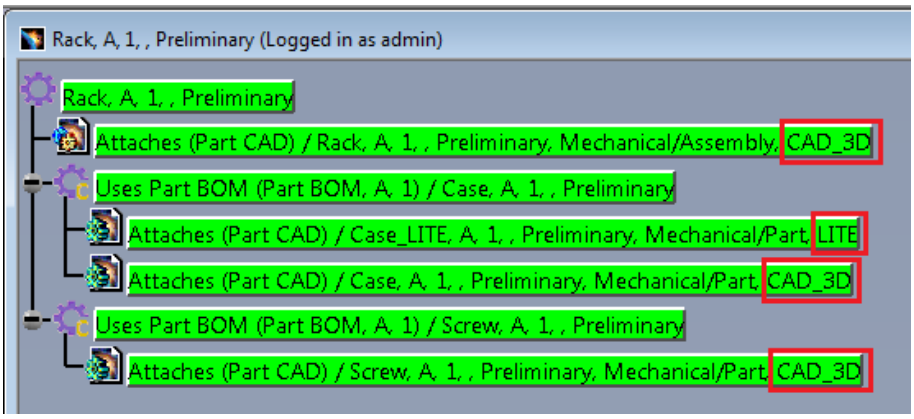
Picture 168: Sample rep type value list configuration

The PWB Configuration item settings “Default3DRepType” and “Default2DRepType” have to be set to the default type for 3D geometry files (CATParts and CATProducts) and 2D geometry files (CATDrawings), respectively.



Picture 169: Sample Part CAD filter configuration

If this functionality is switched on, then newly created CAD Documents will automatically have the rep type attribute value set to their default value. The value can be changed to a different value (e.g. “LITE”). Only CAD Documents with the default rep type value will be loaded.

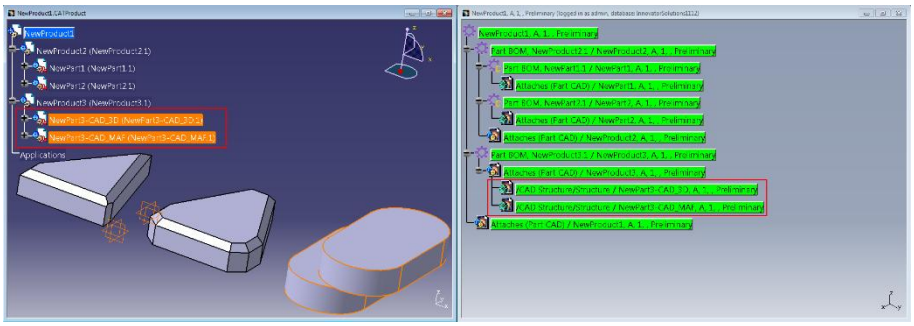


Picture 170: Different rep type values on CAD Documents in a structure

Additional Rep Types

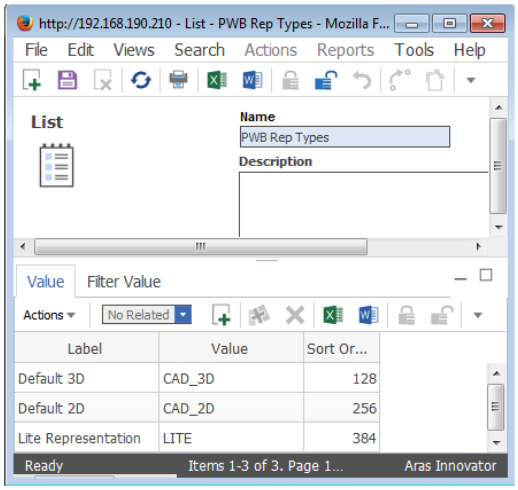
It is possible to load additional 3D rep types in addition to e.g. “CAD_3D” in the CATProduct structure at the same time.

This is only possible when the different 3D rep type CATParts are defined as non-BOM CAD Documents. BOM-relevant CAD Documents are the CAD Document items that are related directly to the Part item with the “Part CAD” relation. Non-BOM CAD Documents are the “CAD Structure/Structure” child nodes of the BOM-relevant CAD Documents:

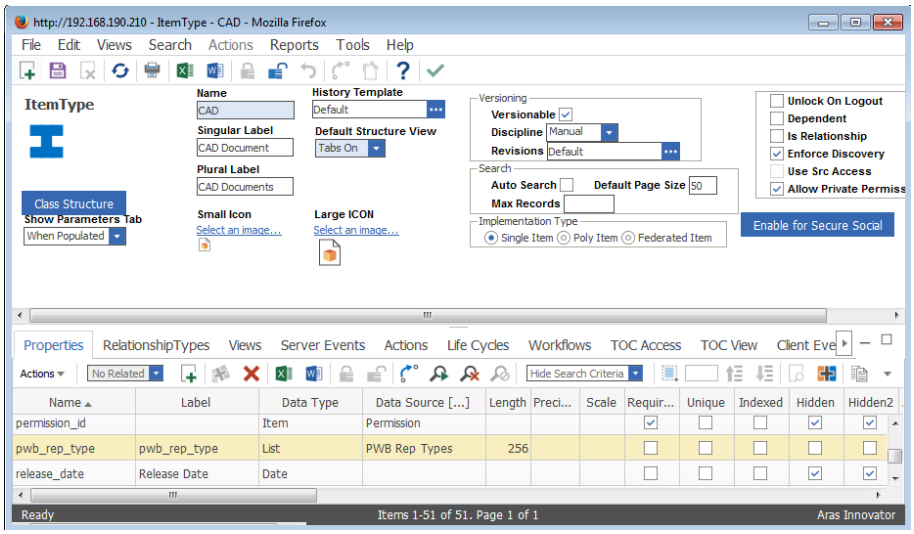


Picture 171: Two CATParts with different rep types related to the same part loaded at the same time

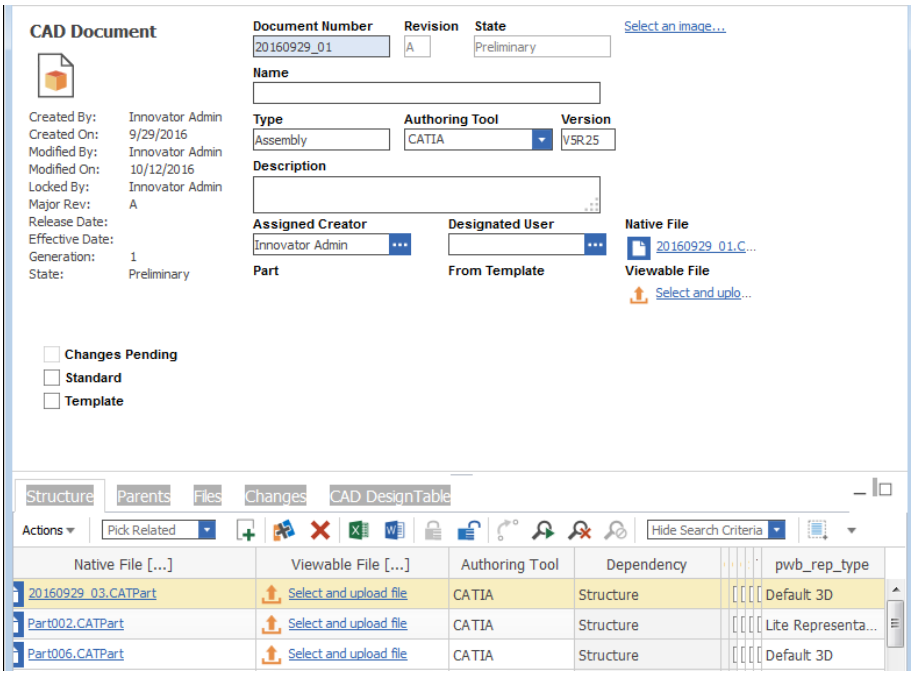
To use the Rep Type Filter functionality, you have to create a list of rep types in Aras Innovator and assign the list to the “pwb_rep_type” attribute at CAD. Then you can change the “pwb_rep_type” of a CAD Document.



Picture 172: Create / Update list of Rep Types



Picture 173: Assign Rep Type List to CAD



Picture 174: Change pwb_rep_type

On the client you have to add the list of filter3dRepTypes to the Schema file. This is a subset of the “pwb_rep_type” list that only holds 3D rep types.

Example:

```
<filter3dRepTypes>
   <repType3d name="CAD_3D" />
   <repType3d name="CAD_MAF" />
   <repType3d name="LITE" />
   <repType3d name="CAD_KIN" />
   <repType3d name="CAD_WFR" />
</filter3dRepTypes>
```

Optional.

Attach additional Non-Bom CATParts to Part

When using the BOM Part Structure Data Model by default only one CATPart can be attached to an Aras Innovator Part (Component). It can be configured to allow additional Non-BOM CATParts attached to an Aras Innovator Part (Component).

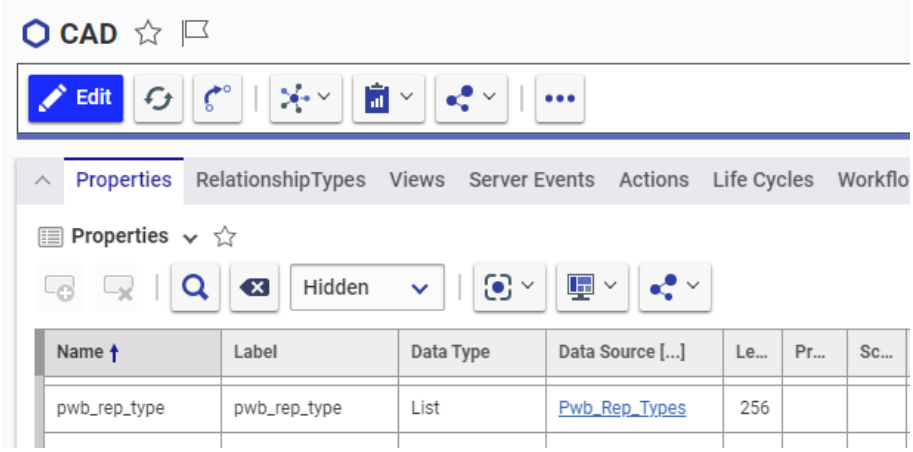
When loading the Component, the additional CATParts are only loaded into the CATIA session if they are referenced by a CATIA file which is part of the normal BOM structure. It is also possible to load a Non-BOM CATPart directly.

To create a Non-BOM CATPart it is possible to set the PLM type of a CATPart. There is also a functionality to relate the active Non-BOM CATPart to an Aras Innovator Part.

To use the functionality the usage of Non-BOM CATIA files in BOM Part Structure Data Model must be enabled.

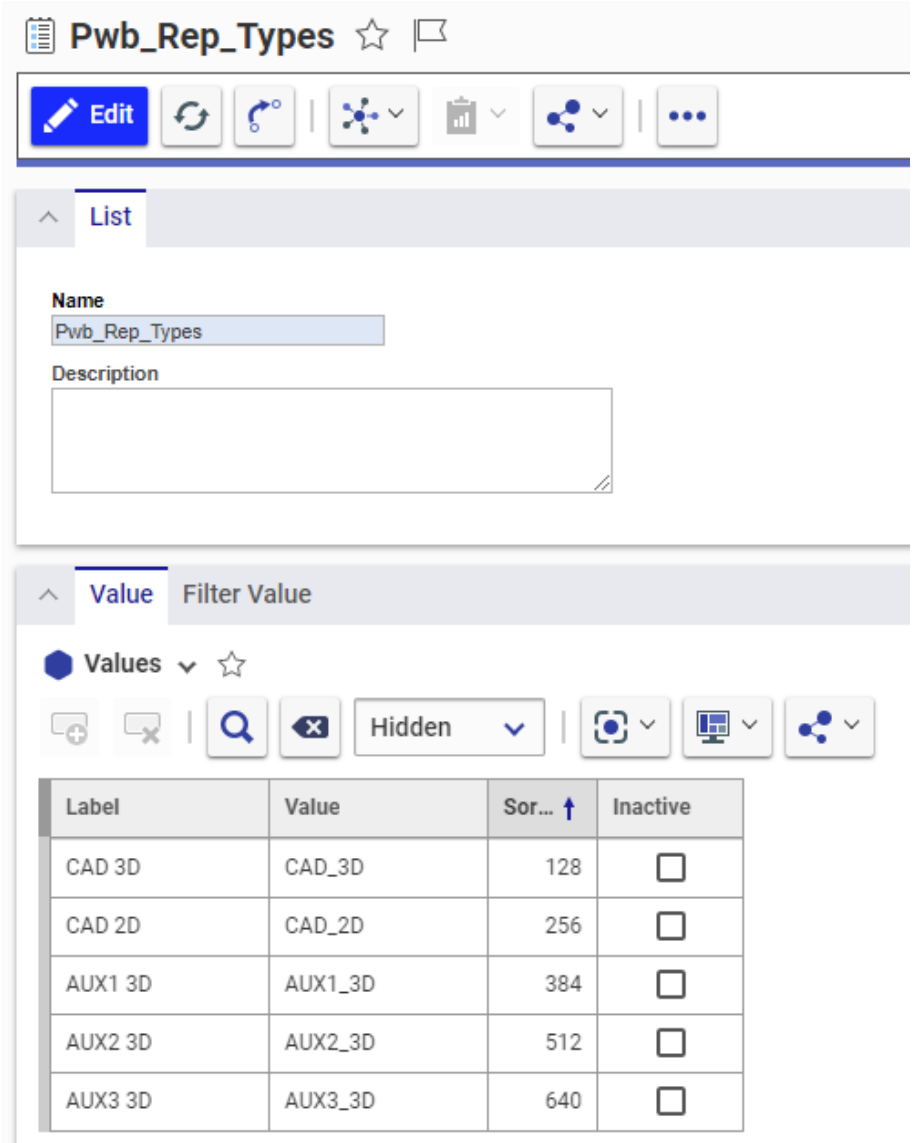
On the server side the following configuration is required:

The functionality uses an additional property at the CAD Document item to hold the representation type.



Picture 175: Configuration on Item Type “CAD”

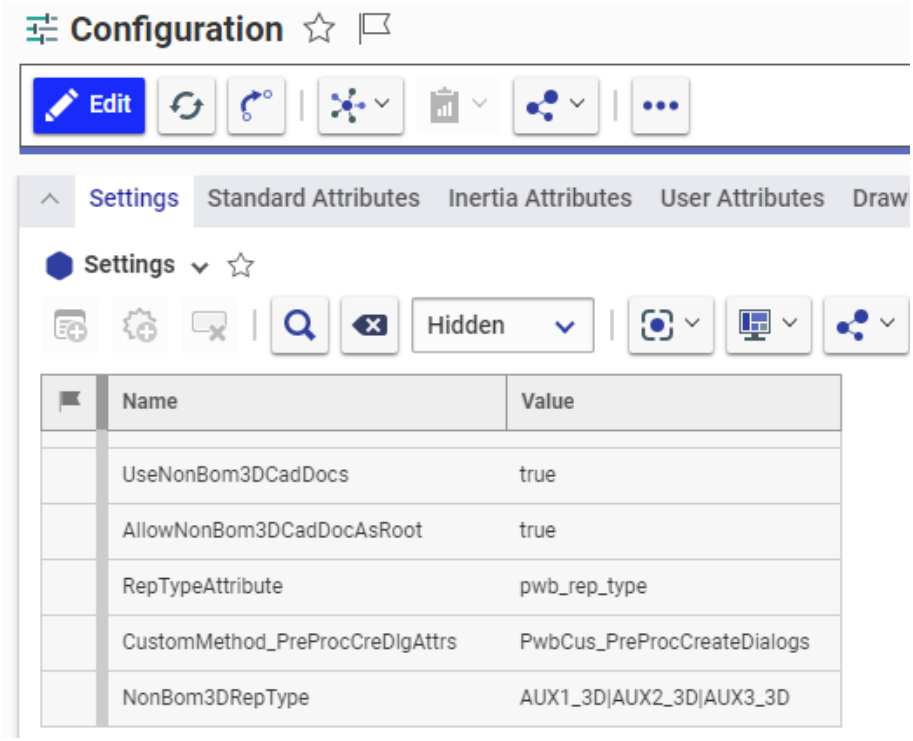
With a list of valid representation types:



Picture 176: List of valid representation types

For the normal CATParts / CATDrawings the PDM Workbench uses the representation types CAD_3D and CAD_2D (default).

The property name which holds the representation type and some other settings must be configured in the PWB Configuration item:



Picture 177: Configurations for Representation Types in the PWB Configuration

- UseNonBom3DCadDocs=true
  General allow the usage of Non-BOM CATIA files when using BOM Part Structure Data Model.
- AllowNonBom3DCadDocAsRoot=true
  Allow the usage of a Non-BOM CATIA file in its own window.
- RepTypeAttribute=pwb_rep_type
  Property at the Item Type “CAD” that holds the representation type.
- NonBom3DRepType=AUX1_3D|AUX2_3D|AUX3_3D
  List of Representation Types (separated by “|”) that indicate a Non-BOM CATPart.
- (Default3DRepType=CAD_3D)
  Representation Type that indicates a normal BOM CATIA file (if set, the value must not be empty).
- (Default2DRepType=CAD_2D)
  Representation Type that indicates a CATDrawing.
- CustomMethod_PreProcCreDlgAttrs=PwbCus_PreProcCreateDialogs
  Automatically set the Representation Type without selecting it in the “PDM Create” dialog of a CATPart.

Sample custom method to set the representation type to AUX_3D for a non BOM CATPart:

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
   string PdmType = this.getProperty("Type");
   string PdmClassification = this.getProperty("Classification");

   PwbServerApiObj.Log("Called method -> '" + PdmType +
                  "' / '" + PdmClassification + "'" );

   IDictionary<string, string> PartInputDialogDict = null;
   if (PdmType == "Part")
```

```
{
   // Part dialog attributes
   Item PartInputDialogItem = getPropertyItem("PartDialogAttrs");
   if (PartInputDialogItem == null)
   {
      return getInnovator().newError(
         "Could not retrieve part dialog attributes !");
   }
   else
   {
      PartInputDialogDict =
         PwbServerApiObj.DialogAttrsItemToDictionary(
            PartInputDialogItem);
      if (PartInputDialogDict == null)
      {
         return getInnovator().newError(
            "Could not convert part dialog attributes !");
      }
   }
}

IDictionary<string, string> CadDocInputDialogDict = null;
if (PdmType == "CAD")
{
   // CAD dialog attributes
   // The CAD document dialog is only needed when a CAD type
   // needs to be created
   Item CadDocInputDialogItem =
      this.getPropertyItem("CadDocDialogAttrs");
   if (CadDocInputDialogItem == null)
   {
      return getInnovator().newError(
         "Could not retrieve CAD document dialog attributes !");
   }

   CadDocInputDialogDict =
      PwbServerApiObj.DialogAttrsItemToDictionary(
         CadDocInputDialogItem);
   if (CadDocInputDialogDict == null)
   {
      return getInnovator().newError(
         "Could not convert CAD document dialog attributes !");
   }
}

// CATIA standard properties
// 'CadPartNumber', 'CadRevision', 'CadNomenclature',
// 'CadDefinition', 'CadDescriptionReference', 'CadFileName',
// 'CadIsBomNode'
IDictionary<string, string> CatiaStdPropsDict = null;
Item CatiaStdPropsItem = this.getPropertyItem("CadStdProps");
if (CatiaStdPropsItem == null)
{
   // return getInnovator().newError(
   //   "Could not retrieve CATIA standard properties !");
}
else
{
   CatiaStdPropsDict =
      PwbServerApiObj.DialogAttrsItemToDictionary(
         CatiaStdPropsItem);
   if (CatiaStdPropsDict == null)
   {
      // return getInnovator().newError(
      //   "Could not convert CATIA standard properties !");
   }
}

// Resulting output dialog attributes
IDictionary<string, string> OutputDialogDict = new Dictionary<string, string>();

IDictionary<string, string> InputDialogDict = null;
if ((PdmType == "CAD") || (PdmType == "Part"))
{
   if (PdmType == "CAD")
   {
      InputDialogDict = CadDocInputDialogDict;

      if (CatiaStdPropsDict != null)
      {
         string CadFileName = null;
```

```
            if (CatiaStdPropsDict.TryGetValue(
                  "CadFileName", out CadFileName) &&
               !String.IsNullOrEmpty(CadFileName) &&
               CadFileName.EndsWith(".CATPart"))
            {
               string IsBomNode = null;
               if (CatiaStdPropsDict.TryGetValue(
                     "CadIsBomNode", out IsBomNode) &&
                  !String.IsNullOrEmpty(IsBomNode))
               {
                  if (IsBomNode == "false")
                  {
                     List<string> NonBomRepTypes = new List<string>();
                     NonBomRepTypes.Add("AUX1_3D");
                     NonBomRepTypes.Add("AUX2_3D");
                     NonBomRepTypes.Add("AUX3_3D");

                     string currentRepType = null;
                     if (InputDialogDict.TryGetValue(
                           "pwb_rep_type", out currentRepType) &&
                        !String.IsNullOrEmpty(currentRepType))
                     {
                        if (!NonBomRepTypes.Contains(currentRepType))
                        {
                           InputDialogDict.Remove("pwb_rep_type");
                           InputDialogDict.Add("pwb_rep_type", NonBomRepTypes[0]);
                        }
                     }
                     else
                     {
                        InputDialogDict.Remove("pwb_rep_type");
                        InputDialogDict.Add("pwb_rep_type",
                                            NonBomRepTypes[0]);
                     }
                  }
                  else
                  {
                     InputDialogDict.Remove("pwb_rep_type");
                     InputDialogDict.Add("pwb_rep_type", "CAD_3D");
                  }
               }
               else
               {
                  InputDialogDict.Remove("pwb_rep_type");
                  InputDialogDict.Add("pwb_rep_type", "CAD_3D");
               }
            }
         }
      }
      else
      {
         InputDialogDict = PartInputDialogDict;
      }
   }

   Item OutputDialogItem =
      PwbServerApiObj.DialogAttrsDictionaryToItem(
         InputDialogDict);

   return OutputDialogItem;
}
```

On the Client the following settings are required in the Schema.xml file:

```
<settings>

   ...

   <!-- enable the possibility to register a new CATPart as -->

   <!-- non BOM and relate it to the selected Aras Part -->

   <setting name="RelateNonBomToPartEnableRegisterNew"
            value="true"/>

</settings>
```

```
<object name="/Part/Component" displayName="NLS_Component"
        icon="Aras_Component" >

   ...

   <contextAction name="RelateNonBomToPart"
                  usedIn="QueryDialog"/>

   ...

</object>


<object name="/CAD/Mechanical/Part" displayName="NLS_CATPart"
        icon="CATPart" isDefaultFor="CATPart">

   ...

   <contextAction name="SetPdmType" />


   <form name="Query">

   ...

   <formAttribute name="pwb_rep_type" widgetType="ComboBox"
                  mode="output" visibleLength="15"
                  required="false"
                  entryAllowed="true" dataSource="RepType" />

   </form>

   ...

</object>

...

<attribute name="pwb_rep_type" displayName="NLS_pwb_rep_type"
           dataSource="RepType" />

...

<dataSource name="RepType" type="ValueList">

   <value name="AUX1_3D" displayName="AUX1_3D" />

   <value name="AUX2_3D" displayName="AUX2_3D" />

   <value name="AUX3_3D" displayName="AUX3_3D" />

   <value name="CAD_3D" displayName="CAD_3D" />

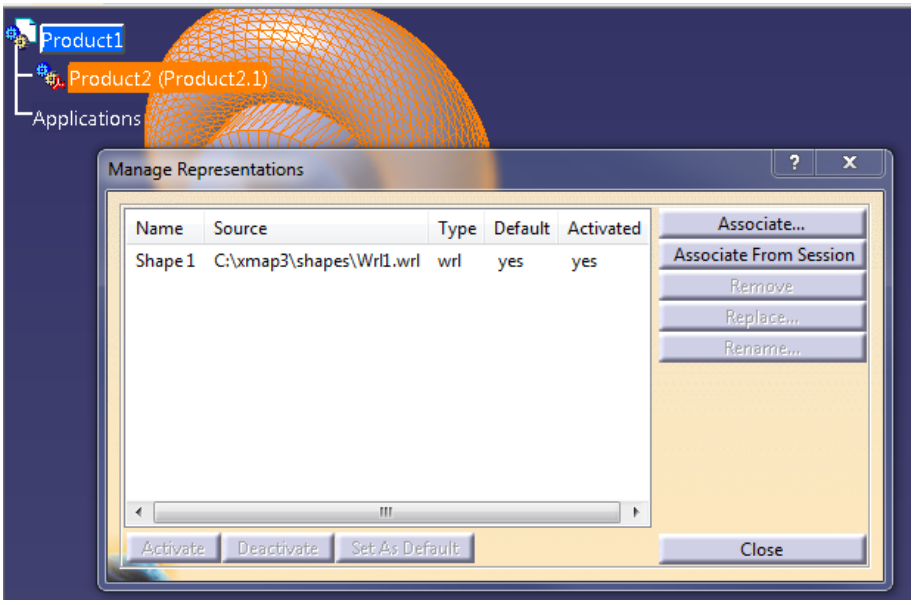
</dataSource>
```

If you are using a custom method to get the filtered models of a Part (CustomMethod_GetFilteredModelsForParts) you have to make sure this method does not return CADs using the property `pwb_rep_type="AUX1_3D|AUX2_3D|AUX3_3D".`

Support generic Shape Representations

Depending on the CATIA installation it is possible to add a geometry file of a type like CATShape, jt, pkg, wrl, STL or others as a representation to a Component. The present functionality supports to store these additional geometry files in Aras Innovator. The extra geometry files are treated as read-only files, just like CATIA V4 model files.

It is possible to configure up to five additional types to be supported.

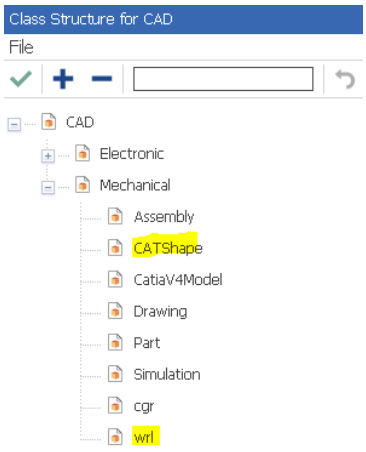


Picture 178: Manage Representations

Configuration

Server

To use additional file types like CATShape or wrl, new classifications for the Item Type “CAD” need to be defined.



Picture 179: Class Structure “CAD” – Added “CATShape” and “wrl”

The PWB Configuration item settings “GenericShapeTypeX” have to be set to the type, where X is a number from 1 to 5.

The PWB Configuration item settings “GenericShapeExtensionX” have to be set to the corresponding file extension, where X is a number from 1 to 5.

To generate thumbnails in CATIA for the new types, you have to extend the “CreateThumbnailsFromTypes” in the PWB Configuration.

GenericShapeType1	/CAD/Mechanical/CATShape
GenericShapeExtension1	CATShape
CreateThumbnailsFromTypes	.CATPart|.CATDrawing|.CATShape|.model|.cgr|.wrl
GenericShapeType2	/CAD/Mechanical/wrl
GenericShapeExtension2	wrl

Picture 180: PWB Configuration – “CreateThumbnailsFromTypes”

Client

The new types have to be configured in the Schema file on the client. The easiest thing is to orientate yourself by the definition of the CATIA V4 model.

Checks

Check File Versions in Structure

In the Schema file it can be configured that the file versions should be checked in the structure.

Example:

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

Default value: “true”

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

Check CATParts in Visualization Mode

In the Schema file it can be configured that it will be checked for CATParts loaded in Visualisation mode.

Example:

```
<checkCatPartsInVisuMode value="false" />
```

Default value: “true”

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

Check CATIA Part Numbers before Update

In the Schema file it can be configured that the CATIA Part Numbers are checked for invalid characters before update.

Example:

```
<checkCatiaPartNumbersBeforeUpdate value="false" />
```

Default value: “true”

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

Check for CAD Owner

In the Schema file it can be configured if it is possible to load structures which have been created by a different CAD integration (“external_owner” attribute value is not “T-Systems.Mechanical.CATIA”).

If this setting is set to “true” then it is not possible to load structures from a different integration.

Example:

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

Default value: “false”

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

Check for CAD Document CATIA Release at PDM Update

In the Schema file the functionality can be switched on that asks the user before overwriting a file which has been created with a lower release of CATIA V5.

Example:

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

Default value: “false”

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

Optional Check for broken Links at Update

Normally creating a CATProduct with broken links in PDM gives warnings. It is possible to configure the integration to give errors instead, thereby stopping the update process.

In the Schema file the following has to be added in order to stop Synchronize if there are broken links at new CATProducts.

Example:

```
<allowBrokenLinkAtNewProduct value="false" />
```

Default value: “true”

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

Delete Relations of non-loaded Instances

Normally you can only delete relations during PWB Update if the relation was loaded by the PDM Workbench. With this functionality it is possible to delete relations which are not loaded.

In the PWB Configuration item you have to add the setting “DeleteNotLoadedInstances”:

DeleteNotLoadedInstances	true

Picture 181: Sample DeleteNotLoadedInstances configuration

Default value: “false”

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

Check CAD Links

When CATIA documents with 3D links need to be imported this functionality helps the user to determine which documents have to be imported in which order, and which documents have to be in the CATIA session so the links are created correctly.

To retrieve the CAD Documents which already exist in PDM by default the CAD Document number is compared to the CATIA file name. If the CAD Documents in PDM have been assigned a PDM-generated number then the original file name has been overwritten. If the original file name has been saved in another PDM attribute this other attribute can be queried instead.

If, for instance, the original file name has been saved in the “name” attribute of the CAD Document, then this functionality can be configured to query for the “name” attribute when comparing the value with the CATIA file name.

For this the PWB Configuration item setting “CadLinkCheckQueryAttribute” can be set to a different PDM attribute, e.g. “name”.

CadLinkCheckQueryAttribute	name

Picture 182: Sample CadLinkCheckQueryAttribute configuration

Context Action "Check CAD Links"

This action is available in the CATIA V5 window. It allows to check the CAD links for the selected object.

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CheckCadLinks" />
```

Miscellaneous Context Actions

Context Action "Properties"

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

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

Context Action "Unclaim"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog" />
```

Context Action "Claim"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CheckOut" usedIn="PdmWindow|QueryDialog" />
```

Context Action "Unclaim all"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CheckInAll" usedIn="PdmWindow" />
```

Context Action "Claim all"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CheckOutAll" usedIn="PdmWindow" />
```

Disable Confirmation Messages for Claim/Unclaim Actions

The Confirmation messages can be switched on/off by the schema file setting

`<setting name="DisableConfirmPromptOnClaimUnclaim" value="true"/>`.

Then the confirmation message will not be shown if the Claim/Unclaim action was successfully performed.

Optional Availability of Claim and Unclaim Toolbar Icons in the Toolbar

The icons can be switched on by the schema file setting

```
<setting name="EnableClaimUnclaimMenu" value="true"/>
```

Context Action "Duplicate"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Duplicate" usedIn="PdmWindow|QueryDialog" />
```

Context Action "Duplicate-Delete"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="DuplicateDelete" usedIn="PdmWindow|QueryDialog" />
```

Context Action "Duplicate-Supersede"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="DuplicateSupersede" usedIn="PdmWindow" />
```

Context Action "Insert PDM Node"

This action is available in the CATIA V5 window. It allows to insert an item directly into the CATIA structure of the selected object.

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Insert" />
```

Insert from Aras Innovator keep "PDM Query" Dialog

The "PDM Query" dialog stays open. The user can insert multiple items from the same "PDM Query" dialog.

Configuration

To get the behavior that the "PDM Query" dialog is closed after insert of the item, you have to add the setting "InsertFromArasCloseAfterInsert" to the settings section of the Schema file:

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

Context Action "Replace Node"

This action is available in the CATIA V5 window. It allows to replace the selected item in the CATIA structure.

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Replace" />
```

It is possible to replace an instance with a new CAD without a native file. In this case the native file is created on the fly in CATIA and saved to Aras Innovator during the next update.

It is possible to replace a node below an embedded CATIA Component.

You do no longer need to open a separate window during replace.

Context Action "Execute"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Execute" usedIn="PdmWindow" />
```

Context Action "Update structure relations"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="UpdateSubCmpRels" usedIn="PdmWindow" />
```

Context Action "Update parent relation"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="UpdateParentCmpRel" usedIn="PdmWindow" />
```

Context Action "Newest Generation Info"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="NewestVersionInfo" usedIn="PdmWindow" />
```

Context Action "Update Item"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="UpdateItem" usedIn="PdmWindow|QueryDialog" />
```

Context Action "PDM Create in Context"

This action is available in the CATIA V5 window. It allows to create a new item directly in the CATIA structure.

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="ContextCreate" />
```

Context Action "Highlight"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="Highlight" usedIn="PdmWindow" />
```

Context Action "Copy element attributes"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="CopyAttributes" usedIn="PdmWindow|QueryDialog" />
```

Context Action "Delete"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="DeleteItem" usedIn="PdmWindow" />
```

Context Action "Synchronize to BOM"

In the Schema file the context action can be enabled.

Example:

```
<contextAction name="SyncStrucFileToPart" usedIn="PdmWindow|QueryDialog"/>
```

A custom server method can be implemented for this functionality. A sample method for this, "Pwb_Sample_SyncCadToBom", exists. The name of the custom method can be configured in PWB Configuration item setting "CustomMethod_SyncCadToBom".

Non-BOM CATParts and CATProducts

In the BOM Part Structure Data Model it is possible to define CATParts and CATProducts in the CATIA structure to be defined as not BOM-relevant. In this case no corresponding Part items will be created in PDM.

The PWB Configuration item setting "UseNonBom3DCadDocs" has to be set to "true":

UseNonBom3DCadDocs	true

Picture 183: Sample UseNonBom3DCadDocs configuration

Performance Optimization for non-BOM CATParts in BOM Part Structure Data Model

When using `UseBomPartStructure=true` and `UseNonBom3DCadDocs=true`, `GetFilteredCadDocsForParts()` first expands the normal CADs for the structure and after this it expands the non-BOM structure for each CAD (CATProduct). This should be done in an optional custom code.

This would allow to improve the performance if

a) there are only free non-BOM CATParts which are not attached to any CATProduct and

b) if there are non-BOM sub-structures only for special CADs.

Configuration

Added server setting `ExpandNonBomCadInBom=false` (default = `true`).

Support for the new CAD Structure Instance Handling introduced in Aras Innovator 9.4 and 10.0

A new relation with the name “CAD Instance” has been introduced, which contains instance information for “CAD Structure” relations.

By default, the new CAD Instance relation is used to store CATIA instance information.

If the setting “UseCadInstance” in the used PWB Configuration item is set to “false” then the old instance handling (using three attributes on “CAD Structure” to store the instance name, the instance description, and the transformation matrix) is used.

UseCadInstance	true

Picture 184: Sample UseCadInstance configuration

“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 CATProduct file is taken, the instance information from PDM is ignored.

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

Configuration

The server setting “UseCadIsMaster” has to be defined and set to “true” to enable the new behavior.

Repair instance with UseCadIsMaster = true

By default, the setting UseCadIsMaster=true causes only the structure from the CATProducts to be used during loading.

The setting UseCadIsMasterRepairInstance=true in the PWB Schema.xml file:

```
<setting name="UseCadIsMasterRepairInstance" value="true" /> <!--
Default value is "false" -->
```

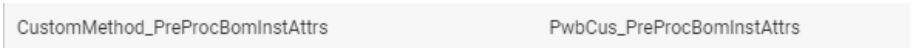
changes this behavior to repair the broken links in the structure. Broken links to be repaired will occur if some UUIDs of the child CATIA files were changed (this could happen if these files are updated / replaced in Aras without using PDM Workbench).

Custom Method for set-based Create of Instance Names

In BOM Part Structure Data Model (PWB Configuration item setting `UseBomPartStructure=true`) the new set-based method replaces the existing method defined in the PWB Configuration item setting `CustomMethod_PreProcRelInstCreAttrs`.

Configuration

To customize the values of the BOM Instance properties (instance name and instance description) you must set the value of the PWB Configuration item setting “CustomMethod_PreProcBomInstAttrs” to the name of your custom method:



Picture 185: Sample CustomMethod_PreProcBomInstAttrs configuration

Sample method to set the instance name:

```
// Sample custom method PwbCus_PreProcBomInstAttrs
//
// PWB Configuration
// CustomMethod_PreProcRelInstances = PwbCus_PreProcBomInstAttrs
//
// This method creates a instance names for BOM Instances that
// uses item_number.
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
   PwbServerApiObj.Log("PwbCus_PreProcBomInstAttrs");

   // Input information
   Item SourceItem = this.getPropertyItem("SourceItem");
   if (SourceItem == null)
   {
      throw new Exception(
         "PwbCus_PreProcBomInstAttrs -> SourceItem == null !");
   }

   Item TargetItem = this.getPropertyItem("TargetItem");
   if (TargetItem == null)
   {
      throw new Exception(
         "PwbCus_PreProcBomInstAttrs -> TargetItem == null !");
   }

   Item RelationItem = this.getPropertyItem("RelationItem");
   if (RelationItem == null)
   {
      PwbServerApiObj.Log("PwbCus_PreProcBomInstAttrs -> RelationItem == null");
   }

   Item Instances = this.getRelationships();
   if (Instances == null)
   {
      throw new Exception(
         "PwbCus_PreProcBomInstAttrs -> Instances == null !");
   }

   // existing instance names for Relation
   List<string> instanceNames = PwbServerApiObj.StringToList(
      this.getProperty("InstanceNames", ""));

   // depending on some other customization it is possible that
   // item_number is already available in TargetItem
   string targetPN = TargetItem.getProperty("item_number", "");
   if (String.IsNullOrEmpty(targetPN))
   {
      // try to get targetPN from existing instance names to
      // avoid additional select in Aras
      if (instanceNames != null && instanceNames.Count > 0 &&
         !String.IsNullOrEmpty(instanceNames[0]))
      {
         PwbServerApiObj.Log("instanceNames[0] -> " + instanceNames[0]);

         // GetFileNameWithoutExtension can be only used if
         // there are no slashes or backslashes
         // in a saved instance name in Aras
         int position = instanceNames[0].LastIndexOf(':');
         if (-1 == position)
         {
            targetPN = Path.GetFileNameWithoutExtension(
```

```
                instanceNames[0]);
            }
            else
            {
                targetPN = Path.GetFileNameWithoutExtension(
                    instanceNames[0].Substring(position + 1));
            }
        }
        else
        {
            // for the first instance we have to fetch the
            // targetPN from Aras
            TargetItem.setAction("get");
            TargetItem.setAttribute("select", "item_number");
            Item Result = TargetItem.apply();
            targetPN = Result.getProperty("item_number");
        }
    }

    // The Characters ':' and '!' cannot be used inside an instance
    // name
    targetPN = targetPN.Replace(':', '_').Replace('!', '_');

    if (Instances.isCollection())
    {
        PwbServerApiObj.Log(
            "PwbCus_PreProcBomInstAttrs -> Instances == Collection");

        int j = 1;
        for (int r = 0; r < Instances.getItemCount(); r++)
        {
            var Instance = Instances.getItemByIndex(r);
            PwbServerApiObj.Log(
                "PwbCus_PreProcBomInstAttrs -> " + "Instances -> Instance " + r);

            string existingEmbeddedPrefix = "";
            var origInstancename = Instance.getProperty(
                "pwb_cad_instance_name", "");
            int position = origInstancename.LastIndexOf(':');
            if (position != -1)
            {
                existingEmbeddedPrefix =
                    origInstancename.Substring(0, position + 1);
            }

            PwbServerApiObj.Log(
                "PwbCus_PreProcBomInstAttrs -> " +
                "Instances -> origInstancename " + origInstancename);

            for (int i = j; ; i++)
            {
                if (instanceNames == null ||
                    !instanceNames.Contains(existingEmbeddedPrefix
                        + targetPN + "." + i))
                {
                    Instance.setProperty(
                        "pwb_cad_instance_name",
                        existingEmbeddedPrefix + targetPN
                        + "." + i);
                    j = i + 1;
                    break;
                }
            }
        }
    }
    else
    {
        PwbServerApiObj.Log(
            "PwbCus_PreProcBomInstAttrs -> " + "Instances == Singel Item");

        string existingEmbeddedPrefix = "";
        var origInstancename = Instances.getProperty(
            "pwb_cad_instance_name", "");

        int position = origInstancename.LastIndexOf(':');
        if (position != -1)
```

```
      {
         existingEmbeddedPrefix =
            origInstancename.Substring(0, position + 1);
      }
      for (int i = 1; ; i++)
      {
         if (instanceNames == null || !instanceNames.Contains(
            existingEmbeddedPrefix + targetPN + "." + i))
         {
            Instances.setProperty("pwb_cad_instance_name",
               existingEmbeddedPrefix + targetPN + "." + i);
            break;
         }
      }
   }

   return this;
}
```

Configurable CATIA Components Support

It is possible to load and update CATProduct structures which contain embedded CATIA components. Depending on the Part Number prefix the embedded component nodes can either be “skipped”, that is, the node is treated as if it does not exist, but its child nodes are processed, or they can be “ignored”, that is, the node and all its child nodes are treated as if they do not exist.

The configurable node behavior functionality has to be switched on by defining the Schema file setting “catiaNodeBehaviorDefinitions”:

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

Optional.

The behavior settings “SkipNode” and “IgnoreNode” are allowed.

Support Electrical / Tubing

With this functionality it is possible to use functions like “Electrical Harness”, “Electrical Wire Routing”, “Piping Design”, “Tubing Design”, … of the CATIA “Equipment & Systems Engineering” section.

In the Schema file the following settings have to be made:

a) To allow leaf components of any type:

```
<catiaNodeBehaviorDefinitions ignoreLeafComponents="true" />
```

b) To allow special types of components:

```
<catiaNodeBehaviorDefinitions ignoreLeafComponents="false" >
   <catiaNodeBehavior catiaNodeType="ElecWireLight"
                      behavior="IgnoreNode" />
</catiaNodeBehaviorDefinitions>
```

Support for Relating a new CATIA File to an existing Part

The currently active CATIA document (only CATParts or CATDrawings) can be related to an existing BOM Part item. If there is already a corresponding CAD Document related to the Part the document's file can be overwritten.

For Part items the context action "Relate active File to Part" can be added in the Schema file.

Example:

```
<contextAction name="RelateToPart" usedIn="QueryDialog" />
```

Local Workspace Information

It is possible to check the status of the CATIA documents which are downloaded to the local working directory (PWB_XMAP). A list displays the local files and information about their corresponding CAD Documents in PDM if they exist.

The icon which opens the window which contains the list of files/documents can be switched on by removing the entry "LocalWorkspace" from the "removeToolbarIcons" definition in the Schema file.

Example:

```
<removeToolbarIcons>
   <!-- <icon name="LocalWorkspace" /> -->
   <icon name="Register" />
   <icon name="Synchronize" />
   <icon name="NewPwbWindow" />
</removeToolbarIcons>
```

Optional.

The PDM attributes which are shown can be configured in the Schema file.

This is an example configuration:

```
<form name="LocalWorkspace" defaultSortAttribute="filename">
   <formAttribute name="modified" visibleLength="8" required="false"
                  listViewRelevant="true" />
   <formAttribute name="filename" visibleLength="10" required="false"
                  listViewRelevant="true" />
   <formAttribute name="item_number" visibleLength="10"
                  required="false" listViewRelevant="true" />
   <formAttribute name="major_rev" visibleLength="6" required="false"
                  listViewRelevant="true" />
   <formAttribute name="generation" visibleLength="6" required="false"
                  listViewRelevant="true" />
   <formAttribute name="name" visibleLength="10" required="false"
                  listViewRelevant="true" />
   <formAttribute name="state" visibleLength="10" required="false"
                  listViewRelevant="true" />
   <formAttribute name="description" visibleLength="10"
                  required="false" listViewRelevant="true" />
   <formAttribute name="created_on" visibleLength="10"
                  required="false" listViewRelevant="true" />
```

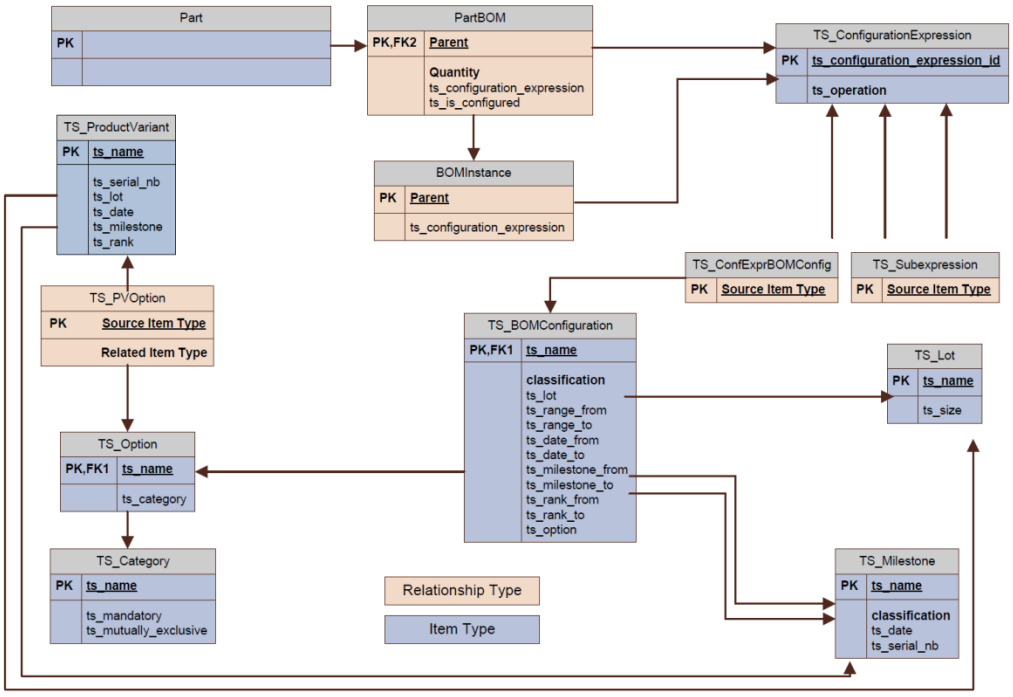
```
    <formAttribute name="modified_on" visibleLength="10"
                   required="false" listViewRelevant="true" />
    <formAttribute name="created_by_id" visibleLength="10"
                   required="false" listViewRelevant="true" />
    <formAttribute name="modified_by_id" visibleLength="10"
                   required="false" listViewRelevant="true" />
    <formAttribute name="locked_by_id" visibleLength="10"
                   required="false" listViewRelevant="true" />
</form>
```

Configuration of BOM Part Structure

In the BOM Part Structure Data Model it is possible to create product configurations where, depending on the currently set configuration context, only a sub-set of the product structure is expanded and loaded. With this functionality it is possible to create and to work on different configurations of the same product.

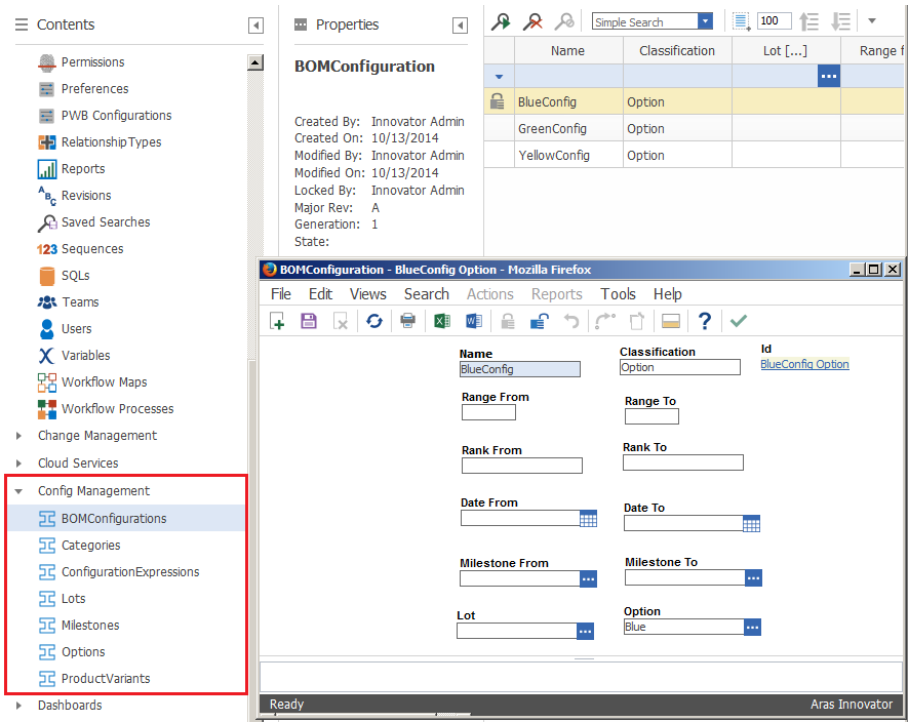
Using the BOM Configuration Management costs additional license fees.

In addition to the four standard PDM Workbench packages which have to be imported with the Aras Innovator import tool (ArasPwbDM, ArasPwbDM_PLM, ArasPwbDM_Core, and ArasPwbDM_Cui_default), four additional packages have to be imported: BomConfig, BomConfig_PLM, BomConfig_Core, and BomConfig_Cui_default. These four packages contain the item types which are used by the BOM Configuration Management functionality:



Picture 186: BOM Configuration Management Type Overview

Importing the packages will enable the administrator to access the types in the Aras Innovator web client and to use them to create different product configurations.



Picture 187: BOM Configuration Management Types in Aras Innovator

Possibility to call a Server Method for a PDM Item

It is possible to call custom server methods with a PDM item and optionally with a dialog as input.

The Schema file has to contain the line

```
<customContextAction name="PwbCustomServerMethod"
                     usedIn="PdmWindow|QueryDialog"
                     confirm="true"
                     multiple="false"
                     dialog="Query" />
```

after the `<contextAction ... />` definitions of an object.

The *dialog* attribute is optional and refers to a “form” definition of the same “object” in the Schema file.

The *confirm* and the *multiple* attributes are also optional, and they make it possible to not show the confirmation dialog to the user (`confirm="false"`) and to pass all items of a multi-selection to the method at once, and only call the method once (`multiple="true"`).

If the file “PWBSchemaDisplayNames_Aras_Aras.CATNls” contains the NLS string for the server method, for instance

```
PwbCustomServerMethod = "Custom Server Method";
```

then the NLS name is shown to the user.

Please check “Pwb_Sample_ContextAction” for a sample on which you can build your own custom action code (`multiple="false"`).

Reconnect at Update

Up to PDM Workbench version 20.0 the “Reconnect” the administrator had to add the property “pwb_orig_cad_partnumber” to CAD, or in case of BOM Part Structure mode to Part. Starting with PWB 21.0 the property will be set during import of the PWB data model. (If the property pwb_orig_cad_partnumber was already added manually, the administrator must remove this property from the import package ArasPwbDM_PLM in the files: “ArasPwbDM_PLM\PLM\Import\ItemType\CAD.xml" and “ArasPwbDM_PLM\PLM\Import\ItemType\Part.xml”).

By default, PDM Workbench saves the original Partnumber from the CAD system to the property pwb_orig_cad_partnumber. To use a different property, you can use the PWB setting “OrigCadPartnumberAttr” in the PWB Configuration:



Picture 188: Sample OrigCadPartnumberAttr configuration

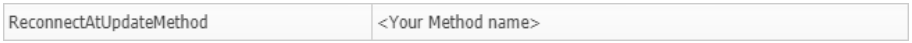
To prevent the original part number from the CAD system from being saved, you can set the “OrigCadPartnumberAttr” property to an empty value.

The following setting in the Schema file enables the “Import with Reconnect” button in the “Update to PDM” dialog:

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

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 item setting “ReconnectAtUpdateMethod”:



Picture 189: PWB Configuration item setting “ReconnectAtUpdateMethod”

A sample method for this, “Pwb_Sample_CadTreeIconInfo”, exists. You can use this as the base for your own implementation.ost

Sample

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

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{

Innovator InnovatorObj = this.getInnovator();

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

PartnumbersToIgnoreHashSet.Add("Part1");
PartnumbersToIgnoreHashSet.Add("Part2");
PartnumbersToIgnoreHashSet.Add("Part3");
PartnumbersToIgnoreHashSet.Add("Part4");
PartnumbersToIgnoreHashSet.Add("Part5");
PartnumbersToIgnoreHashSet.Add("Part6");
PartnumbersToIgnoreHashSet.Add("Part7");
PartnumbersToIgnoreHashSet.Add("Part8");
PartnumbersToIgnoreHashSet.Add("Part9");
PartnumbersToIgnoreHashSet.Add("Product1");
PartnumbersToIgnoreHashSet.Add("Product2");
PartnumbersToIgnoreHashSet.Add("Product3");
PartnumbersToIgnoreHashSet.Add("Product4");
```

```
PartnumbersToIgnoreHashSet.Add("Product5");
PartnumbersToIgnoreHashSet.Add("Product6");
PartnumbersToIgnoreHashSet.Add("Product7");
PartnumbersToIgnoreHashSet.Add("Product8");
PartnumbersToIgnoreHashSet.Add("Product9");


// 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;
}
```

Clean up / Housekeeping of PWB_XMAP Directory

The XMAP is a directory, which is used as a local storage by the PDM Workbench. It must exist and its path must be defined either by the environment variable “PWB_XMAP” or by the *xmap* value attribute of the Schema file.

The XML tag *xmap* is optional. If it does not exist in the Schema file, the XMAP path must be defined by the environment variable “PWB_XMAP”. If both values are set (environment variable and xml XMAP attribute *value*), the environment variable is taken.

An additional optional attribute (*sizeThreshold*) exists for the *xmap* element in the Schema file. If the xmap attribute *sizeThreshold* exists in your schema and represents a numerical value larger than 0, the PDM Workbench will execute some clean-up actions at the end of the Load and Update commands.

A numerical value for *sizeThreshold* larger than “0” means a size in megabyte. If the sum of the file sizes of all CATIA and jpg files in the exchange map directory is larger than this threshold, then some files, **which are not needed by your current CATIA session**, are automatically deleted - from oldest to newest - until the remaining file size sum is less than the threshold.

If you want to switch off the automatic clean-up actions, you can set the *sizeThreshold* to “0” or remove the xmap attribute *sizeThreshold* from your Schema file.

To enable the automatic clean-up functionality, you have to add the attribute *sizeThreshold* to the XML tag *xmap* and define a numerical value larger than “0” for the attribute *sizeThreshold*, which is the size in Megabyte. A value of “0” means that the functionality is switched off.

Example:

```
<PWBSchema system="Aras" customization="Aras"
           displayName="NLS_System"
           visibleLength="15" allowedLength="64">
   ...
   <xmap value="C:\PWB_XMAP" sizeThreshold="200" />
```

This means, that your exchange map directory is “C:\PWB_XMAP” and a *sizeThreshold* of 200 Megabyte is active for this directory.

Example of setting a system environment variable and defining an XMAP *sizeThreshold*:

```
   <xmap value="%LOCALAPPDATA%\PWB_XMAP" sizeThreshold="200" />
```

This means, that your exchange map directory is “%LOCALAPPDATA%\PWB_XMAP” (as resolved by the Windows system) and a *sizeThreshold* of 200 Megabyte is active for this directory.

A numerical value for *sizeThreshold* larger than “0” means a size in Megabyte. If the sum of the file sizes of all CATIA and jpg files in the exchange map directory is larger than this threshold, then some files, **which are not needed by your current CATIA session**, are automatically deleted at the end of the Load and Update commands - from oldest to newest - until the remaining file size sum is less than the threshold.

If you want to switch off the automatic clean-up actions, you can set the *sizeThreshold* to “0” or remove the XMAP *sizeThreshold* attribute from the Schema file.

Attention:
The clean-up process will potentially **delete** all CATIA and jpg files from the exchange map directory, which are not needed by your current CATIA process. If you activate the *sizeThreshold*, you will get an automatic clean-up, but you have to make sure that all users do not manually store files in that directory and that no other CATIA process is using the same directory at a time, because the files there could be deleted.

The exchange map directory should be exclusively used by the PDM Workbench in a single CATIA process, if you activate the *sizeThreshold*.

Setting Configuration Information on Structure Relations

It is possible to set configuration information (configuration expression items) to “Part BOM” and “BOM Instance” relations in the PDM structure window.

The PWB Configuration item setting “UseSeparateRelationsForInstances” has to be set to “true”:



Picture 190: Sample UseSeparateRelationsForInstances configuration

In the Schema file a *contextAction* tag with the name “SetConfigInfoOnRelations” has to be added to the “Part BOM” relation definition.

Example:

```
<contextAction name="SetConfigInfoOnRelations" usedIn="PdmWindow" />
```

Also, the definition of the “TS_ConfigurationExpression” object has to be added to the Schema file.

Example:

```
<object name="TS_ConfigurationExpression"
        displayName="NLS_TS_ConfigurationExpression"
        icon="Aras_BomConfigItem" >
   <description>
```

```
      <descAttribute name="ts_name" />
   </description>
   <action name="Query" />
   ...
   <form name="Query">
      <formAttribute name="ts_name" widgetType="SingleLineEditor"
                     mode="update" visibleLength="15"
                     required="false" listViewRelevant="true" />
      <formAttribute name="ts_operation"
                     widgetType="SingleLineEditor" mode="update"
                     visibleLength="15" required="false"
                     listViewRelevant="true" />
   </form>
   ...
</object>
```

Excluding Child Node Types for Creation

Before a new CATIA structure is imported to PDM the desired type of the PDM node can be defined. Generally, all the available PDM types can be set on the child nodes.

With this functionality it is possible to exclude certain types from being shown in the context menu. The setting “DisallowChildNodeTypesForCreate” can be defined for an assembly node, and the list of disallowed children PDM node types can be defined as the setting value, divided by the “|” character.

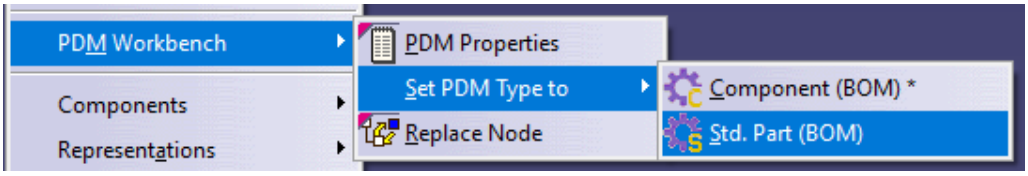
Example:

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

   <settings>
      <setting name="DisallowChildNodeTypesForCreate"
               value="/Part/Standard|/Part/StdContainer"/>
   </settings>

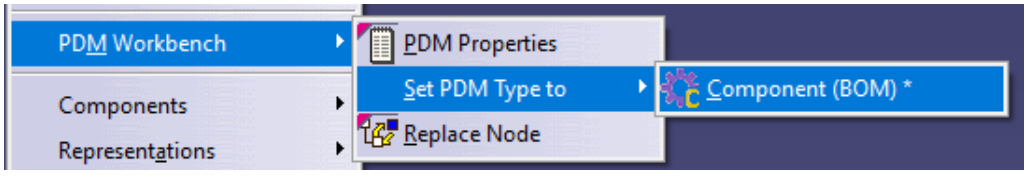
   <description>
   ...
```

Here the context menu is shown without the setting above:



Picture 191: All PDM Types can be selected

Here the context menu is shown with the setting above:



Picture 192: Only a sub-set of the PDM Types can be selected

Use Server Method for Quantity

For some relations it may be necessary to set the quantity manually instead of using the quantity of the CATIA PSN structure.

Configuration

Create a server method that counts the relevant BOM Instances, example “PwbCus_UpdateQuantity”:

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

string Type = this.getType();

if (Type == "Part BOM")
{
   int count = PwbServerApiObj.GetRelationshipCount(this, "BOM Instance");
   this.setProperty("quantity", count.ToString());
   return this.apply("update");
}
else
{
   return this.getInnovator().newError("PwbUpdateQuantity: Relation Type: '"
                                        +Type+"' not supported");
}
```

The name of the server method must be set in the PWB Configuration item setting “UpdateQuantityMethod”:

UpdateQuantityMethod	PwbCus_UpdateQuantity

Picture 193: Sample UpdateQuantityMethod configuration

Released Cache Mode

The PDM Workbench can efficiently use CGR cache files, which are stored in the Aras Innovator database, for the geometrical visualization of product structures. This is useful when loading large product structures in visualization mode to CATIA.

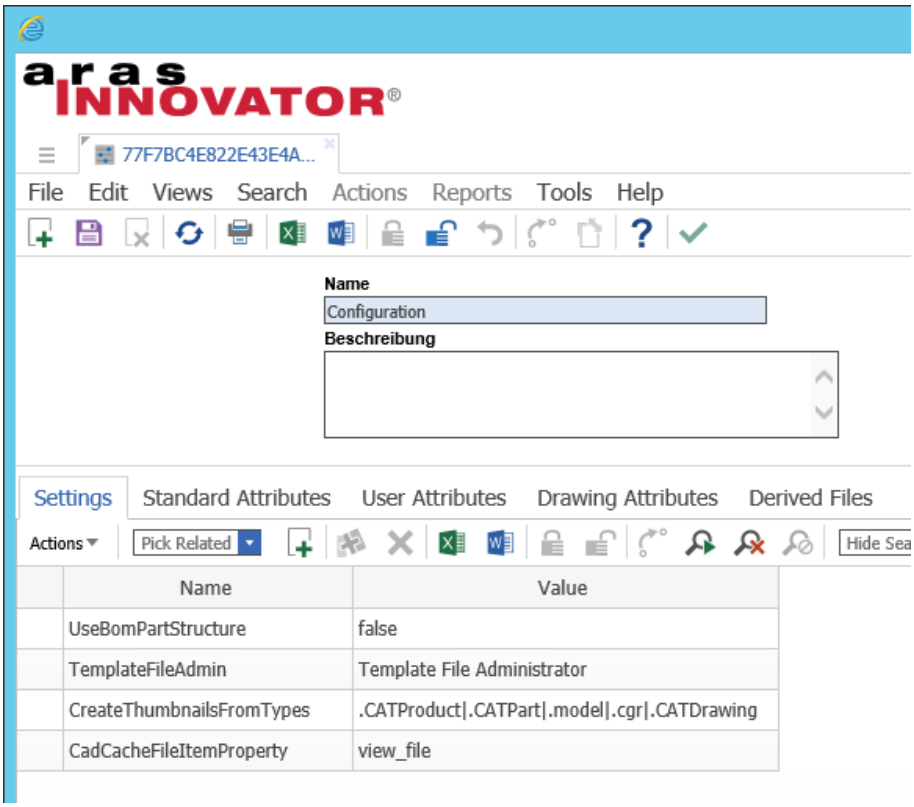
The PWB cache mode uses CATIA tools to create CGR cache files based on native files like CATIA V5 CATParts or CATIA V4 models, uses Aras Innovator to store and manage these files linked to a CAD Document item in Aras Innovator and uses the CATIA released cache capability to provide the data to the user when loading a product structure.

Step1: Define the File item property for cache files

The CGR cache files have to be stored in PDM on a File item property of the CAD Document item or the customized item type that you use instead of the Item Type “CAD”.

There already exist several File item properties (Data Type = Item, Data Source = File) on the Aras Innovator Item Type “CAD”. The property “native_file” is reserved for the original file, e.g. a CATPart. You can select one of the others (e.g. “viewable_file”, “view_file”, etc.) or customize the Item Type “CAD” to provide an additional property for the PWB cache files, e.g. “pwb_cache_file”.

We use the “view_file” property in our example and therefore set “CadCacheFileItemProperty” to “view_file” in the PWB configuration settings on the Aras Innovator server.



Picture 194: Sample CadCacheFileItemProperty configuration

If “CadCacheFileItemProperty” does not exist in your configuration settings or its value is an empty string, then the PWB cache mode is recognized as switched off.

Step 2: Create the cgr cache files

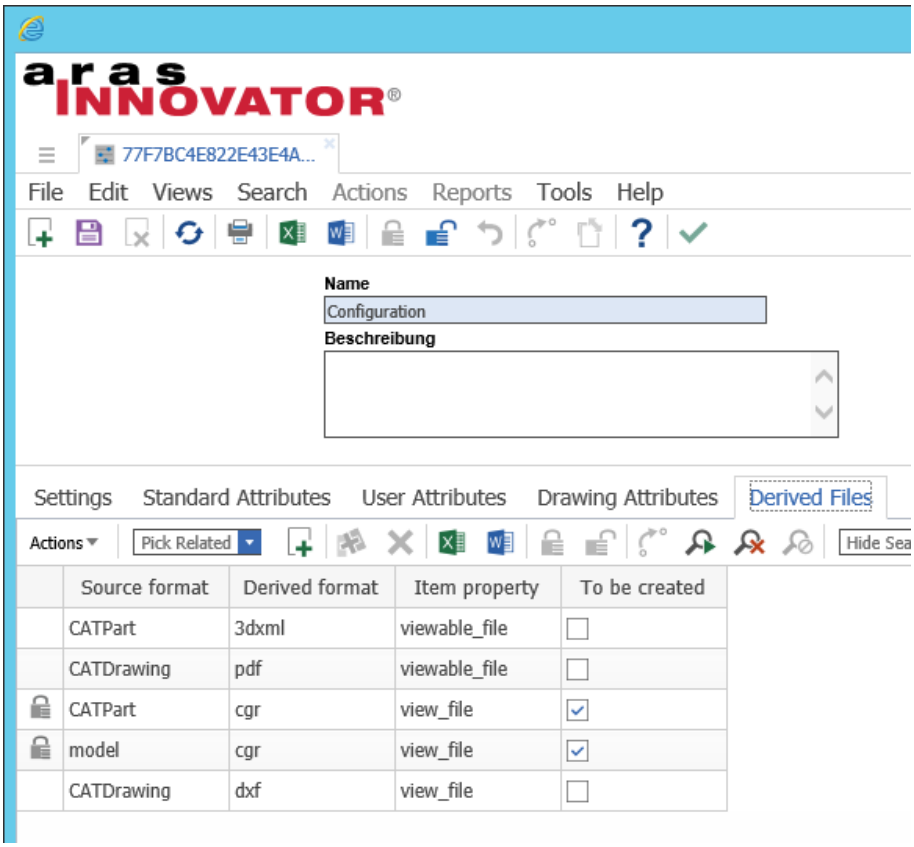
The CGR cache files can be created in an external process like the regular CATIA CATDMUUtility or on the user’s machine when uploading a native file (CATPart, model) or as a combination of both processes, e.g. CATDMUUtility for existing objects and the user’s machine for any new update.

External process:

You can use the Aras Conversion Server capability and setup a task which extracts the native file from a CAD Document item of your choice, runs the CATIA CATDMUUtility to generate a CGR file for that native file and then stores the resulting CGR file on the File item property of your choice (s. above).

Internal process:

The PWB provides the capability to create and store derived files when a user is uploading new or modified files to Aras Innovator. These derived files are created in a CATIA SaveAs operation of the current PWB user’s CATIA session. Each SaveAs operation must result in a single file with the same name as the source file but a different suffix.



Picture 195: Setting the derived file creation to “view_file”

Some examples are provided in the PDM Workbench Aras Innoator data model and can be picked and modified to your requirements. You can also create new lines, as these are regular Aras Innovator relations.

This feature can be used to automatically create and store a CGR cache file for some selected native file formats. Take care that the defined item property value for your CGR files corresponds to the “CadCacheFileItemProperty” value that you defined above. Otherwise, the CGR files would be created and stored in Aras Innovator but cannot be used as PWB cache files (which could also be a scenario).

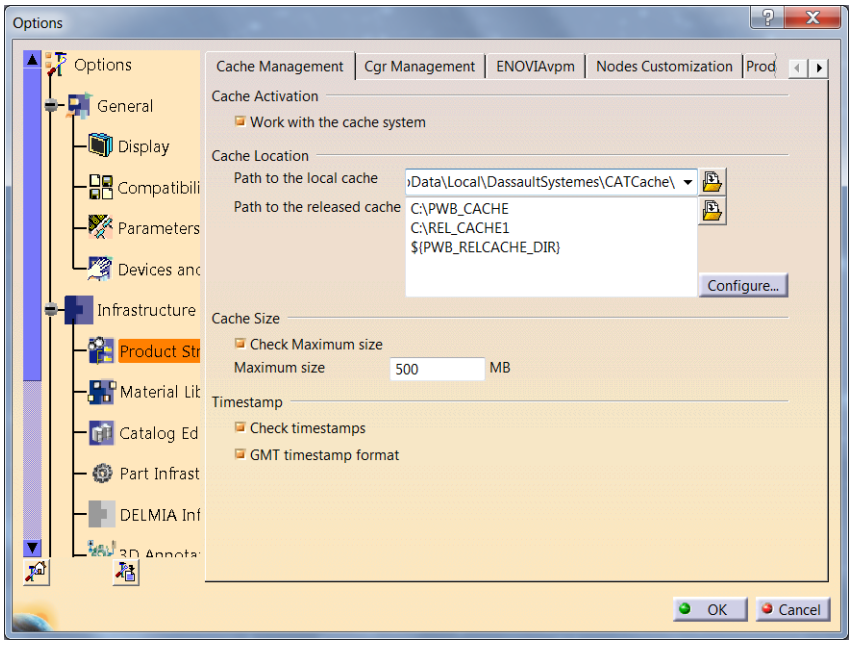
Step 3: Provide and declare a CATIA released cache directory for PWB

The user or an administrator has to create a local directory on the user’s machine, which is exclusively used for PWB cache files (e.g. C:\PWB_CACHE).

Furthermore, this directory has to be declared in the CATIA Tools+Options released cache setting (s. examples in the picture below).

If you have declared multiple directories in that CATIA setting list and you cannot or do not want to use the first directory in the list as your PWB released cache directory, then you can set the system environment variable “PWB_RELCACHE_DIR” to the directory string that you want to use. If the environment variable “PWB_RELCACHE_DIR” is not set, then the first directory in the list is used.

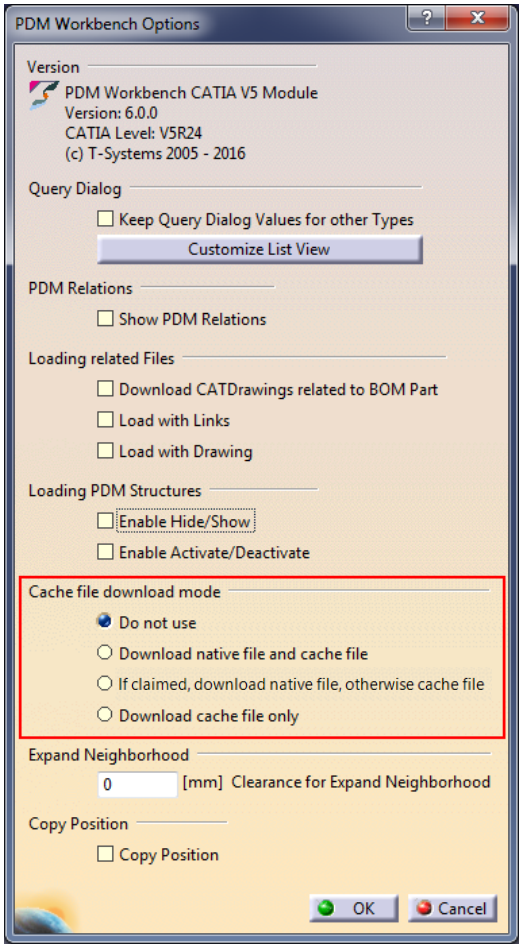
You need a CATIA DMU Navigator license to configure or view this released cache setting in CATIA.



Picture 196: Defining the CATIA Released Cache directory

Step 4: Set the cache file download mode in user settings

Each user can define the personal download mode related to cache files in the PDM Workbench Options.



Picture 197: Setting the Released Cache PWB options

You should set it according to your regular or current task. Any change will have immediate effect - from the next download on - for files, which do not exist in the local PDM Workbench exchange directory or are out of date there.

- **Do not use**
  This is the mode of previous PDM Workbench releases. Only native files will be downloaded from the server, even if cache files are provided there.

  This mode can still make sense, if you regularly perform a detailed design based on a small number of loaded files.

- **Download native file and cache file**
  Both files are downloaded for each CAD item, the native file, and the cache file.

  This mode makes sense, if you want a fast initial visualization of your product, but usually switch most or all of it to design mode later.

- **If claimed, download native file, otherwise cache file**
  The download will distinguish between items, which are claimed by the current user, and other files.

  It only makes sense for small product structures and if your work methodology is in a way, that you generally claim all items in PDM that you expect to work on, before you load a product structure to CATIA.

- **Download cache file only**
  The native file would not be locally available for immediate access but can be downloaded on demand later.

  This mode makes sense for design reviews, where you need the visualization data only, and for loading large product structures, where you might switch only a small portion of it to design mode later.

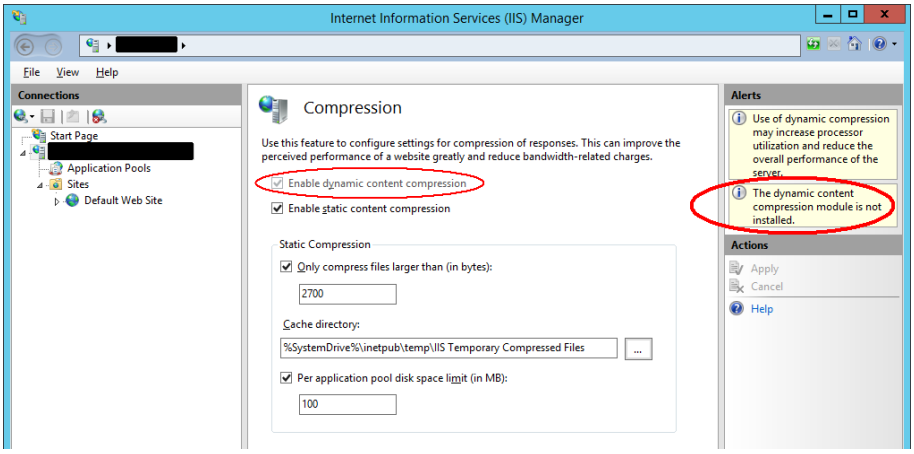
HTTP Compression

The dynamic content compression of http requests and responses is supported.

Configuration

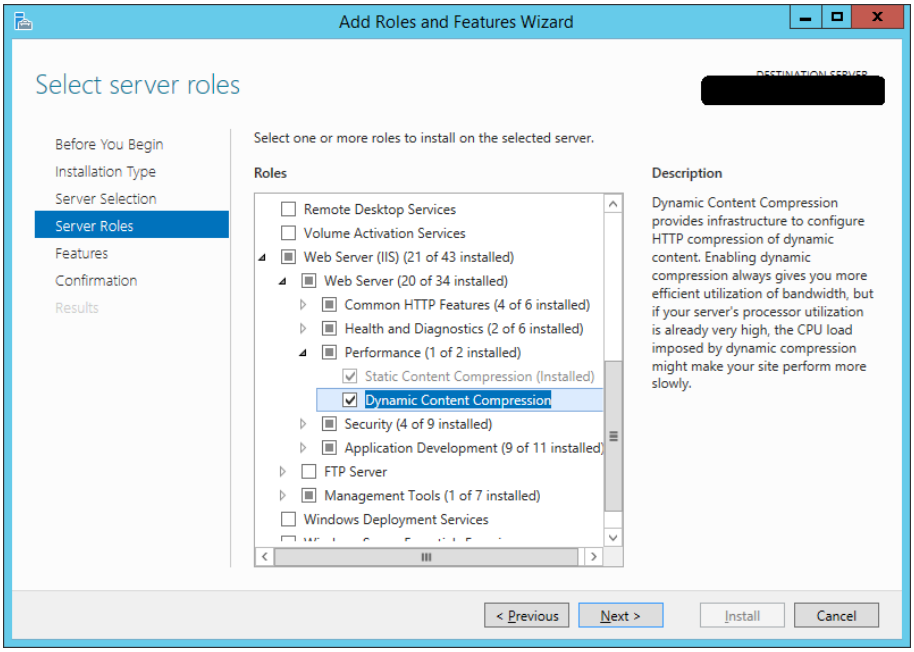
The dynamic content compression feature of the Microsoft IIS has to be installed and activated on the Aras Innovator server(s).

Check that the dynamic content compression is activated (see *Picture 198: IIS – Compression settings*).



Picture 198: IIS – Compression settings

Install the “Dynamic Content Compression” (see *Picture 199: IIS - Install dynamic compression module*).



Picture 199: IIS - Install dynamic compression module

Set the environment variable `PWB_HTTP_COMPRESSION` to “gzip” or “deflate” for the PDM Workbench Client processes.

“Open in CATIA” from the Aras Innovator Client

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

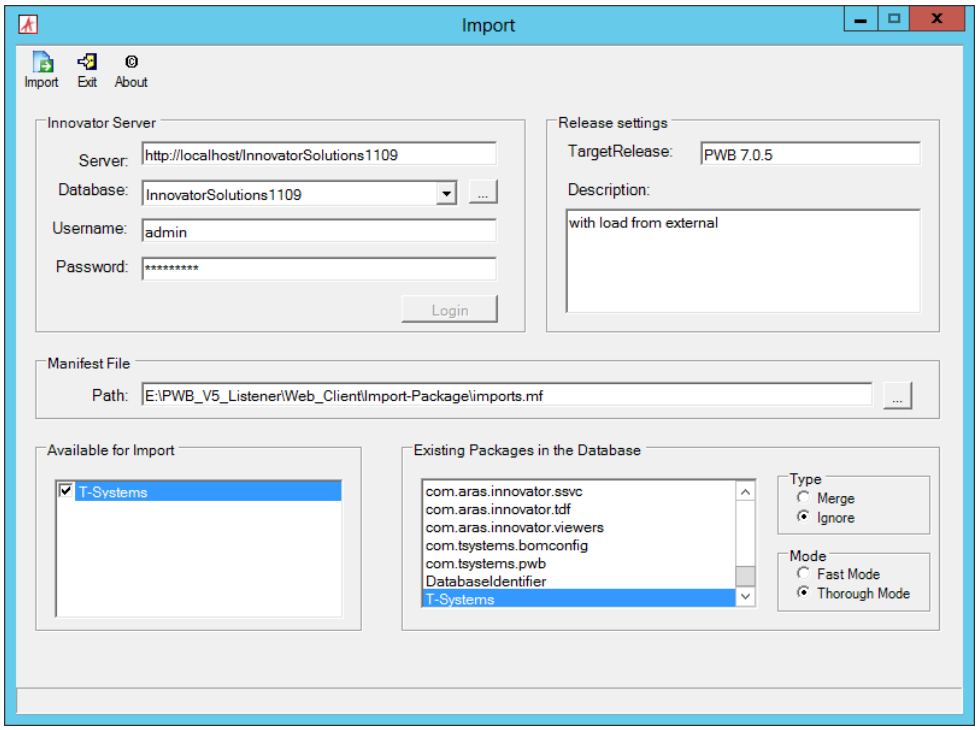
The precondition for this functionality is that CATIA V5 is started with PDM Workbench, and the user is logged in.

Configuration in Aras Innovator

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

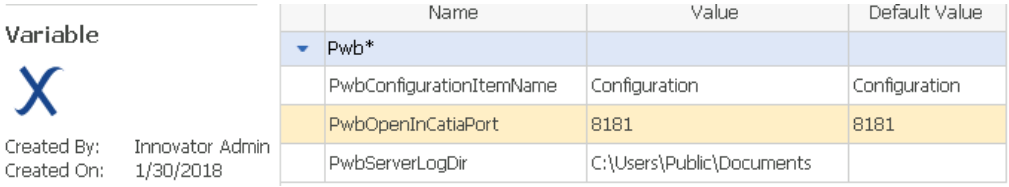
“PwbDataModel_[XX.X]\Specific\OpenInCAD\OpenInCATIA\imports.mf” -
The "Open in CATIA" functionality.

“PwbDataModel_[XX.X]\Specific\OpenInCAD\OpenInCATIA_BomConfig\imports.mf” -
Additional package if the "BOM Part Structure Configuration" functionality is installed.



Picture 200: Import utility

In Aras Innovator the variable “PwbOpenInCatiaPort” has to be set to a specific port, for example “8181”, which is used by the listener service:



Picture 201: Innovator variables

Installation of the PDM Workbench Listener

The listener is installed during the installation of the CATIA V5 package for the PDM Workbench.

The installation steps can be done manually, too. Please see the following section for details.

In this example, the PDM Workbench client is installed in this directory:

```
C:\Program Files\T-Systems\PWBCATV5_R34_20.0.0_Aras_37
```

So, for a successful installation of the listener service these two files have to be in `C:\Program Files\T-Systems\PWBCATV5_R34_20.0.0_Aras_37\win_b64\code\bin`

- PwbListenerService.exe
- PwbListenerService.cfg

The log behavior of the PWBListener service can be configured in two ways:

1. Using a configuration file:
   The PWBListener searches for the configuration file in the following places:
   a) use the environment variable `PWB_LISTENER_CONFIG_FILE=<full path to config file>`

b) If not found use "PwbListenerService.cfg" in the directory of PWBListenerService.exe.

c) If not found use "<PWB Client installation>\config\PwbListenerService.cfg" (To be exact it uses the path "..\..\..\config\PwbListenerService.cfg" starting from the directory of PWBListenerService.exe)

The Log level can be configured in the "PwbListenerService.cfg" file with the setting:
`LOG_LEVEL=<ERROR|INFO|DEBUG>` (default: `ERROR`)

The log directory can be configured in the "PwbListenerService.cfg" file with the setting:
`LOG_DIR=<log directory>` (default: temp directory of the service user, normally "C:\Windows\temp")

2. Using environment variables:
Every environment variable will overwrite the corresponding setting in the config file.

Set the log level:

`PWB_LISTENER_LOG_LEVEL= <ERROR|INFO|DEBUG>` (default: `ERROR`)

Set the log directory:
`PWB_LISTENER_LOG_DIR=<log directory>` (default: temp directory of the service user, normally "C:\Windows\temp")

The "PwbListenerService.cfg" configuration file has also to contain values for the port number, the CATIA V5 installation directory, and the full path of the PDM Workbench CATIA environment file, which was created during the PDM Workbench installation.

The PDM Workbench installation directory, and the PDM Workbench exchange map directory are optional values.

This is an example content of that configuration file:

```
------------------------------------------------------------------
LISTENER_PORT=8181
CATIA_DIR=C:\Program Files\Dassault Systemes\B34
CATIA_ENV_FILE=C:\Program Files\T-
Systems\PWBCATV5_R34_20.0.0_Aras_37\config\pwbcatiaenv.txt
------------------------------------------------------------------
```

Install the PwbListenerService.exe as a Service

Open a command (cmd) window as administrator.

A windows service can be installed with the sc tool or the .NET installutil, please refer to:

https://docs.microsoft.com/en-us/dotnet/framework/windows-services/how-to-install-and-uninstall-services

or

https://support.microsoft.com/en-us/help/251192/how-to-create-a-windows-service-by-using-sc-exe

Installation with sc.exe:

```
>sc create PwbListenerService binPath= "%PWB_DIR%\win_b64\code\bin\PwbListenerService.exe"
start= auto DisplayName= PWBListener
>sc description PwbListenerService "Triggers PDM Workbench in CATIA V5 to load a structure
from Aras Innovator into CATIA V5"
>sc start PwbListenerService
The service can be uninstalled again with
>sc delete PwbListenerService
```

Installation with .NET installutil.exe:

```
>C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe
%PWB_DIR%\win_b64\code\bin\PwbListenerService.exe
The service can be uninstalled again with
>C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe /u
%PWB_DIR%\win_b64\code\bin\PwbListenerService.exe
```

Please check in the services.msc snapin that the PWBListener is running and start the service if necessary.

For changing values in the configuration file “PwbListenerService.cfg” stop the PWB listener service, change the values in the file, and start the service again.

“Open in CATIA” uses dynamic Setting how to fetch Properties from Aras Innovator

By default, Open in CATIA expects all needed PDM properties to be available in the AML passed to the PWBListener via Web Socket.

If you create a custom AML for example from an external DMU application and you don’t have all necessary properties, it is possible to send a lean AML that only holds the structure defining properties.

Configuration

To enable this feature, you must add the property “FetchPropertiesDuringLoad” to the PWBToCatiaCmd or the PWBAddTempToCatiaCmd command:

```
// default, open expanded structure
var command = "COMMAND:PWBToCatiaCmd";

// change command to "PWBAddTempToCatiaCmd"
if (addTemp === true)
{
   command = "COMMAND:PWBAddTempToCatiaCmd";
}

// advise PWB to fetch the properties from the db during load
command += "|FetchPropertiesDuringLoad";

socket.send(command);
socket.send(structure);
socket.close();
```

If you activate the PWB Configuration item setting `SendToCadUseAmlAttributes=false`, the properties are always fetched from PDM during load.

“Open in CATIA” with Construction Spaces

The Construction Space feature enables the user to set a construction space (assembly/installation space) to filter CAD structures.

The construction space itself represents an axis-aligned bounding-box which is defined by a min point and a max point. This axis-aligned bounding box represents a virtual three-dimensional cuboid, whose sides are the boundaries for the construction space.

Only structures which are located inside the construction space or intersect the construction space are loaded into the CAD application. Because of that the Construction Space feature is integrated into the “Open in CATIA” functionality. The feature is compatible with both CAD and BOM mode for “Open in CATIA”.

Configuration

When installing the “Open in CATIA” data model the Construction Space feature is already included. Configurations might need to be done at the “PwbConstructionSpace” life cycle. *Please be aware that names of the life cycle states should not be modified to ensure the functionality of the Construction Space feature.* The identities to promote life cycle states can be modified to your needs.

Access Rights

In the TOC, the Construction Space can be found at
Administration ➔ PDM Workbench ➔ Construction Space.

The identity “All Employees” has access to the Construction Space ItemType. Restrictions can be made via the TOC editor.

The usage of the Construction Space ItemType is recommended for the usage of the Construction Space feature.

Construction Space items which are in the “Released” state can be accessed by all users. In the “Draft” state only the creator can use the Construction Space for testing. If only creators should be allowed to access the Construction Space, then it should be left in the “Draft” state.
In the “Obsolete” state the construction space is no longer shown.

All users of the “Aras PLM” identity can promote the life cycle state of the construction space. The identities to promote life cycle states can be modified in the life cycle map.

“Open in Aras” in CATIA V5 Client

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

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 202: File “deepLinking.ts or .js”

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

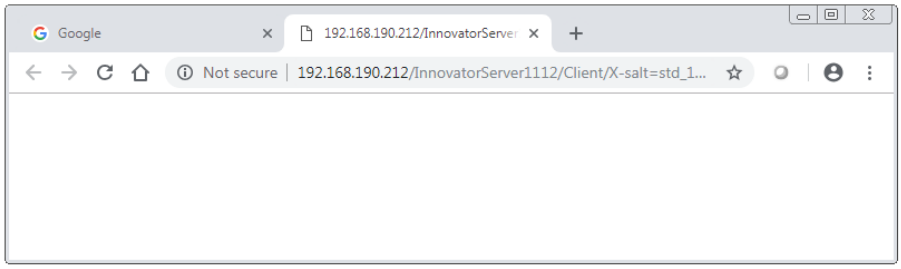
At all clients the browsers cache of the default browser must be cleared to enforce the use of the changed script.

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

For later Aras Innovator version this behavior may be changed.

For Aras Innovator 11 only

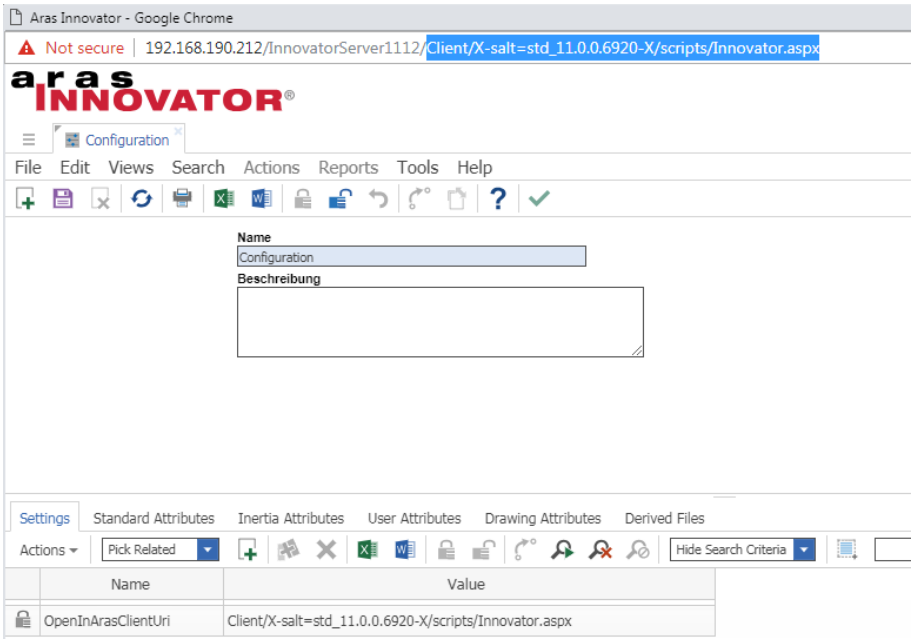
With Aras Innovator 11 the “Open in Aras” functionality opens a new empty tab in an additional default browser window.



Picture 203: Empty tab

To avoid this window, you must set the PWB Configuration “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 204: Aras Innovator web client with Client URL

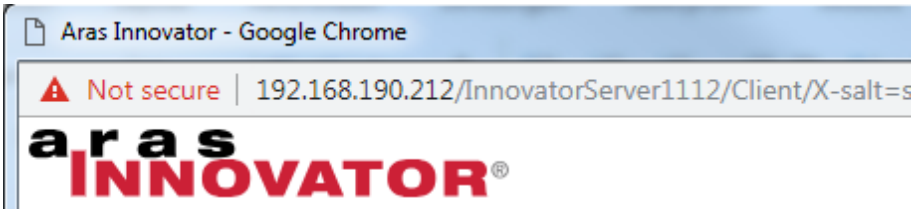
This setting must not exist in Aras Innovator 12.

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 205: Window title "Aras Innovator"

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

ArasWindowTitleSubString	Aras Innovator

Picture 206: Sample ArasWindowTitleSubString configuration

"Open in Aras" in AICD environment

The AICD Cloud environment is split up in different URLs for Server, Client, CAD Client, Authentification server and Vault.

Example:

Your AICD instance lives at

https://aicdpartner.aras.com/tsystems/

Your server endpoint is:

https://s-aicdpartner.aras.com/tsystems/

To use this server you have to configure `PWB_SOAP_TARGET_URL` like:

```
SET PWB_SOAP_TARGET_URL=
   https://s-aicdpartner.aras.com/tsystems/InnovatorServer.aspx
```

By default, the PDM Workbench uses the same URL to open an item in the web client. In case of AICD environment you need a different URL to send an Item to the web client, therefore you must set `PWB_ARAS_CLIENT_TARGET_URL` in the start script of CATIA:

```
SET PWB_ARAS_CLIENT_TARGET_URL=
   https://aicdpartner.aras.com/tsystems
```

"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 CATIA client (this can be done in the environment script of the PDM Workbench "catia_env.bat")
`SET PWB_WEBBROWSER=<browser>`

Optional a second environment variable can be set to specify the browser options:
`SET PWB_WEBBROWSER_OPTIONS=<browser options>`

Example:

```
SET PWB_WEBBROWSER=msedge
```

```
SET PWB_WEBBROWSER_OPTIONS=--inprivate
```

Configurable Properties for CustomMethod_PreProcRelInstCreAttrs

It is possible to configure a set of Aras Innovator properties of the child to be available in `CustomMethod_PreProcRelInstCreAttrs`. Beside these Aras Innovator properties the CATIA Part Number of the child is also available (like Nomenclature).

Configuration

The list of required properties must be configured in the settings section of the Schema file. To set multiple properties you have to use “|” as delimiter:

```
<settings>
   <setting name="PreProcRelInstCreAttrsUseChildProps"
            value="item_number|classification"/>
</settings>
```

In your custom method you can access the properties like:

```
Item TargetItem = this.getPropertyItem("TargetItem");
string targetPN = TargetItem.getProperty("item_number", "");
string classification =
       TargetItem.getProperty("classification", "");
```

Creating and deleting Relations without updating the Source Item

It is possible to create and delete relations without applying “AML update” to the source item.

Configuration

The server setting “CreateDeleteRelsUpdSrcItem” has the default value ”false”. The old behavior can be changed back by setting the value to “true”.

Align CAD Structures with multiple Versions of the same CAD

By default, it is not possible to load a multilevel CAD Structure which holds different versions of the same CAD. Because it is not possible to load different versions of the same CATIA file (with equal file names) into CATIA at the same time.

This functionality aligns the structure by using the latest version of the CAD concerned in the structure for all occurrences.

Configuration

To enable the behavior, the server setting “AlignVersionsInCadStructure” must be set to “true”.

Deny Load if the same File is loaded in different Version

If a different version of a CATPart / CATProduct (from Aras Innovator) is already loaded in CATIA, the user is asked to close this file before loading the new structure from Aras Innovator. If the file was loaded from disk, the Aras Innovator meta data are applied to the already loaded file.

Note: CATParts loaded in visualization mode are not handled as loaded. This means, the geometry will be downloaded from Aras Innovator. If such a CATPart is loaded into design mode, the correct geometry is shown.

Automatic Save of PWB Session File

It is possible to save the current PDM information to a local session file. If the same user is logged in with an empty CATIA (no windows open), this session can be restored. It is possible to save the session automatically during every load from PDM and after update.

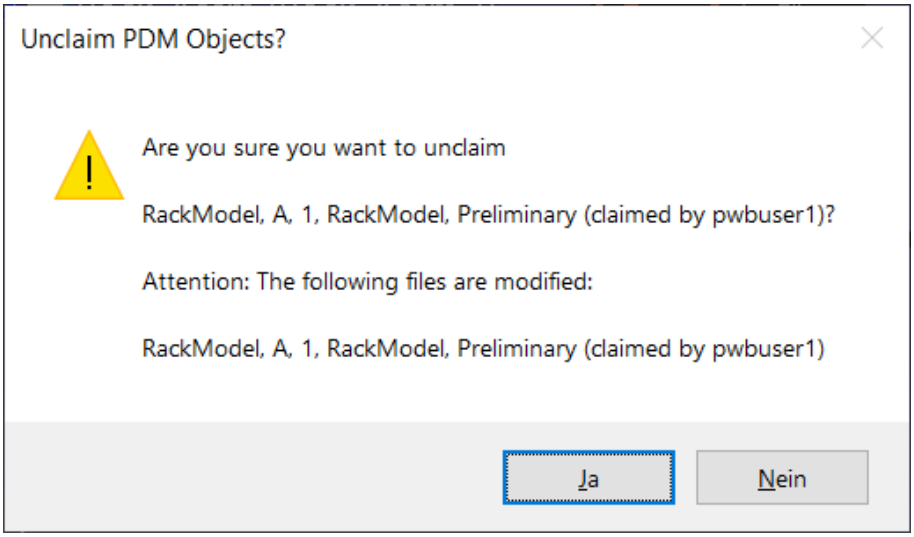
Configuration

To enable the functionality, you must set “QuickSaveLocalPwbSession” in the *settings* section of the Schema file on the CATIA client.

```
<settings>
   ...
   <setting name="QuickSaveLocalPwbSession" value="true" />
</settings>
```

Warning when the User wants to unclaim modified Files

There is a functionality which warns the user when he is about to unclaim CATIA files that are modified in the session:



Picture 207: Warning dialog at “Unclaim”

Configuration

This functionality is switched on by this Schema file setting:

```
<setting name="ModifiedFilesAtCheckInWarning" value="true" />
```

Compatibility with ACI

With ACI it is possible to export CATIA V5 data from 3DEXPERIENCE and import the data into Aras Innovator or to export CATIA V5 data from Aras Innovator and import the data into 3DEXPERIENCE. The imported data from 3DEXPERIENCE can be used with PDM Workbench. Data generated with PDM Workbench can be used in 3DEXPERIENCE.

If you want to continue working with both systems, you should always change a certain geometry in the same system and only use this geometry unchanged in the other system. This is because converting data always reduces the quality of the data.

To distinguish between original CATIA V5 data and CATIA V5 data from 3DEXPERIENCE PDM Workbench uses the `external_owner` property.

Data created by PDM Workbench uses
`external_owner = T-Systems.Mechanical.CATIA`

Data created by ACI uses
`external_owner = T-Systems.Mechanical.3DExperience`

By default a query in PDM Workbench will only find objects with `external_owner=T-Systems.mechanical.CATIA`. To find other objects, you must extend the *dataSource* for *ExternalOwner* in your PWB Schema.xml file:

```
<dataSource name="ExternalOwner" type="ValueList">
  <value
    name="T-Systems.Mechanical.CATIA,T-Systems.Mechanical.3DExperience"
    displayName=""/>
</dataSource>
```

With this change PDM Workbench query will also find objects with `external_owner=T-Systems.mechanical.3DExperience`.

By default PDM Workbench loads all CATParts / CATProducts / CATDrawings with a *external_owner* other than *T-Systems.mechanical.CATIA* in read only mode.

If the default behavior 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 setting “CustomMethod_GetCustomItemInfo”:

CustomMethod_GetCustomItemInfo	PwbCus_GetCustomItemInfo

Picture 208: Sample CustomMethod_GetCustomItemInfo configuration

Sample method:

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

// 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;
}

// handle additional external owners like T-Systems.Mechanical.3DExperience
if (String.IsNullOrEmpty(ItemObj.getProperty("source_id")) && // not a relation
    "Document" != ItemObj.getType() &&                        // not a designTable
    PwbServerApiObj.GetExternalOwnerFromCadClient()!=
         PwbServerApiObj.GetExternalOwner(ItemObj))        // wrong external_owner
{
   IsEditAllowed = false;
   IsClaimAllowed = false;
   ClaimAsNewGeneration = 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());

bool UpdateAllowed = !IsReleased;

if (!UpdateAllowed)
{
   ClaimAsNewGeneration = true;
}


bool NewGenHasAlreadyBeenCreated=(ItemObj.getProperty("pwb_claimed_is_new_gen")=="1");
bool CreateNewGenAtUpdate = ((!UpdateAllowed) || (!NewGenHasAlreadyBeenCreated));

// Returning the values.

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

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

}
```

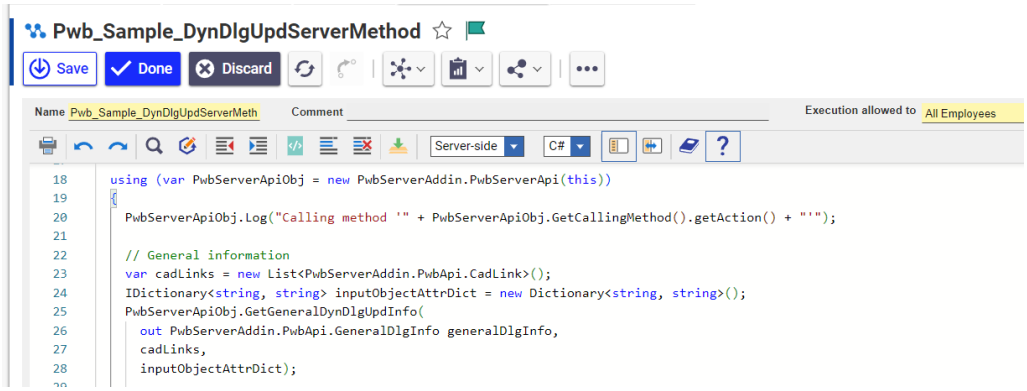
Dynamic Dialog Functionality

It is possible to have user dialogs which are not defined in a static way in the Schema file but defined dynamically in a custom Aras Innovator server method. This makes it possible to dynamically define at runtime both the dialog attribute definitions and their values.

Configuration

A custom C# server method which is called by the PDM Workbench and which returns the dialog information depending on the content of the call has to be defined.

A sample method for this, “Pwb_Sample_DynDlgUpdServerMethod”, exists.



Picture 209: Sample dynamic dialog custom server method

This method has to be defined in the *PWBSchemata*/*PWBSchema*/*dynDlgServerMethod* XML attribute in the Schema file, as shown in this example:

```
<PWBSchemata>

   <PWBSchema system="Aras" customization="Aras" serverConfig=""
              displayName="NLS_System" visibleLength="15"
              allowedLength="64" UseBomPartStructure="false"
              dynDlgServerMethod="Pwb_Sample_DynDlgUpdServerMethod" >

   <!—

      <PWBSchema system="Aras" customization="Aras" serverConfig=""
                 displayName="NLS_System" visibleLength="15"
                 allowedLength="64" UseBomPartStructure="false" >
   -->
```

The XML code in green shows the regular definition without the dynamic dialog functionality commented out.

The dialog definitions in the Schema file are the same as before, except they contain no attribute definitions, since the dialog content is requested from the Aras Innovator server dynamically:

```
<object name="/CAD/Mechanical/Part" displayName="NLS_CATPart"
        icon="CATPart" isDefaultFor="CATPart">

   ...

   <!-- Define custom context action(s) with dialog-->

   <!-- If using dynamic dialogs (dynDlgServerMethod defined) use empty
        dialog definition for custom action -->
```

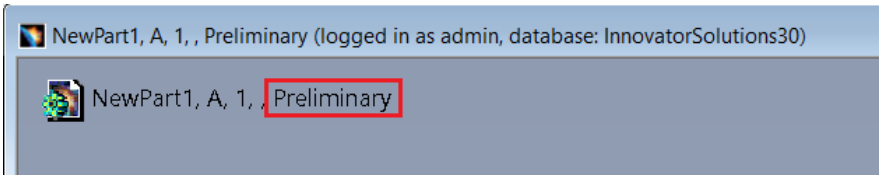
```
<customContextAction name="Pwb_Sample_ContextAction"
                     usedIn="PdmWindow|QueryDialog" confirm="false"
                     dialog="Pwb_Sample_ContextActionDynDlg" />
<customContextAction name="Pwb_Sample_DynContextPromote"
                     usedIn="PdmWindow|QueryDialog" confirm="false"
                     dialog="Pwb_Sample_ContextPromoteDlg" />


<!-- If not using dynamic dialogs use dialog definition using
     attributes -->
<!--
   <customContextAction name="Pwb_Sample_ContextAction"
                        usedIn="PdmWindow|QueryDialog" confirm="false"
                        dialog="Pwb_Sample_ContextActionFixedDlg" />
-->
...
<!-- Dialog definitions without any attributes: -->
<form name="Pwb_Sample_ContextActionDynDlg" />
<form name="Pwb_Sample_ContextPromoteDlg" />


<form name="Pwb_Sample_ContextActionFixedDlg">
   <formAttribute name="item_number" displayName="NLS_item_number"
                  widgetType="SingleLineEditor" mode="output" ... />
   ...
   <formAttribute name="classification" widgetType="SingleLineEditor"
                  mode="output" visibleLength="15" required="false" />
</form>
```

Usage

An example is shown with the custom context action method “Pwb_Sample_ContextPromoteDlg”. The starting point is a CAD Document in the “Preliminary” lifecycle state:



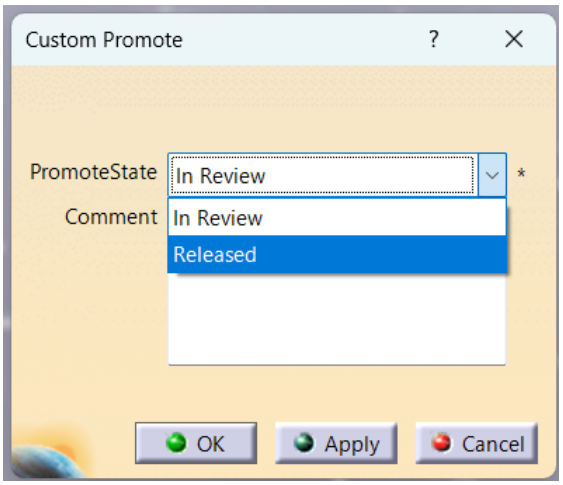
Picture 210: CAD Document in “Preliminary” state

The user clicks on the custom action “Custom Promote”:



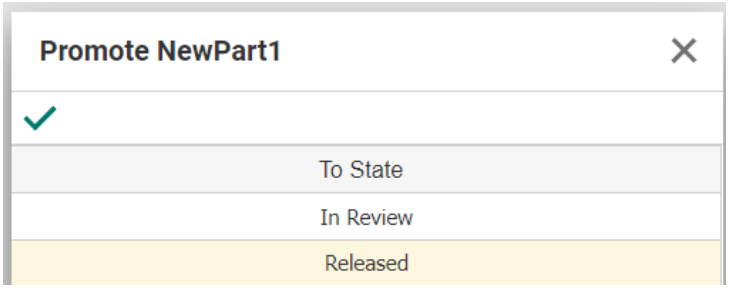
Picture 211: “Custom Promote” custom action

A dialog appears with the currently available target lifecycle states based on the current “Preliminary” lifecycle state of the CAD Document:



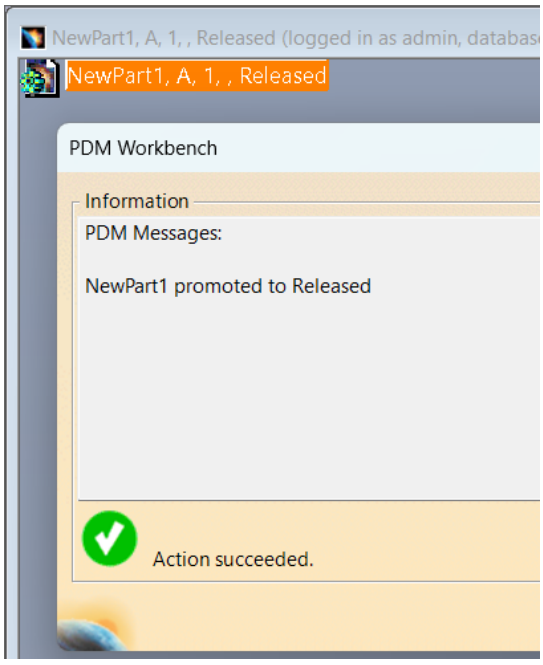
Picture 212: Lifecycle states in dialog based on “Preliminary”

These are the same states that are available in the Aras Innovator web client:



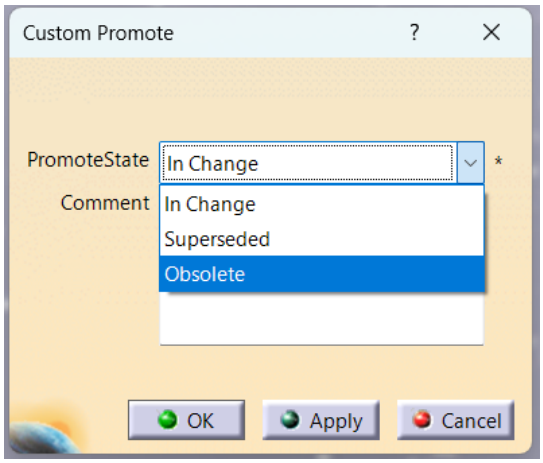
Picture 213: Lifecycle states in browser based on “Preliminary”

If the user clicks on “Custom Promote” the CAD Document is promoted to the selected lifecycle state:



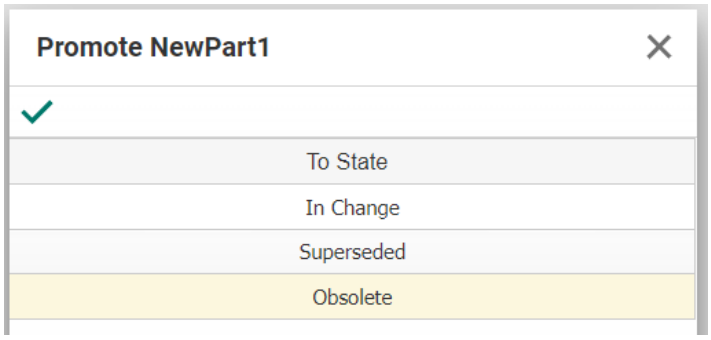
Picture 214: Result of “Custom Promote” action

Selecting “Custom Promote” again now shows different target lifecycle states based on the different current lifecycle state “Released”:



Picture 215: Lifecycle states in dialog based on “Released”

Again, this is the same as in the Aras Innovator web client.



Picture 216: Lifecycle states in browser based on “Released”

Activate 'DynContextPromote' functionality by default

The 'DynContextPromote' functionality shows a dialog to the user, where the user can select the new state of a CAD. The list is generated dynamically in Aras Innovator using the user and the CAD to determine the possible promote states.

Giving the path to root information for specific problematic PSN nodes to the user

If there are some inconsistent CATIA structures, like multiple instances with the same instance name or instances without an instance name in CATIA, it is not possible to update such a (sub-)structure to Aras Innovator. (The only known way such structures occur in Aras Innovator is for a migrated data from a legacy system). For such structures the user gets a message which holds the complete path to root to the problematic CATProduct. This allows the user to repair the structure before doing the update.

CHAPTER 8

Client Schema File Configuration

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

Structure of the Schema File

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

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

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

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

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

   <!-- customization of Aras -->
   <PWBSchema system="Aras" customization="PDM-Customization"
              serverConfig="" 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*.

- “serverConfig” This attribute is optional. If it exists and its value is not an empty string, then it defines the name of the PWB Configuration item to be used on the server. This capability can be used to provide multiple PWB Configuration items on the server and let the client define which one to use.
  If the attribute *serverConfig* is not set or its value is an empty string, then the relevant name of the PWB Configuration item to be used on the server is defined by the Aras Innovator variable “PwbConfigurationItemName”.
- “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 CATIA V5.
- “allowedLength” Contains the allowed length of the values inserted in the text editor widgets in characters.

NLS Support for Display Names

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

In this case, the value for the frame's display name “NLS_UserData” is defined in the file “PWBSchemaDisplayNames_Aras_Aras.CATNls”:

In the Schema file:

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

In the file “PWBSchemaDisplayNames_Aras_Aras.CATNls”:

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

Configuration Settings

The configuration of the PDM Workbench 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 33

soapTargetUrl	optional	see “SOAP target URL” on page 33
externalSoapClientCallPathEnvVar	optional	see “SOAP Client Call Path Environment Variable” on page 34
externalFileClientCallPathEnvVar	mandatory	see “File Client Call Path Environment Variable” on page 34
addTempPrefix	optional	see “AddTemp PrefixAdd Temp Configuration” on page 133
catiaWindow	optional	see “CATIA V5 Window Configuration” on page 75
pwbWindow	optional	see “PWB Window Color” on page 82
pwbCompareStructureWindow	optional	see “Comparing PDM Structure Trees” on page 83
maxExpansionLevel	optional	see “Maximum Expansion Level” on page 34
colorSupersededNodesAsOutdated	optional	see “Mark superseded Nodes” on page 40
setReadOnly	optional	see “Read only” on page 34
neverReloadCatiaDocAfterUpdate	optional	see “Reload after Update” on page 34
checkForCadOwner	optional	see “Check for CAD Owner” on page 155
checkFileVersionsInStructure	optional	see “Check File Versions in Structure” on page 155
checkCatPartsInVisuMode	optional	see “Check CATParts in Visualization Mode” on page 155
checkCatiaPartNumbersBeforeUpdate	optional	see “Check CATIA Part Numbers before Update” on page 155
allowLoadingDifferentFileVersion	optional	see “Allow loading different File Versions” on page 35
downloadDrawingRelated3DFile	optional	see “Download related 3D File” on page 144
showOutOfDateLinkInfo	optional	see “Show out-of-date Link Info” on page 35
startUpdateAfterCreateNewCatiaFile	optional	see “Update after Create new CATIA file” on page 35

uploadChangedFilesAtSyncFailure	optional	see “Upload changed Files at Sync Failure” on page 35
checkAuthoringToolVersion	optional	see “Check for CAD Document CATIA Release at PDM Update” on page 156
createThumbnailwithWindows	optional	see “Create Thumbnails by CATIA” on page 107
allowBrokenLinkAtNewProduct	optional	see “Optional Check for broken Links at Update” on page 156
displayCustomQueryButtons	optional	see “Display Custom Query Buttons” on page 49
requiredDialogAttributes	optional	see “Required Attributes” on page 36
queryDialogIsModal	optional	see “Modal Dialog” on page 49
checkIfNoQueryAttrsFilled	optional	see “Check if no Query Attribute filled” on page 49
showDisplayStringInQueryListView	optional	see “Display String in Query List View” on page 49
maximumDescriptionAttributeLength	optional	see “Maximum Description Attribute Length” on page 36
showDataSourceNlsValues	optional	see “Show Data Source NLS Values” on page 36
showCreateVersionAtUpdate	optional	see ““Create new file versions at update?” Button” on page 56
defaultCreateVersionAtUpdateValue	optional	see “Default create Version Value” on page 56
queryDialogDimensions	optional	see “Configuring the Size of the “PDM Query” Dialog” on page 48
duplicateStructureDialog	optional	see “Duplicate Structure Configuration” on page 70 and 71
duplicateStructure	optional	see “Context Action “Duplicate Structure”” on page 69
duplicateStructureRequest	optional	see “Duplicate Structure Configuration” on page 71
showReconnectAtUpdate	optional	see “Reconnect at Update” on page 168
showUnlockAfterUpdate	optional	see ““Unclaim after Save” on page 57
showCreatePartsAtUpdate	optional	see “Show create Parts” on page 57

defaultCreatePartsAtUpdateValue	optional	see “Default create Parts Value” on page 58
showCreateDialogsMultiColumn	optional	See “Show “PDM Create” Dialogs with Multi Column” on page 55
renameExistingExMapFilesAtLoad	optional	see “Rename existing Files in Exchange Map” on page 37
duplicateRelatedDrawings	optional	
showLoadWithLinkInQuery	optional	see “Automatically loading CATDrawings or linked CATParts” on page 50
showLoadWithDrawingInQuery	optional	see “Automatically loading CATDrawings or linked CATParts” on page 50
removeToolbarIcons	optional	see “Toolbar Configuration” on page 41
filter3dRepTypes	optional	see “Additional Rep Types” on page 146
catiaEnvironment	optional	see “Customer-Specific Environment” on page 41
autoLoginOnSessionExpired	optional	see “Auto Login on Session expired” on page 37
sessionSettings	optional	see “Session Settings” on page 37
installedLanguages	mandatory	see “Language Settings” on page 39
dateFormat	mandatory	see “Date Format” on page 39
lastModificationDateAttribute	mandatory	see “Last Modification Date Attribute” on page 40
standardPartUserDefPropAttribute	optional	see “Standard Part Functionality for BOM Part Structure Data Model” on page 95
partClasses	optional	see “Part Classes” on page 40
standardPartFiles	optional	see “Standard Part Functionality” on page 95
boundingBoxAttributes	optional	see “Bounding Box Definition” on page 139
catiaNodeBehaviorDefinitions	optional	see “Configurable CATIA Components Support” on page 164

templateFiles	optional	see “Management of CATIA Templates” on page 101
updateCatiaUserDefinedProperties	optional	see “CATIA User Defined Attributes” on page 130
updateCatiaLinksInPdm	optional	see “Link Support” on page 142 and “Basic Multi-Model Link Support” on page 145
attrsModifiableAtRevise	optional	see “Modifiable Attributes” on page 67
expandStructureForRevise	optional	see “Expand Structure” on page 68

“object”: 1 - n

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

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

“relation”: 1 - n

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

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

“attribute”: 0 - n

The definition of PDM attributes that are referenced in dialogs.

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

“pwbAttribute”: 0 - n

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

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

“dataSource”: 0 - n

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

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

PDM Attributes and Form Attributes

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

The *attribute* definition contains the following attributes:

- “name” Mandatory, must correspond to the PDM attribute's name.
- “displayName” Mandatory. As described in “NLS Support for Display Names” the NLS string for the “displayName” XML attribute is defined in the CATNls file specific to the PDM system and the customization.
- “dataSource” Optional. The data source includes the possible values for this attribute.
- “isFileName” Optional. If is set to “true” the value of the corresponding input file name is checked about illegal[2] characters when creating a file.
- “isPartNumber” Optional.
- “autoName” Optional.
- “isDerived” Optional.

Example:

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

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

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

A form definition contains form attributes which reference the previously defined PDM attribute.
The *form* attribute definitions contain the following attributes:

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

  Default is “output”.
- “visibleLength” Optional, the length of the text editor widget in characters.
- “allowedLength” Optional, the length of 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”.

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

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

  This attribute should only be added to “Query” forms. Please refer to the *PDM Workbench User Manual* for more information.

  Default is “false”.
- “displayOnly” Possible values are “true” or “false”. If “true”, then the display value of the value of the data source will be used.
- “entryAllowed” Possible values are “true” or “false”. If “true”, then the user can enter a text additional to the attached data source.

Example:

```
<form name="Query">

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

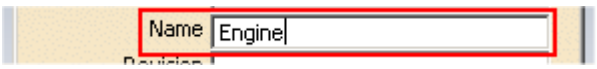
Form definitions generally refer to classes of PDM objects (query form, properties form, etc.). The definition of PDM object classes is described in chapter **PDM Objects**.

Description of the Widget Types

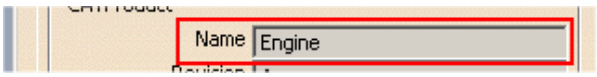
There are nine different widget types available to build up dialogs with. All widget types except “SingleLineEditor”, “MultiLineEditor” and one mode of “NameValueList” can only be used on attributes that have certain kinds of Data Sources attached. Data Sources are a container of a limited set of values.

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

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

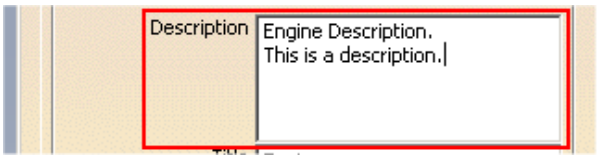


Picture 217: Single Line Editor Widget, update mode



Picture 218: Single Line Editor Widget, output mode

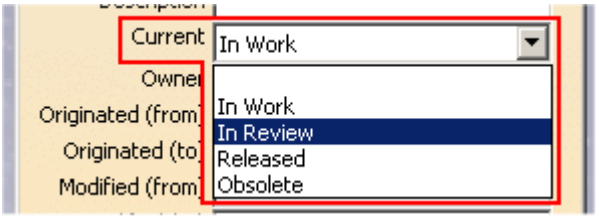
MultiLineEditor Supports “update” and “output” mode.
Can be used for attributes with no data source attached and also for attributes with data sources of type “ValueList”.



Picture 219: 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 220: Combo Box Widget, select mode

SingleCheckBox

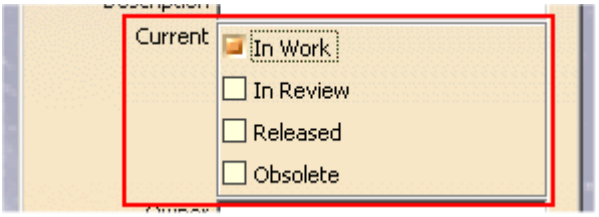
Supports “select” and “output” mode.
Needs an attribute with a data source of type “BooleanValueList”.
This widget should be used only for required attributes or for attributes that are only displayed, already set to a value and cannot be updated.



Picture 221: Single Check Box Widget, select mode

CheckBoxes

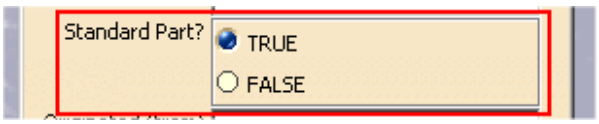
Supports “select” and “output” mode.
Possible for attributes with data sources which contain several values (type “ValueList”, “BooleanValueList” or “invokeMessage”) and where the user can select more than one value.



Picture 222: Check Boxes Widget, select mode

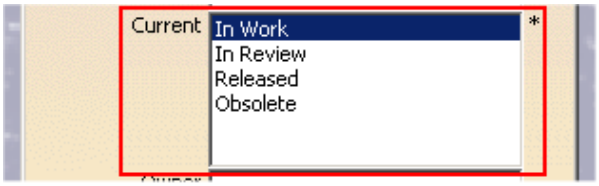
RadioButtons

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



Picture 223: Radio Buttons Widget, select mode

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



Picture 224: Single Selector List Widget, select mode

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

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

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

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

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

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

Mandatory.

PWB Workbench toolbar - embed login / logout buttons in a fly-out toolbar.

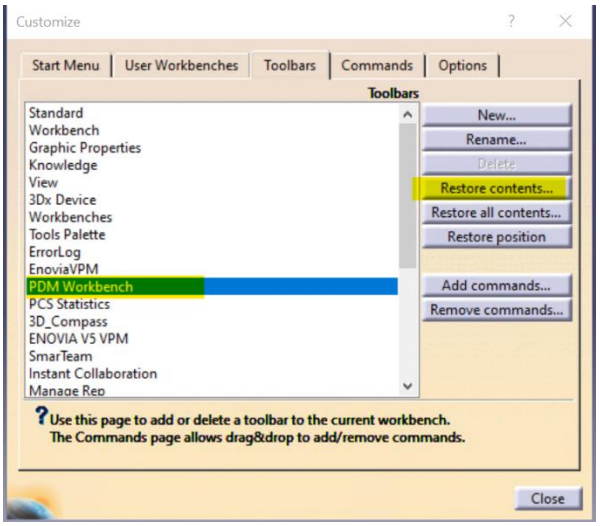
It is possible to use a fly-out toolbar icon for the PWB login / logout command.

To activate the fly-out command activate the following setting in the PWBSchema.xml file:

```
<setting name="UseFlyoutLogoutCmd" value="true" /> <!-- Default
value is "false" -->
```

If the user does not see the flyout logout command because he has customized the PWB toolbar, it is possible to use the command:

Tools->Customize->Toolbars->Restore contents for "PDM Workbench"



Picture 225: Restore contents for "PDM Workbench"

PDM Objects

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

An *object* XML tag contains the following attributes:

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

Example:

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

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

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

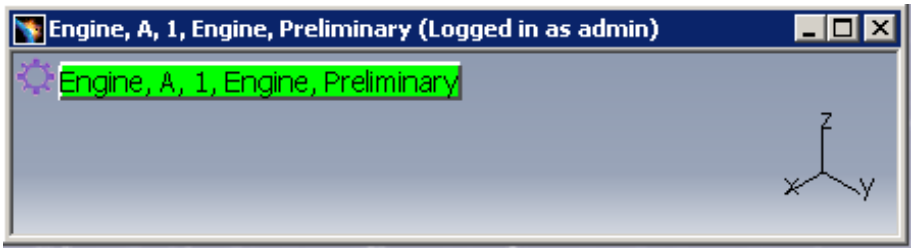
As described in “NLS Support for Display Names” the NLS string for the XML attribute *displayName* is defined in the CATNls file specific to the PDM system and the customization.

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>
```



Picture 226: PDM Node in PWB window

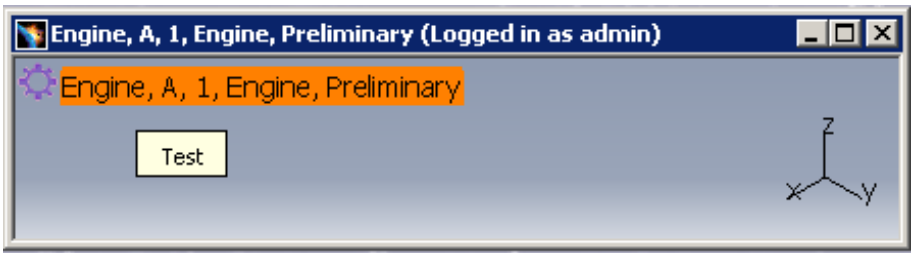
Tooltip of PDM Objects

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

Example:

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

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



Picture 227: Tooltip of PDM Node in PWB window

Actions on PDM Objects

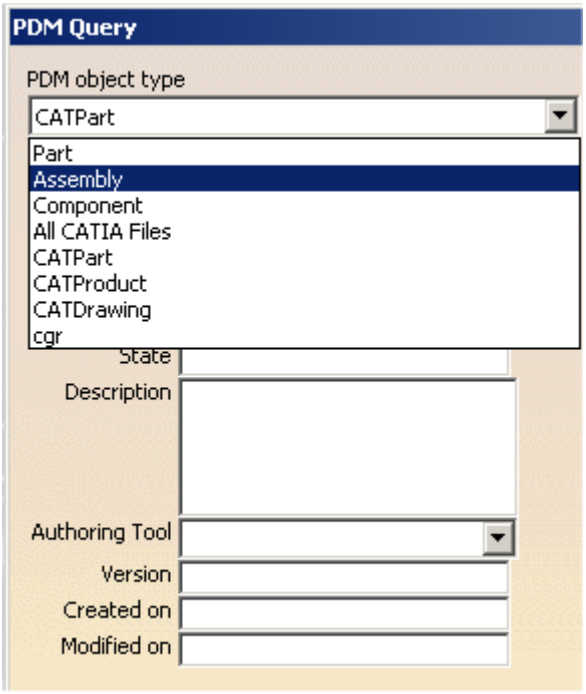
Actions that can be performed with PDM objects have to be defined in the Schema file. There are two kinds of actions: So-called toolbar actions, which are started by clicking on an icon in the PDM Workbench toolbar, and context actions, which are started by right-clicking on the node and selecting one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

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

Example:

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

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



Picture 228: Select PDM object type in “PDM Query” dialog

Context Actions

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

Example:

```
<!-- * all PWB context actions permitted for this object * -->

<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog" />

<contextAction name="CheckOut" usedIn="PdmWindow|QueryDialog" />

<contextAction name="CheckInAll" usedIn="PdmWindow|QueryDialog"
               removeFromCatiaWindow=="true" />

<contextAction name="CheckOutAll" usedIn="PdmWindow|QueryDialog"
               removeFromCatiaWindow=="true" />

<contextAction name="Highlight" usedIn="PdmWindow" />

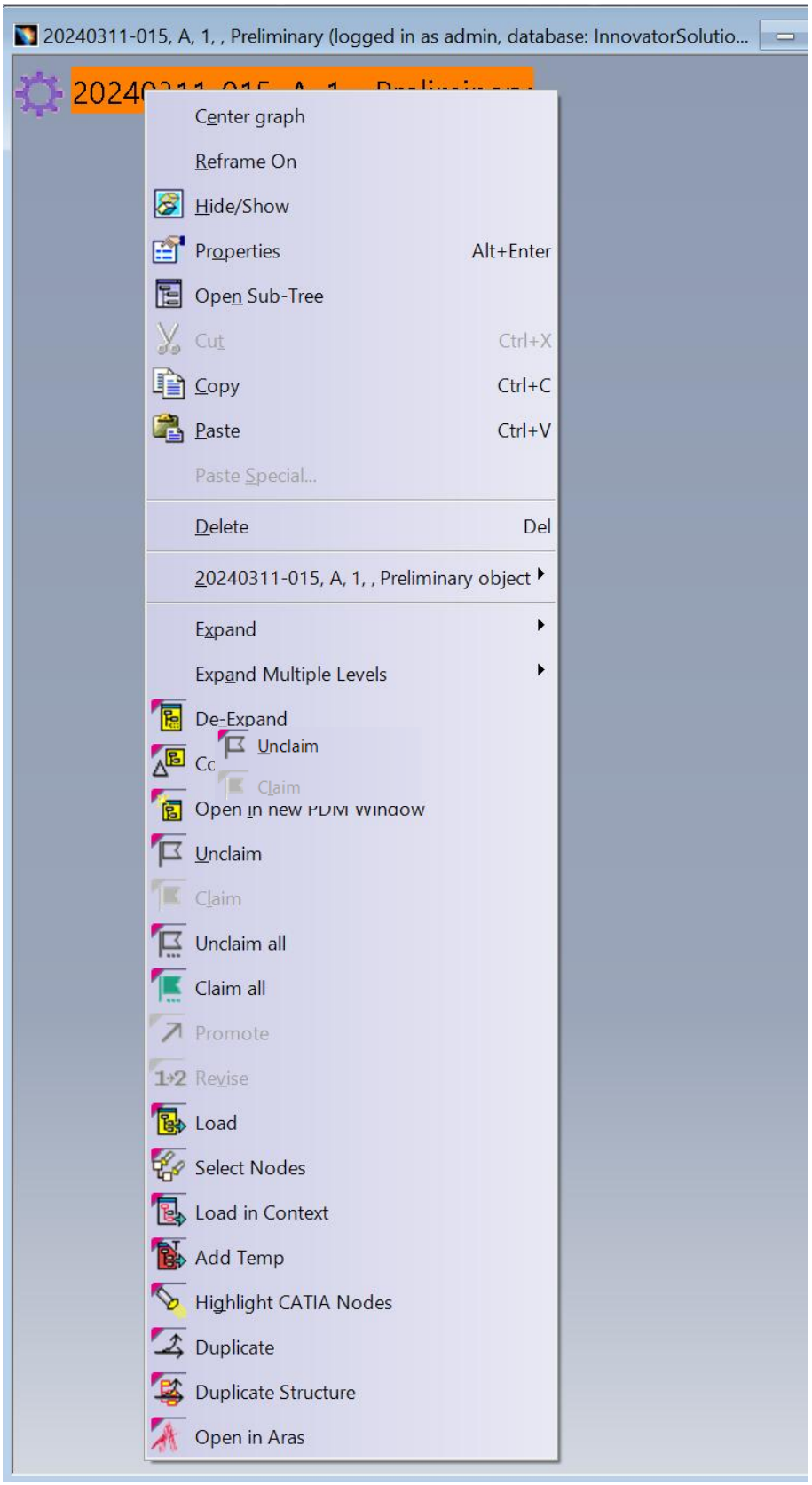
<contextAction name="Insert" removeFromCatiaWindow="true" />

<contextAction name="Replace" removeFromCatiaWindow="true" />
```

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

The attribute removeFromCatiaWindow is used to control the appearance of the context actions in the PSN tree. If set to false the action is removed from the context menu of the PSN tree.

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



Picture 229: Context actions for the type /Part/Assembly - Example

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

If the context action “Unclaim” (CheckIn) is defined, then the dialog forms “CheckInNew” and “CheckIn” must also be defined.

If the context action “Claim” (CheckOut) is defined, then the dialog form “CheckOut” must also be defined.

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

The following context actions are available:

Context action name	Description
Properties	see “Context Action “Properties”” on page 157
CheckIn	see “Context Action “Unclaim”” on page 157
CheckOut	see “Context Action “Claim”” on page 157
CheckInAll	see “Context Action “Unclaim all”” on page 157
CheckOutAll	see “Context Action “Claim all” on page 157
Duplicate	see “Context Action “Duplicate”” on page 158
DuplicateStructure	see “Duplicate Structure Configuration” on page 69
DuplicateDelete	see “Context Action “Duplicate-Delete”” on page 158
DuplicateSupersede	see “Context Action “Duplicate-Supersede”” on page 158
Insert	see “Context Action “Insert PDM Node”” on page 158
Replace	see “Context Action “Replace Node”” on page 158
Expand	see “Context Action “Expand”” on page 85
MultipleExpand	see “Context Action “Expand Multiple Levels”” on page 85
CustomExpand	see “Context Action “Custom Expand”” on page 85
DeExpand	see “Context Action “De-Expand”” on page 86
MultipleExpandWithBomChildren	see “Context Action “Expand Multiple Levels including Bom Children”” on page 86
CompareStructure	see “Comparing PDM Structure Trees” on page 83
Promote	see “Context Action “Promote”” on page 93
Revise	see “Context Action “Revise”” on page 67
Execute	see “Context Action “Execute”” on page 159
UpdateSubCmpRels	see “Context Action “Update structure relations”” on page 159
UpdateParentCmpRel	see “Context Action “Update parent relation”” on page 159

LoadStructure	see “Context Action “Load”” on page 137
LoadStructureCurrent	see “Context Action “Load (Current)”” on page 138
LoadInContext	see “Context Action “Load in Context”” on page 138
AddTemp	see “Context Action “Add Temp”” on page 133
OpenInNewPdmWindow	see “Context Action “Open in new PDM Window”” on page 138
Open	see “Context Action “Open File”” on page 138
OpenWithLinks	see “Context Action “Open File with Link”” on page 138
TempOpen	see “Context Action “Open File temporarily”” on page 138
OpenDrawingWithRelated3D	see “CATDrawing: Loading referenced Data as “Current”” on page 139
OpenDrawingWithRelated3DCurrent	see “CATDrawing: Loading referenced Data as “Current”” on page 139
OpenRelatedDrawings	see “Context Action “Open related drawings”” on page 139
OpenCatalog	see “CATIA Catalogs” on page 115
OpenCatalogForEdit	see “CATIA Catalogs” on page 115
NewestVersionInfo	see “Context Action “Newest Generation Info”” on page 159
UpdateItem	see “Context Action “Update Item”” on page 159
ContextCreate	see “Context Action “PDM Create in Context”” on page 159
Highlight	see “Context Action “Highlight”” on page 159
CopyAttributes	see “Context Action “Copy element attributes”” on page 160
DeleteItem	see “Context Action “Delete”” on page 160
DeleteRelation	see “Context Action “Delete relation”” on page 83
CreateNewVersion	see “Context Action “Create new generation”” on page 134
DeleteNewestVersion	see “Context Action “Delete newest generation”” on page 134
UnlinkAndDeleteNewestVersion	see “Context Action “Unlink and delete newest generation”” on page 134
RelateToPart	see “Support for Relating a new CATIA File to an existing Part” on page 165
ShowNeighbor	see “Context Action “Show Neighborhood”” on page 139

SyncStrucFileToPart	see “Context Action “Synchronize to BOM”” on page 160
SelectNodes	see “Selecting Nodes in the PDM Structure Window” on page 84
CheckCadLinks	see “Context Action “Check CAD Links”” on page 157
SetConfigInfoOnRelations	see “Setting Configuration Information on Structure Relations” on page 171

Disabling Context Menu Items in the CATIA Structure Window

In the corresponding “contextAction” definition the XML attribute “removeFromCatiaWindow” needs to be set to “true”:

```
<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog"
               removeFromCatiaWindow="true" />
```

If that is done the corresponding context action will not be available in the CATIA window anymore. It may still be available as a context menu in the PDM structure window or as a context menu in the “PDM Query” dialog result list.

PDM Object Forms

The following forms can be defined for an object class:

“Query”, “Properties”, “UpdateItem”, “CheckInNew”, “CheckIn”, “CheckOut”

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

Example:

```
<form name="Query">

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

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

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

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

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

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

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

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

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

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

PDM Relations

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

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

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

Example:

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

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

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

Description of PDM Relations

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

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

Example:

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

“Relationship” Tags

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

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

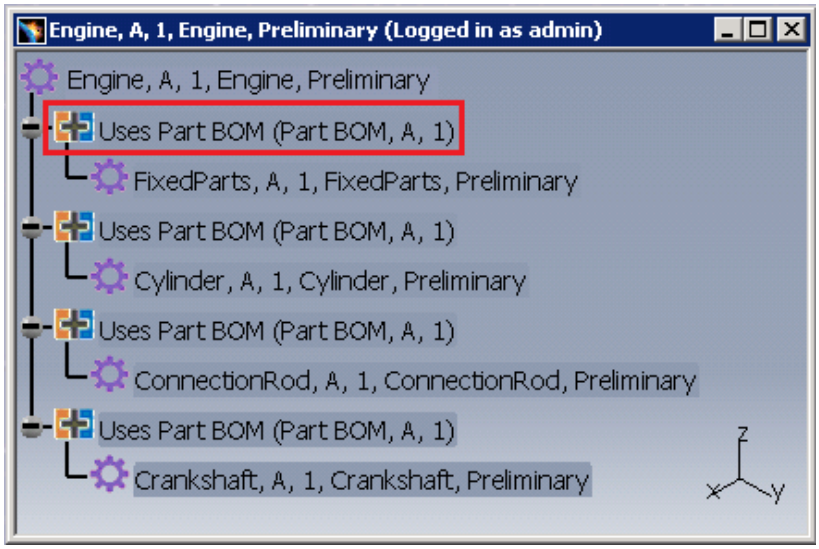
Example:

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

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

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

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



Picture 230: Relation icon with relationship and description attribute

The display of a relationship name in the PDM tree can be switched off by defining the XML attribute “showNameInPdmTree” with the value “false” for a relationship definition:

```
<relation name="/CAD Structure/Structure"
          displayName="NLS_CADStructure_Structure"
          icon="Aras_Relation"
          createAllowed="true" expandAllowed="true" info="AsSaved">

   ...

   <leftToRightRelationship
             name="/CAD Structure/Structure_LeftToRight"
             displayName="NLS_CADStructure_Structure_LeftToRight"
             multipleExpand="true"
             showNameInPdmTree="false" />

   <rightToLeftRelationship
             name="/CAD Structure/Structure_RightToLeft"
             displayName="NLS_CADStructure_Structure_RightToLeft"
             multipleExpand="true" />
```

"Left and Right Object" Classes

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

In the following example the relation is between CATIA parts:

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

PDM Relation Forms

The following forms can be defined for a relation class:

"Properties", "UpdateItem", "Create"

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

Example:

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

Data Sources

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

Data Source "Value" tag

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

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

Static data sources can be of type:

SingleValue: the data source contains only one static element.

Example:

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

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

Example:

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

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.

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 “PDM 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>
```

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

Example:

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

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

Example:

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

Complete Example of using a Data Source Tag

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

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

CHAPTER 9

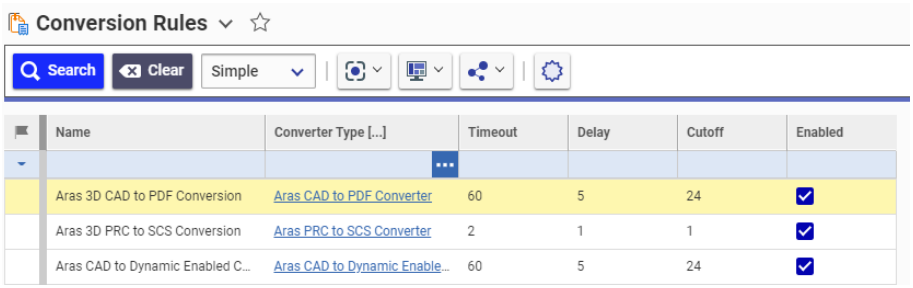
Troubleshooting

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

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

This issue has been observed in Aras Innovator 12 SP14 and higher.

With Dynamic Viewer or Streaming Viewer enabled, the conversion service adds CAD Instances to the CAD Structure. Additional CAD instances will compromise the CAD Structure.



Picture 231: Dynamic CAD Converter enabled

To disable the creation of CAD Instances by the Conversion service modify the method *CR_3DCADtoDyn_SetFiles* or *CR_3DCADtoPDF_SetFiles* for older version of Aras Innovator.

In the method "*CR_3DCADtoDyn_SetFiles*" add "`bool ExternalOwnerTSI(List<String> depIds);`" to internal interface IDataAccessLayer:

```
internal interface IDataAccessLayer
  {
    Item SetDynamicAssemblyEnabled(List<String> depIds, bool
dynamicAssemblyEnabled);
    Item FillBoundingBoxes(List<String> depIds, string
fileShatteredModel);
    Item CreateCadInstances(List<String> depIds, string
fileShatteredModel);
    Item GetSuccessResult();
    Item AddSczFileRepresentation(XmlNode nativeFileIdNode, string
fileScz, string kind);
    Item AddCadDataLoadingConfiguration(List<String> depIds);
    Item GetFile(string fileId);
    bool ExternalOwnerTSI(List<String> depIds);
}
```

Add following function after the closing brackets of the "`public Item CreateCadInstances(List<String> depIds, string fileShatteredModel)`" function:

```
public bool ExternalOwnerTSI(List<String> depIds)
{
    if(depIds.Count > 0)
    {
        string cadId = depIds.ToArray()[0];
        Item cadItem = _inn.newItem("CAD", "get");
        cadItem.setID(cadId);
        cadItem.setAttribute("select", "external_owner");
        cadItem = cadItem.apply();

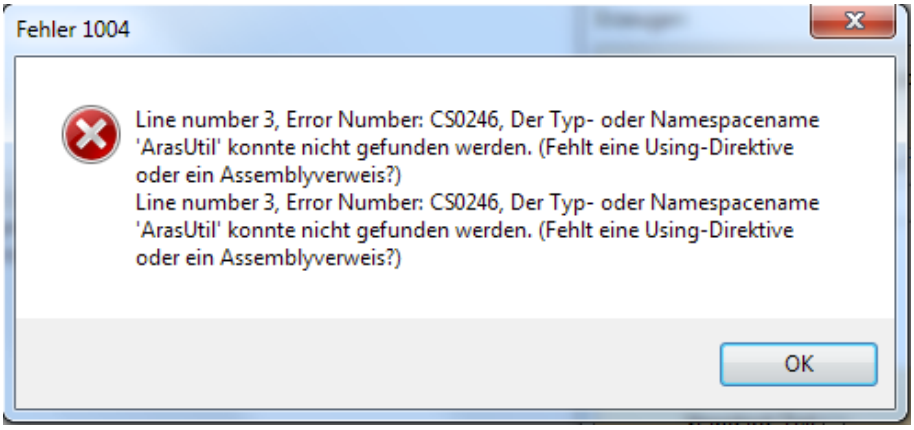
        if (cadItem.isError())
        {
          return false;
        }

        string external_owner = cadItem.getProperty("external_owner");
        if (external_owner.StartsWith("T-Systems"))
        {
          return true;
        }
    }
    return false;
}
```

Modify the following function:

```
if (dynamicAssemblyEnabled || streamingAssemblyEnabled)
{
    bool externalOwnerTSI = _dataAccessLayer.ExternalOwnerTSI(depIds);
            if (!externalOwnerTSI)
            {
                methodResult = _dataAccessLayer.CreateCadInstances(depIds,
_files["shatteredModel"]);
                if (methodResult.isError())
                {
                    return methodResult;
                }
            }
}
```

Type or Namespace “ArasUtil” could not be found …



Picture 232: Name space could not be found

Please replace calls of the class “ArasUtil” with calls to “PwbServerAddin.PwbServerApi”.

The usage of the class “ArasUtil” in custom server methods is not allowed anymore and must be replaced with calls to “PwbServerAddin.PwbServerApi”.

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

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

```
PdmObject.PerformAction (Lock)

ArasUtil.AddOwner -> LockStatus:'0'

ArasUtil.AddOwner -> caught exception:'An item with the same key has
already been added.'

   at System.Collections.Generic.Dictionary`2.Insert(TKey key, TValue
value, Boolean add)

   at
System.Linq.Enumerable.ToDictionary[TSource,TKey,TElement](IEnumerable`1
source, Func`2 keySelector, Func`2 elementSelector, IEqualityComparer`1
comparer)

   at Aras.Server.Core.FileContainerItemsOnAfterUpdate.Main(XmlDocument
inDom, XmlDocument outDom, Object eventData) in
E:\Builds\Innovator_RELS11-0-SP11\6812-RELS11-0-
SP11\Innovator.git\CompilableCode\Core\InternalMethods\Files\FileContainer
ItemsOnAfterUpdate.cs:line 67
```

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

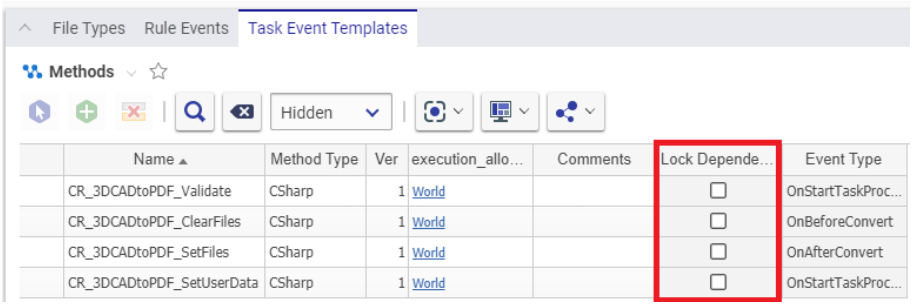
Conversion Tasks fail for updated or new CAD Files from PDM Workbench

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



Picture 233: “Aras 3D CAD to PDF Conversion” Rule

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



Picture 234: Uncheck "Lock Dependencies Aras 3D CAD to PDF Conversion" Rule

CATIA crashes when loading large assemblies with very old CATIA files

Comment out the line in win_b64/code/dictionary/CATUExitIO.dico by adding a # to the start of the line:

```
CATUExitIO          CATIUExitIO          libPWBModelLib
```

→

```
# CATUExitIO          CATIUExitIO          libPWBModelLib
```

If this line is commented out, the released cache functionality can only be used in a case, that you download both, CGR and native geometry, the later automatic fetch of the native files is not possible if this line is commented out.
This problem should be solved in V5-6R2023 SP7 (BR10000400355).

Before open: Load the Assembly Workbench by creating a new CATProduct (or comment out the CATIA environment variable "CATNoStartDocument=1" that with our default settings will lead to a new CATProduct is opened in the AssembyWorkbench on CATIA start). The Product can then be closed before opening, so most probably avoiding an interference with loading DLLs or CATSettings by pre'loading instead of during the open.

Comment out the lines in the environment file that relates to Q-Checker.

Don't start a CATIA macro when starting CATIA (it then runs in some different mode).