

# SIEMENS

## COMOS

### Process 3D Integration Administration

#### Operating Manual

Trademarks	1
Plant Modeler	2
COMOS PDMS Integration	3
COMOS 3D viewing	4
COMOS NX - Routing Mechanical interface	5
References	6

## Legal information

### Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

<b>⚠ DANGER</b>
indicates that death or severe personal injury <b>will</b> result if proper precautions are not taken.
<b>⚠ WARNING</b>
indicates that death or severe personal injury <b>may</b> result if proper precautions are not taken.
<b>⚠ CAUTION</b>
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.
<b>CAUTION</b>
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.
<b>NOTICE</b>
indicates that an unintended result or situation can occur if the relevant information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

### Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

### Proper use of Siemens products

Note the following:

<b>⚠ WARNING</b>
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

### Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

### Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

# Table of contents

<b>1</b>	<b>Trademarks .....</b>	<b>9</b>
<b>2</b>	<b>Plant Modeler .....</b>	<b>11</b>
2.1	Default settings in COMOS.....	11
2.1.1	Project properties: Setting the default location for pipes and equipment .....	11
2.1.2	Defining the default location for pipes.....	12
2.1.3	Defining the default location for equipment .....	13
2.1.4	Assigning references .....	14
2.1.5	Deleting references .....	15
2.1.6	Configuring the document management settings .....	15
2.1.7	Configuring the MicroStation settings .....	16
2.1.8	Navigating to references .....	17
2.1.9	ABO objects .....	18
2.1.9.1	Settings for ABO objects.....	18
2.1.9.2	Setting a start node for ABO objects .....	19
2.1.9.3	Removing ABO objects from the start node .....	20
2.1.9.4	Clearing ABO start points .....	20
2.1.10	The "@J Project > @G Project settings, general" node in the base data .....	20
2.1.11	P&ID data.....	21
2.1.12	Linking between P&ID and PPC .....	22
2.1.13	Cell libraries in COMOS.....	22
2.1.13.1	Creating a cell library in COMOS.....	22
2.1.13.2	Importing a cell library into COMOS .....	23
2.1.13.3	Editing a cell library stored in COMOS .....	23
2.2	Installation.....	24
2.2.1	Installing Plant Modeler.....	24
2.2.2	Updating Plant Modeler .....	25
2.3	Configuration.....	26
2.3.1	Settings in the COMOS configuration file for Plant Modeler.....	26
2.3.2	Making subsequent changes to the configuration file.....	29
2.3.3	Status queries .....	29
2.3.3.1	Prerequisites for the 3D status query .....	29
2.3.3.2	Editing status queries in the configuration file in COMOS.....	31
2.3.4	Saving the ABO start node in the configuration file .....	32
2.3.5	Default directories and settings in the configuration file .....	32
2.3.6	Settings for the "HVAC" add-in in the configuration files .....	35
2.3.7	Communication mode.....	35
2.3.7.1	Setting communication mode.....	35
2.3.7.2	Keys for communication mode.....	36
2.4	Attribute Transfer Editor.....	38
2.4.1	Introduction .....	38
2.4.2	Making project settings .....	39
2.4.3	Create a template .....	39
2.4.4	Assign template .....	40

2.5	Default settings in Plant Modeler .....	41
2.5.1	Configuring settings in Plant Modeler .....	41
2.5.2	Plant Modeler KeyIns .....	41
2.5.2.1	Available Plant Modeler KeyIns .....	41
2.5.2.2	Defining AccuDraw shortcuts for Plant Modeler KeyIns .....	42
2.5.3	MicroStation handle .....	42
2.5.4	The "COMOSPlantModelerDemo.cel" cell library .....	43
<b>3</b>	<b>COMOS PDMS Integration .....</b>	<b>45</b>
3.1	Activating COMOS PDMS Integration for COMOS .....	45
3.2	Cats&Specs .....	46
3.2.1	General information regarding catalogs (Cats) .....	46
3.2.2	Pipe parts from the pipe part catalog (PPC objects).....	47
3.2.2.1	General.....	47
3.2.2.2	"External 3D Interface" tab.....	47
3.2.2.3	Assigning an AVEVA PDMS template .....	47
3.2.2.4	Import template .....	50
3.2.2.5	Mapping AVEVA PDMS attributes to COMOS .....	52
3.2.2.6	Field "Mask for PCOM name generation".....	57
3.2.2.7	Field "Material" .....	57
3.2.2.8	Automatically adding additional pipe parts to the PipeSpec.....	57
3.2.2.9	Export settings .....	58
3.2.2.10	Bolts .....	59
3.2.2.11	Note "Not checked" .....	60
3.2.2.12	Answers for selectors.....	60
3.2.2.13	"VDM Data sheet" tab .....	61
3.2.3	PipeSpec elements .....	62
3.2.3.1	General.....	62
3.2.3.2	"External 3D Interface" tab.....	62
3.2.3.3	"PDMS PipeSpec elements" tab .....	62
3.2.4	COCO tables.....	65
3.2.4.1	General.....	65
3.2.4.2	Mapping via the standard table.....	65
3.2.5	Naming conventions.....	67
3.2.5.1	General.....	67
3.2.5.2	Formula for name generation.....	68
3.2.5.3	Window for name generation .....	69
3.2.5.4	Name generation for Cate.....	70
3.2.5.5	Name generation for bolting sets (BTSE) .....	72
3.3	Using COMOS PDMS Integration.....	72
3.3.1	Installation .....	72
3.3.1.1	Local installation.....	73
3.3.1.2	Citrix Server .....	74
3.3.1.3	Citrix Client.....	74
3.3.2	Preparing the COMOS database .....	76
3.3.3	Project properties .....	76
3.3.4	UDAs .....	77
3.3.5	Creating an INI file .....	78
3.4	Configuration basics for interface operations.....	78
3.4.1	Interface objects.....	78
3.4.2	Classes and subclasses.....	79

3.4.3	Class definition objects .....	80
3.4.4	Subclass definition objects of the "Pipe" and "TaggedItem" classes .....	81
3.4.5	Subclass definition objects of the "Query" class .....	82
3.4.6	Subclass definition objects of the "Document" class .....	83
3.4.7	"@PDMSMAP" folder .....	84
3.4.8	Name mapping .....	85
3.4.9	Level rules of name mapping .....	86
3.4.10	Structural behavior .....	87
3.4.11	LocObj .....	87
3.4.12	Name directory .....	88
3.4.13	String Parameters .....	89
3.4.14	Owner restriction rules .....	90
3.4.15	Model information .....	91
3.4.16	Pre/Post executables .....	92
3.4.17	Character mapping .....	92
3.4.18	Units mapping .....	92
3.4.19	Site mapping and zone mapping .....	93
3.4.20	Operation messages .....	93
3.4.21	Connection information .....	94
3.5	Configuring interface operations .....	95
3.5.1	Workflow .....	95
3.5.2	Maintaining the standard table for PDMS element types .....	98
3.5.3	Standard tables for classes and subclasses .....	99
3.5.4	Creating the "@PDMSMAP" folder .....	99
3.5.5	Configuring the "@PDMSMAP" folder .....	100
3.5.5.1	Activating operation messages .....	100
3.5.5.2	Defining global variables .....	100
3.5.5.3	Using pre/post executables .....	101
3.5.5.4	Using character mapping .....	102
3.5.5.5	Using units mapping .....	103
3.5.5.6	Using site mapping and zone mapping .....	104
3.5.6	Creating a class definition object .....	105
3.5.7	Creating subclass definition objects .....	105
3.5.8	Configuring subclass definition objects of the "Pipe" and "TaggedItem" classes .....	106
3.5.8.1	Defining PDMS element types .....	106
3.5.8.2	Defining structural behavior .....	107
3.5.8.3	Defining the base object for the creation of interface objects .....	108
3.5.8.4	Configuring owner restriction rules .....	109
3.5.8.5	Configuring the model .....	111
3.5.9	Working with String Parameters in the name directory .....	112
3.5.9.1	Structure of a String Parameter .....	112
3.5.9.2	Configuring String Parameters .....	114
3.5.9.3	Configuring the "Comos attribute", "PDMS attribute/expression", "GetFunction", and "SetFunction" columns .....	117
3.5.10	Configuring unit and location mapping in name mapping .....	119
3.5.10.1	Structure of the tables for unit and location mapping .....	119
3.5.10.2	Algorithm for generating a PDMS name .....	120
3.5.10.3	Algorithm for generating a COMOS path name .....	121
3.5.10.4	Algorithm for generating a COMOS object through name mapping .....	123
3.5.10.5	Configuring unit mapping and location mapping .....	124
3.5.10.6	Configuring level rules .....	127
3.5.11	Configuring COMOS queries from PDMS .....	128

3.5.11.1	Overview .....	128
3.5.11.2	Creating and configuring COMOS queries .....	128
3.5.11.3	Creating definition objects for queries.....	129
3.5.11.4	Configuring a subclass definition object for queries .....	129
3.5.12	Configuring "Import DocLinks" .....	132
3.5.12.1	Defining the entry for the target document in PDMS .....	132
3.5.12.2	Configuring "Import DocLinks" in COMOS.....	133
3.5.12.3	Defining the String Parameter for the document name .....	133
3.5.12.4	Configuring general settings .....	134
3.5.12.5	Defining owner restriction rules for draft objects.....	134
3.5.13	Synchronizing settings .....	135
3.5.14	Importing AVEVA design templates.....	136
3.5.15	Configuring COMOS interface objects.....	137
3.5.15.1	Assigning the class, subclass, and PDMS element type .....	137
3.5.15.2	Overwriting the inherited model information .....	138
<b>4</b>	<b>COMOS 3D viewing.....</b>	<b>141</b>
4.1	Requirements.....	141
4.2	Script adjustments.....	141
<b>5</b>	<b>COMOS NX - Routing Mechanical interface .....</b>	<b>143</b>
5.1	Overview .....	143
5.2	Default settings in COMOS.....	143
5.3	Default settings in NX.....	144
<b>6</b>	<b>References .....</b>	<b>145</b>
6.1	Plant Modeler .....	145
6.1.1	User interface reference.....	145
6.1.1.1	Control elements on the "Plant Modeler" tab .....	145
6.1.1.2	"MessageSystem <mode> as <node>" window.....	147
6.1.1.3	Attribute Transfer Editor .....	148
6.1.1.4	"References" tab .....	149
6.1.1.5	"Center line routing" tab .....	150
6.1.1.6	"Data exchange" tab .....	151
6.1.2	Base data reference.....	151
6.1.2.1	Structure of the "PLM" node in the base data.....	151
6.2	COMOS PDMS Integration .....	158
6.2.1	INI file .....	158
6.2.2	Communication process.....	160
6.2.2.1	Communication modes .....	160
6.2.2.2	COMOS to PDMS .....	160
6.2.2.3	PDMS to COMOS .....	162
6.2.3	Database .....	164
6.2.3.1	Structure of the standard table for PDMS element types .....	164
6.2.3.2	Structure of the standard table for classes .....	165
6.2.3.3	Structure of the standard table for subclasses.....	165
6.2.3.4	Attribute properties.....	166
6.2.3.5	Attributes of the "@PDMSMAP" folder .....	166
6.2.3.6	Attributes of the subclass definition objects in the "Pipe" and "TaggedItem" classes .....	169
6.2.3.7	Attributes of the subclass definition objects of the "Query" class .....	172

6.2.3.8	Attributes of the subclass definition objects of the "Document" class .....	173
6.2.3.9	Attributes of the interface objects.....	175
6.2.3.10	Attributes of the design template objects.....	176
6.2.3.11	Project parameters.....	177
6.3	COMOS 3D viewing.....	179
6.3.1	Sample scripts.....	179
6.3.1.1	Sub OnProjectOpen(Project) .....	179
6.3.1.2	Exchange directory .....	180
6.4	COMOS NX - Routing Mechanical interface.....	181
6.4.1	Sample scripts.....	181
6.4.1.1	Sub OnProjectOpen(Project) .....	181
6.4.1.2	COMOS exchange directory .....	182
6.4.2	User interface reference .....	184
6.4.2.1	"NX 3D object mapping" tab .....	184





# Trademarks

## Trademarks

Registered trademark: COMOS®



# Plant Modeler

The CAD software product Microstation is a product of the Bentley Systems company and is hereafter referred to simply as Microstation.

## 2.1 Default settings in COMOS

### 2.1.1 Project properties: Setting the default location for pipes and equipment

#### Principle

In Plant Modeler, you can create pipes, pipe parts, and equipment without them immediately being assigned to a COMOS object. If pipes and equipment are not assigned to a COMOS object, COMOS does not know where the objects are to be created in the database and in the Navigator.

For this reason, on the "Plant Modeler" tab in the project properties, you must define a default node under which these types of objects should be grouped. You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Control elements on the 'Plant Modeler' tab".

#### Recommendation

Do not define a default location for pipes in the project properties. When you create a pipes, a window opens prompting you to select a location or owner.

For equipment, you must create a location.

#### Definition of the "MSI Plant Modeler" tab in the base data

The "MSI Plant Modeler" tab is located underneath the "@J Project" node on the "Base objects" tab in the Navigator. This means that the "MSI Plant Modeler" tab is displayed in the project properties. The "MSI Plant Modeler" tab has the following properties:

- Name: "MSI"
- Description: "Plant Modeler"
- Inheritance source: "PLM Microstation > Y Microstation Catalog > CTAB Microstation Catalog Tabs > MSI Plant Modeler"
- Node on the "Base objects" tab in the Navigator: "PLM Microstation"  
The "PLM Microstation" node must be available on the "Base objects" tab.

## See also

Defining the default location for pipes (Page 12)

Defining the default location for equipment (Page 13)

Structure of the "PLM" node in the base data (Page 151)

The "@J Project > @G Project settings, general" node in the base data (Page 20)

## 2.1.2 Defining the default location for pipes

### Procedure

To define the default location for pipes, proceed as follows:

1. In COMOS, open the project properties.
2. Click on the "Plant Modeler" tab.
3. Next to the "Default location for pipes" field, click "...".
4. In the "Create Default Location for Pipes" window, select the desired node.

The node you selected will appear in the "Selection" field.

5. Click "OK".

The node you selected will be displayed on the "Plant Modeler" tab.

6. Click "Reinitialize" on the "Plant Modeler" tab.
7. To save this setting, click "OK" on the "Plant Modeler" tab.

### Result

The default location for pipes is set. In addition, any 3D objects that you unassign from cells in MicroStation are moved to the specified node and are available for reassignment.

You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Canceling assignments between COMOS objects and cells".

### Creation mode

The node you set as the default location has to have the "Free" creation mode.

## Procedure

To check the creation mode, proceed as follows:

1. In the base project, open the properties of the object you have defined as the default location.
2. On the "System" tab, check if the "Free" option is enabled in the "Creation mode" control group. Enable the "Free" option if it is disabled.
3. Click "OK" to save your settings.

### 2.1.3 Defining the default location for equipment

#### Procedure

To define the default location for equipment, proceed as follows:

1. In COMOS, open the project properties.
2. Click on the "Plant Modeler" tab.
3. Next to "Default location for equipment", click "...".
4. In the "Create Default Location for Equipment" window, select the desired node.  
The node you selected will appear in the "Selection" field.
5. Click "OK".  
The node you selected will be displayed on the "Plant Modeler" tab.
6. Click "Reinitialize" on the "Plant Modeler" tab.
7. To save this setting, click "OK" on the "Plant Modeler" tab.

#### Result

The default location for equipment is set. In addition, any 3D objects that you unassign from cells in MicroStation are moved to the specified node and are available for reassignment.

You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Canceling assignments between COMOS objects and cells".

#### Creation mode

The creation mode of the node you have defined as the default location must be set to "Free".

## Procedure

To check the creation mode, proceed as follows:

1. In the base project, open the properties of the object you have defined as the default location.
2. On the "System" tab, check if the "Free" option is enabled in the "Creation mode" control group. Enable the "Free" option if it is disabled.
3. Click "OK" to save your settings.

### 2.1.4 Assigning references

## Procedure

To assign a reference, proceed as follows:

1. In COMOS, open the project properties.
2. Click on the "Plant Modeler" tab.

You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Control elements on the 'Plant Modeler' tab".

3. Click on the "..." button next to the corresponding field.
4. Select the required node in the next window.
5. Click "OK".

## Result

The reference is entered in the corresponding field.

## See also

Deleting references (Page 15)

Navigating to references (Page 17)

## 2.1.5 Deleting references

### Prerequisite

References are set in the project properties on the "Plant Modeler" tab.

### Procedure

To delete a reference, proceed as follows:

1. Open the properties of the project.
2. Click on the "Plant Modeler" tab.
3. Click on the "Remove reference" button next to the field.  
The reference is no longer displayed in the respective field.
4. Click "OK" to save your settings.

## 2.1.6 Configuring the document management settings

### Procedure

Proceed as follows to configure the document management settings:

1. In COMOS, open the project properties.
2. Click the "Plant Modeler" tab.
3. Click the "..." button next to the "Default seed file" field.
4. In the file selection dialog, select the desired seed file with the defaults for MicroStation.
5. Click "OK".  
The path and file name are entered in the "Default seed file" field.
6. To save this setting, click "OK" on the "Plant Modeler" tab.

### Result

When you start MicroStation, configuration settings from the selected seed file are used.

### See also

Control elements on the "Plant Modeler" tab (Page 145)

### 2.1.7 Configuring the MicroStation settings

#### Procedure

Proceed as follows to configure the MicroStation settings:

1. In COMOS, open the project properties.
2. Click the "Plant Modeler" tab.
3. Click the "..." button next to the "Startup file commands" field.
4. Select the "ComosPlantModeler.Startup" file in the file selection dialog.  
The file is located in the installation directory of MicroStation in the folder "mdlapps", e.g. "C:\Program Files\Bentley\MicroStationV8i\MicroStation".
5. Click "OK".  
The path and file name are entered in the "Startup file commands" field.
6. Enter the following file name in the "Interface settings" field: "comosplantmodeler".
7. To save this setting, click "OK" on the "Plant Modeler" tab.

#### Result

The MicroStation settings have been configured.

#### See also

Control elements on the "Plant Modeler" tab (Page 145)



## 2.1.8 Navigating to references

### Prerequisite

References are set in the project properties on the "Plant Modeler" tab.

### Procedure

To navigate to the references, proceed as follows:

1. Open the project properties.
2. Click on the "Plant Modeler" tab.
3. Select one of the following options:
  - If you want to navigate to the properties of the object, click on "Navigate, properties > Properties" next to the corresponding field.
  - If you want to navigate to the object serving as the reference, click "Navigate, properties > Navigate > Object" next to the corresponding field.
  - If you want to navigate to the base object in the open project, next to the corresponding field, click "Navigate, Properties > Navigate > Base Object".
  - If you want to navigate to the base object in the base project, next to the corresponding field, click "Navigate, Properties > Navigate > Base Object in Base Project".
  - If you want to navigate to the uses of the reference, click "Navigate, properties > Navigate > Uses > <use item>" next to the corresponding field.

### Result

- Once you navigate to the properties, you will see the properties of the object that is entered as the reference.
- Once you navigate to the object, the engineering object will be selected in the Navigator.
- Once you navigate to the base object in the open project, the base object will be selected on the "Base objects" Navigator tab in the open project.
- Once you navigate to the base object in the base project, the base project will be open. There, the base object will be selected on the "Base objects" Navigator tab.
- Once you navigate to the uses of the reference, the object in which the reference is used will be selected on the "Base objects" Navigator tab in the open project.

## 2.1.9 ABO objects

### 2.1.9.1 Settings for ABO objects

#### Purpose of ABO objects

ABO objects are used to assign a COMOS base object to a cell used in MicroStation. You can use them to assign a symbol and dimensions for evaluation on the isometric drawing to a non-COMOS geometry.

The symbols are displayed on the isometry report.

#### Settings

You find the ABO objects in the base project, on the "Base objects" tab under the "PLM Microstation >Y Microstation Catalog > ABO Catalog of Assignable Objects" node. You can create and modify any number of additional ABO objects in the form of a structure tree here. Various settings can be made for these objects.

The following table describes the settings settings you can make in the properties of the "ABO" objects:

Tab in the properties	Setting	Description
"Attributes > Connector mapping"	"CX1" list	Specifies the alignment of the connections. Possible values: <ul style="list-style-type: none"> <li>• Undefined</li> <li>• Left</li> <li>• Right</li> <li>• Top</li> <li>• Bottom</li> </ul> <b>Note:</b> The "CX1" connection serves as the reference for the remaining connections and must always point to the left in the symbol script. Therefore, always assign the value "Left" to the "CX1" connection. You can assign the alignment of the remaining connections freely.
	"CX2" list	
	"CX3" list	
	"CX4" list	
"Script"	Symbol script	In order to display the desired symbol for an object, you must enter a symbol script on the "Script" tab for that object. If you do not enter a symbol script, a default placeholder will be displayed. Use the Symbol Editor to process the symbols. You can find additional information on this topic in the "Reports - Basic Operation" manual, keyword "Symbol Editor".

## Assigning ABO objects

You assign the ABO objects while modifying the cells for use with Plant Modeler. Before generating connectors, create a cell header for the cell and assign an ABO object while you do this. You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Creating attribute information for placed cells".

## Recommendation

Assign the ABO objects when you create the cell library. This ensures that the ABO symbols are used consistently in a project. You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Creating attribute information in the cell library".

## Result of the assignment

During the course of the assignment process, the ABO object is allocated to the P&ID object as a 3D implementation. The symbol stored for the ABO object is displayed as the result.

## Configuring the start node

To make engineering easier, you can set a start node for the ABO objects. See also chapter Setting a start node for ABO objects (Page 19).

### 2.1.9.2 Setting a start node for ABO objects

#### Prerequisite

Plant Modeler and COMOS are not connected.

#### Procedure

To set a start node for ABO objects, proceed as follows:

1. Open the base project.
2. Call the context menu for the "PLM > Y > ABO" node on the "Base objects" tab.
3. Select the "Plant Modeler > Add to ABO start points" command.

#### See also

Removing ABO objects from the start node (Page 20)

Clearing ABO start points (Page 20)

Saving the ABO start node in the configuration file (Page 32)

### 2.1.9.3 Removing ABO objects from the start node

#### Prerequisite

Plant Modeler and COMOS are not connected.

#### Procedure

To remove the ABO objects from the specified start node, proceed as follows:

1. Open the base project.
2. Call the context menu for the "PLM > Y > ABO" node on the "Base objects" tab.
3. Select the "Plant Modeler > Remove from ABO start points" command.

### 2.1.9.4 Clearing ABO start points

#### Prerequisite

Plant Modeler and COMOS are not connected.

#### Procedure

To clear the ABO start points, proceed as follows:

1. Open the base project.
2. Call the context menu for the "PLM > Y > ABO" node on the "Base objects" tab.
3. Select the "Plant Modeler > Clear ABO start points" command.

### 2.1.10 The "@J Project > @G Project settings, general" node in the base data

---

#### Note

If you enter the script for a specific project only, Plant Modeler will only be started for this project. Use the script to define your own project structure.

---

## Script

On the "Base objects" tab in the base project, you find the "@J Project > @G Project settings, general" node. Enter the following script in the properties of the project structure so that Plant Modeler is started and initialized when the project is opened:

```
Sub OnProjectOpen(Project)
'Is called on opening a project
On Error Resume Next

If Workset.Globals.ItemExist("Microstation3D") Then
  If IsObject(Workset.Globals.Microstation3D) Then
    Set CPM = Workset.Globals.Microstation3D
  End If
End If

bCreateCPM = False
bCreateCPM = isEmpty(CPM)
If Not bCreateCPM Then
  If IsObject(CPM) Then
    If CPM Is Nothing Then bCreateCPM = true
  End If
End If

If bCreateCPM Then
  Set CPM = CreateObject("Comos.CPM.M3D.Microstation3D")
  If Not CPM Is Nothing Then
    CPM.InitMicrostationInterface Workset
  End If
Else
  CPM.Disconnect
End If

End Sub
```

### 2.1.11 P&ID data

#### P&ID module

The database must be prepared in such a way that the P&ID module is functional, and you are able to work with PipeSpecs. Among other things, this means that a pipe parts catalog has been set up and PipeSpecs have been created.

Check if a functional P&ID catalog is available. You find these catalogs in the base project on the "Base data" tab, under the following node:

"01 Material > PID Piping and Instrumentation > 01 Catalog P&ID"

You find additional information on this topic in the "Pipe Spec Designer" manual, keyword "Managing the pipe component catalog".

In addition, the components that are to be connected with a MicroStation cell must have the following tabs:

"M3D Plant Modeler", inheritance source: "PLM > Y > CTAB > M3D".

**See also**

Structure of the "PLM" node in the base data (Page 151)

**2.1.12 Linking between P&ID and PPC**

**Overview**

In order to be able to conduct a PipeSpec mapping on the P&ID, the P&ID catalog and the pipe parts catalog must be linked.

You find additional information on this topic in the "Pipe Spec Designer" manual, keyword "Managing the pipe component catalog".

**2.1.13 Cell libraries in COMOS**

**2.1.13.1 Creating a cell library in COMOS**

**Prerequisite**

Plant Modeler and COMOS are connected.

**Procedure**

To create a cell library in COMOS, proceed as follows:

1. Open the base project.
2. Select the "Base objects" tab.
3. Open the "PLM Microstation" node.
4. Right-click the "CELLIB cell libraries" node.
5. Select "Plant Modeler > Create new cell library" from the context menu.

**Result**

A new cell library "LIB" is created underneath the "CELLIB cell libraries" node.

### 2.1.13.2 Importing a cell library into COMOS

#### Procedure

To import a cell library into COMOS, proceed as follows:

1. Open the base project.
2. Open the properties of the "PLM > CELLIB > CLM CellLibraryMaster" node.
3. Click on the "Attributes > Library" tab.
4. Click on the "..." button next to the "Location" field.
5. Select the required CEL file in the file window and confirm your entry.  
The path to the selected file is displayed in the "Location" field.
6. Click on the "Import" button.

#### See also

Editing a cell library stored in COMOS (Page 23)

### 2.1.13.3 Editing a cell library stored in COMOS

#### Prerequisite

The base data includes a cell library. Plant Modeler and COMOS are connected.

#### Procedure

To edit a cell library stored in COMOS, proceed as follows:

1. Open the base project.
2. Open the "PLM > CELLIB" node.
3. Right-click on the desired CLM node.
4. In the context menu, select the "Plant Modeler > Start edit mode" command.  
The corresponding CEL file opens in Plant Modeler.
5. Switch to Plant Modeler and edit the cell library. See also Section The "COMOSPlantModelerDemo.cel" cell library (Page 43).
6. Once you have finished editing the library, switch back to COMOS.
7. Right-click on the CLM node.
8. In the context menu, select the "Plant Modeler > Finish edit mode" command.

## Result

The CEL file closes in Plant Modeler. The DGN file or CEL file opened previously continues to be displayed. Your changes are saved.

## 2.2 Installation

### 2.2.1 Installing Plant Modeler

#### Prerequisite

COMOS is installed. MicroStation V8i is installed on your local computer or in the network. The most recent Plant Modeler ServicePack for the installation is available.

#### Procedure

To install Plant Modeler, proceed as follows:

1. Run the "setup.exe" file which is included in the Plant Modeler setup.
2. Select the required language and click "Next".
3. Click "Next".
4. Select the folder where you want to install the Plant Modeler files.
5. Click "Next".
6. Specify the exchange directory in the "Exchange Directory" field.
7. In the "Language" field you specify the language in which you want to install Plant Modeler.
8. If necessary, enter the path of the seed file you require for the reference file in the "Seed file" field.
9. Click "Next".
10. Click "Install".
11. Click "Finish".

## Result

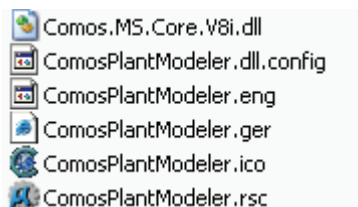
Plant Modeler is installed. The path you defined in the "Exchange directory" field is stored in the following files:

- In COMOS: "COMOS.CPM.config"
- In Plant Modeler: In the configuration file "COMOSPlantModeler.dll.config"

Plant Modeler is installed in the language you selected. Plant Modeler saves this information in the configuration file.



The following files as well as a couple of example cell libraries are stored in the "mdlapps" folder of the Bentley installation directory:



The "ComosPlantModeler\_eng.dgnlib" and "ComosPlantModelerHVACCT\_eng.dgnlib" files control the appearance of the "Piping" and "HVAC" applications. These files are contained in the directory "C:\Documents and Settings\All Users\Application Data\Bentley\MicroStation V8i (SELECTseries 1)\WorkSpace\Interfaces\MicroStation\comosplantmodeler".

The "COMOS.CPM.config" file is created in the COMOS "Config" directory.

## Basic setup

The basic setup is only conducted during the initial installation. Updates are executed as described above. See also chapter Default directories and settings in the configuration file (Page 32).

## See also

Making subsequent changes to the configuration file (Page 29)

Updating Plant Modeler (Page 25)

## 2.2.2 Updating Plant Modeler

### Procedure

To update Plant Modeler, proceed as follows:

1. Start the COMOS Update Center.
2. Update COMOS Plant Modeler by updating the the COMOS components in the usual way via the COMOS Update Center.

### Result

Plant Modeler is updated.

## 2.3 Configuration

### 2.3.1 Settings in the COMOS configuration file for Plant Modeler

#### Information in the configuration file

When you install Plant Modeler, the Plant Modeler basic setup program creates the "Comos.CPM.config" configuration file at this location:

"<COMOS installation directory>\config\COMOS.CPM.config"

The following information is stored in the configuration file:

- Exchange directory for XML files
- Start status of the interface
- Debug status
- Setup for the relevant status comparisons
- Start node for ABO objects
- Start node for hangers and supports

## Example

The following example shows a COMOS configuration file with set status queries:

```
<configuration>
  <appSettings>
    <add key="StartInterface" value="1" />
    <add key="WorkingDirectory" value="C:\Temp\" />
    <add key="ExportToDGN" value="true"/>
    <add key="DEBUG" value="1" />

    <add key="StatusName1" value="Consistency check: Topology P&I
&lt;-&gt; 3D" />
    <add key="StatusValue1" value="7" />
    <add key="StatusStart1" value="Both" />
    <add key="StatusName2" value="Consistency check: Pipe spec mapping
(P&I)" />
    <add key="StatusValue2" value="8" />
    <add key="StatusStart2" value="Comos" />

    <add key="StatusName3" value="Consistency check: P&I-3D Pipe
spec/DN/Object" />
    <add key="StatusValue3" value="9" />
    <add key="StatusStart3" value="Both" />

    <add key="StatusName4" value="Check Comos-Microstation
consistency" />
    <add key="StatusValue4" value="12" />
    <add key="StatusStart4" value="Both" />

    <add key="StatusName5" value="Consistency check: 3D layer" />
    <add key="StatusValue5" value="10" />
    <add key="StatusStart5" value="MicroStation" />

    <add key="TCPPort" value="[adlimi8=9999]" />
    <add key="CommunicationMode" value="TCP" />
    <add key="CommunicationLogEnabled" value="true" />
    <add key="CommunicationWindowEnabled" value="true" />
    <add key="CommunicationPingTimeout" value="10000000" />

    <add key="HS_StartNode1" value="@VIPER|@PPC|1|81" />
    <add key="HS_StartNode2" value="@VIPER|@PPC|2|75" />
    <add key="ABO_StartNode1" value="00VH|PLM|Y|ABO" />
    <add key="ABO_StartNode2" value="PLM|Y|ABO" />
  </appSettings>
</configuration>
```

### Status values in the configuration file

You can execute status queries either in COMOS or in Plant Modeler. The settings in the configuration file determine which status queries are displayed in COMOS and in Plant Modeler.

You have the option to set the desired status queries by opening the configuration file with a text editor and making the necessary changes. The COMOS setup offers an example of the status query which you can use as a template for settings your own status queries.

The following table describes the keys of the corresponding row in a sample configuration file:

Row	Key	Description
<code>&lt;add key="StatusName1" value="Comos Ustation Consistency" /&gt;</code>	<code>= "StatusName1"</code>	This key specifies the name of the status query that is displayed in COMOS and Plant Modeler.
<code>&lt;add key="StatusValue1" value="12" /&gt;</code>	<code>= "StatusValue1"</code>	Specifies the name of the status which is indicated in the COMOSDB base data under "@Status > @D Data > @Status".
<code>&lt;add key="StatusStart1" value="Comos" /&gt;</code>	<code>= "StatusStart1"</code>	This key specifies the application from which the user can start the status query. You can set the following values: <ul style="list-style-type: none"> <li>• "Comos": The status query can only be started from within COMOS.</li> <li>• "Microstation": The status query can only be started from within Plant Modeler.</li> <li>• "Both": The status query can be started from within both COMOS and Plant Modeler.</li> </ul>

### Objects for (pipe) hangers and supports

In the COMOS configuration file, you also have the option to define a start node for the hanger objects. The start node is a base object. The base object underneath this start node are transferred hierarchically to MicroStation, and can there be selected by the user. In this case, the hangers are released on the "Placeable objects" toolbar. They are used like the other components.

### Example start node of the objects for pipe hangers

The following example shows the definition of a start node for objects of hangers in the "COMOS.CPM.config" file:

```
<add key="Hangers and Supports_StartNode1" value="@VIPER|@PPC|1|81"/>
```

### Status entries via the COMOS context menu

You can edit the status entries in the configuration file via a COMOS context menu in the base data. This makes editing the configuration file easier. See also chapter Editing status queries in the configuration file in COMOS (Page 31).

## See also

Making subsequent changes to the configuration file (Page 29)

Communication mode (Page 35)

## 2.3.2 Making subsequent changes to the configuration file

The settings you made during the installation of Plant Modeler are saved in the configuration file. Any settings you make directly in Plant Modeler are also saved in the configuration file. You can change these settings in the configuration file at any time.

User-defined settings can be made.

## Example

```
<add key="LanguageFile" value="[user1=ComosPlantModeler.ENG,user2=ComosPlantModeler.GER]" />
```

## Procedure

To subsequently make changes to the configuration file, proceed as follows:

1. Open the "COMOSPlantModeler.dll.config" file, using a text editor, for example.
2. Change the settings as desired.
3. Save your changes.

## Result

The changed settings will take effect the next time you start MicroStation.

## See also

Default directories and settings in the configuration file (Page 32)

"Data exchange" tab (Page 151)

## 2.3.3 Status queries

### 2.3.3.1 Prerequisites for the 3D status query

#### Prerequisite

In order for the 3D status query to function, an object for the 3D status query must have been prepared beforehand in the database underneath the node in which the status queries are managed.

**Object for the status query**

You can find the node for the status query in the base project, on the "Base objects" tab in the Navigator.

"@System System settings > @D Data > @Status"

You create the new status object in this node. The new status object contains the status values as elements.

You must create the following elements as status values below the object:

Name	Description
1	Identical
2	Only in COMOS
3	Only in Plant Modeler
4	Object is inconsistent

Prepare a query with the following properties below the object:

Property	Description
Name	Query
Description	Object query: Engineering objects
Class	Action
Subclass	Object query: Engineering objects

**Script**

The "Script function for value" column must contain the following script for "@UserStatusValue":

```
Function ColumnValue(RefColObject, ColumnObject, BaseRowIndex,
    BaseColumnIndex)
    ColumnValue = 0
    If WorkSet.globals.ItemExist("Microstation3D") Then
        Set M3D = WorkSet.globals.Microstation3D
        If Not M3D Is Nothing Then
            ColumnValue = M3D.GetStatusValue(ColumnObject)
        End If
    End If
End Function
```

**See also**

Editing status queries in the configuration file in COMOS (Page 31)

### 2.3.3.2 Editing status queries in the configuration file in COMOS

You can specify which status queries are available in COMOS and which are available in Plant Modeler. The entry is made by adding a status to one or both of the following status lists:

- COMOS status list  
Call: In COMOS  
Context menu of an object in the Navigator, "PlantModeler" > "<Check status>" command
- COMOS 3D status list  
Call: In Microstation  
"Change management" toolbar, "Calculate status" button

#### Prerequisite

- See also Section Prerequisites for the 3D status query (Page 29).
- If you want to edit status queries in the configuration file via the COMOS context menu, Plant Modeler and COMOS must not be connected.

#### Procedure

To edit status queries in the configuration file in COMOS, proceed as follows:

1. Open the base project.
2. Open the "@System > @Data > @Status" node on the "Base objects" tab. See also section Making subsequent changes to the configuration file (Page 29).
3. Right-click on the desired status.
4. Select one of the following options:
  - If you want to add the status query to the COMOS status list, select "Plant Modeler > Add to the COMOS status list" from the context menu
  - If you want to add the status query to the COMOS 3D status list, select "Plant Modeler > Add to the COMOS 3D status list" from the context menu.
  - If you want to add the status query to the COMOS status list and the COMOS 3D status list, select "Plant Modeler > Add to both status lists" from the context menu.
  - If you want to remove the status query from a status list, select "Plant Modeler > Remove from status list" from the context menu.
  - If you want to remove all entries from the status list, select "Plant Modeler > Clear the status lists" from the context menu
  - If you want to save the status list in the configuration file, select "Plant Modeler > Save status lists in the configuration file" from the context menu
  - If you want to reload Plant Modeler with the modified settings for the status calls, select "Plant Modeler > Reinitialize Plant Modeler" from the context menu.

### 2.3.4 Saving the ABO start node in the configuration file

#### Prerequisite

Plant Modeler and COMOS are not connected.

#### Procedure

1. Open the base project.
2. Call the context menu for the "PLM > Y > ABO" node on the "Base objects" tab.
3. Select the "Plant Modeler > Save ABO start points in the configuration file" command.

### 2.3.5 Default directories and settings in the configuration file

The "COMOSPlantModeler.dll.config" configuration file for the Microstation add-in is located in the "mdlapps" folder of the Microstation installation directory. This configuration file is the counterpart to the "COMOS.CPM.config" configuration file.



## Configuration file example

An example of the "COMOSPlantModeler.dll.config" configuration file is provided below:

```
<configuration>
  <appSettings>
    <add key="LanguageFile" value="ComosPlantModeler.ENG" />
    <add key="ExchangeFileOut" value="C:\temp\MSMessage.xml" />
    <add key="ExchangeFileIn" value="C:\temp\CMessage.xml" />
    <add key="ReferenceFileLocation"
value="D:\PlantModeler_Microstation\reference files" />
    <add key="ReferenceSeedFile" value="C:\Documents and Settings\All
Users\Application Data\Bentley\MicroStation V8i (SELECTseries
1)\Workspace\System\seed\seed3d.dgn" />
    <add key="ChangeReferences" value="False" />
    <add key="DisableSnappingSwitch" value="false" />
    <add key="ForceImplementationOrder" value="False" />
    <add key="ShowImplementationName" value="True" />
    <add key="ShowImplementationTag" value="False" />
    <add key="ShowCurrentBranchImplementations" value="True" />
    <add key="FilePathEnvVar" value="" />
    <add key="ReferenceFilePathMode" value="1" />
    <add key="TCPPort" value="[adlimi8=9999]" />
    <add key="TCPHost" value="127.0.0.1" />
    <add key="CommunicationMode" value="TCP" />
    <add key="CommunicationPingTimeout" value="30000000" />
    <add key="CommunicationConnectTimeout" value="3000" />
    <add key="CommunicationMaxPings" value="4" />
    <add key="CommunicationLogEnabled" value="true" />
    <add key="CommunicationWindowEnabled" value="true" />
    <add key="CommunicationLogFile" value="c:\log.txt" />
    <add key="CommunicationPassword" value="" />
    <add key="CommunicationThreadSleep" value="50" />
    <DirectConnectorAccept>False</DirectConnectorAccept>
    <DirectConnectorAccept>False</DirectConnectorAccept>
    <add key="DirectConnectorAccept" value="False" />
    <add key="WorkingDirectory" value="C:\Temp\" />
  </appSettings>
</configuration>
```

## Language setting

The keys and values for language settings are listed below:

<b>Key</b>	"ReferenceFileLocation"
<b>Values</b>	<ul style="list-style-type: none"> <li>• For German: "ger"</li> <li>• For English: "eng"</li> </ul>

### Example of a language setting for English

```
"<add key="LanguageFile" value="ComosPlantModeler.ENG" />"
```

### Exchange directory

The keys and values for the exchange directory are listed below:

<b>Key</b>	<ul style="list-style-type: none"> <li>"ExchangeFileOut"</li> <li>"ExchangeFileIn"</li> </ul>
<b>Values</b>	<p>You can freely select the path to the exchange directory. The file names are predefined:</p> <ul style="list-style-type: none"> <li>File name for messages from COMOS: "CMessage.xml"</li> <li>File name for messages from Plant Modeler: "MSMessage.xml"</li> </ul>

Now change the config file in the COMOS installation directory accordingly. Ensure that the entries in the config file and the configuration file are consistent.

You can display the exchange directory directly in MicroStation, but you cannot change it there. Call the "Administration > Settings" menu. You find the exchange directory on the "Data exchange" tab.

### Directory for reference files

The keys and values for the reference file directory are listed below:

<b>Key</b>	"ReferenceFileLocation"
<b>Default value</b>	C:\Documents and Settings\All Users\Application Data\Bentley\MicroStation\Workspace\Projects\Untitled\dgn

### Directory for seed files

The keys and values for the seed file directory are listed below:

<b>Key</b>	"ReferenceSeedFile"
<b>Default value</b>	C:\Documents and Settings\All Users\Application Data\Bentley\MicroStation\Workspace\System\Seed\seed3d.dgn

### Reference file changes

The keys and values for changes to reference files are listed below:

<b>Key</b>	"ChangeReferences"
<b>Values</b>	<ul style="list-style-type: none"> <li>"True"</li> <li>"False"</li> </ul>

## See also

Communication mode (Page 35)

## 2.3.6 Settings for the "HVAC" add-in in the configuration files

### Introduction

For the "HVAC" add-in, the settings are stored both in the configuration file of COMOS and in the configuration file of the Plant Modeler.

### COMOS configuration file

The configurations of COMOS are stored in the configuration file "COMOS.CPM.config". The user has to add the following lines to the configuration file in order to use the "HVAC Centerline Tool":

```
<appSettings> <add key="AddInFile1" value="Comos.CPM.AddIn.HVACCT" /> <add key="AddInType1" value="Comos.CPM.AddIn.HVACCT.ComosHVACCTAddIn" /></appSettings>
```

### Plant Modeler configuration file

The configurations of the Plant Modeler are stored in the configuration file "COMOS.PlantModeler.dll.config". The user has to add the following lines to the configuration file in order to use the "HVAC Centerline Tool":

```
<appSettings> <add key="LanguageFile" value="ComosPlantModeler.eng;ComosPlantModelerHVACCT.eng" /></appSettings><AddIns> <AddIn file="ComosPlantModelerHVACCT" type="PlantModeler.AddIn.HVACCT.CAddInHVACCT" /></AddIns>
```

## 2.3.7 Communication mode

### 2.3.7.1 Setting communication mode

You can set two communication modes for data exchange between Microstation and COMOS.

- File-based
- Via TCP/IP (server: COMOS; client: Microstation)

The file-based communication mode is used by default.

To change the communication mode or the default values, you must add a series of keys to the ComosPlantModeler.dll.config configuration file and possibly to the Comos.CPM.config file. See also Section Keys for communication mode (Page 36).

### Setting TCP/IP communication mode

1. Add the nodes of the `TCPPort`, `TCPHost`, and `CommunicationMode` keys to the `ComosPlantModeler.dll.config` file. See also Section Keys for communication mode (Page 36).

You also have the option of inserting the other keys for the communication mode and specifying values if you do not want to use the default values.

2. Enter a value for the `TCPPort` key.  
Example: 55555 or 127.0.0.1
3. Enter a value for the `TCPHost` key.
4. Enter the value `TCP` for the `CommunicationMode` key.
5. Enter these keys with the same values in the `Comos.CPM.config` file also.
6. Restart Microstation and COMOS.

### Setting file-based communication mode

To switch back to the file-based communication mode, enter the value `FILE` in both files for the `CommunicationMode` key. Restart Microstation and COMOS.

### See also

Settings in the COMOS configuration file for Plant Modeler (Page 26)

Default directories and settings in the configuration file (Page 32)

### 2.3.7.2 Keys for communication mode

#### Keys for communication mode

The following keys are defined for communication mode in Microstation with the specified default values.

With the exception of `TCPHost`, all keys can also be entered in the `Comos.CPM.config` file. Otherwise, the same default values as for Microstation apply for COMOS.

## Example

In the following example, the communication mode "TCP" is used:

```
<add key="TCPPort" value="5555" />
<add key="TCPHost" value="127.0.0.1" />
<add key="CommunicationMode" value="TCP" />
<add key="CommunicationPingTimeout" value="3000" />
<add key="CommunicationConnectTimeout" value="3000" />
<add key="CommunicationMaxPings" value="4" />
<add key="CommunicationLogEnabled" value="true" />
<add key="CommunicationLogFile" value="c:\log.txt" />
<add key="CommunicationPassword" value="" />
<add key="CommunicationThreadSleep" value="50" />
<add key="CommunicationWindowEnabled" value="true" />
```

An example in which the communication mode "TCP" is used is available under "Settings in the COMOS configuration file for Plant Modeler (Page 26)".

## Keys for communication mode

Key	Value
TCPPort	Relevant for TCP/IP Enter a TCP port or an IP address here. This value must be identical in both configuration files. Default value: 54321
TCPHost	Relevant for TCP/IP Enter the name or IP address of the computer on which COMOS is installed here. Default value: 127.0.0.1
CommunicationMode	<ul style="list-style-type: none"> <li>• FILE</li> <li>• TCP</li> </ul> This value must be identical in both configuration files. Default value: FILE
CommunicationPingTimeout	Relevant for file-based communication mode Enter a value in milliseconds indicating when a ping is sent in the absence of a message from COMOS. Default value: 30000
CommunicationConnectTimeout	Relevant for file-based communication mode Enter a value in milliseconds indicating how long Microstation waits for a connection to be established. Default value: 3000
CommunicationMaxPings	Relevant for file-based communication mode Enter how many consecutive attempts Microstation makes to establish a connection with COMOS in the event of a connection failure. Default value: 4

Key	Value
CommunicationLogEnabled	<p>Relevant for both communication modes</p> <p>Enter whether the information exchanged between COMOS and Microstation is to be recorded in a log file.</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul> <p>Default value: false</p>
CommunicationLogFile	<p>Relevant for both communication modes</p> <p>Enter the absolute file path and the file name of the log file to be created.</p> <p>Example: C:\PlantModeler\Logfile.txt</p> <p>Default value: ""</p>
CommunicationPassword	<p>Relevant for both communication modes</p> <p>You can enter a password if you wish. So that the connection can be established, the same password must be entered in the Comos.CPM.config file.</p> <p>This value must be identical in both configuration files.</p> <p>Default value: ""</p>
CommunicationThreadSleep	<p>Relevant for both communication modes</p> <p>Enter a value in milliseconds indicating how often Microstation checks for a message from COMOS.</p> <p>Default value: 50</p>
CommunicationWindowEnabled	<p>Relevant for both communication modes</p> <p>Specify whether a debug tool for communication between COMOS and Microstation should open when Microstation starts up.</p> <p>See also chapter "MessageSystem &lt;mode&gt; as &lt;node&gt;" window (Page 147).</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul> <p>Default value: false</p>

## 2.4 Attribute Transfer Editor

### 2.4.1 Introduction

In the Attribute Transfer Editor, you can create a template in which you define which additional information on an object you wish to transfer to the Plant Modeler from COMOS.

You can assign a scheme to each 3D object and thus only allow certain information to be output in the Plant Modeler. You can call the information from Microstation with the standard tools.

In order to use the Attribute Transfer Editor, the following items must be available:

- Plant Modeler license
- Microstation V8i (SELECTseries 3)

## 2.4.2 Making project settings

If you do not wish to apply the defaults for attribute exchange, modify the settings.

### Procedure

1. Open the "Plant Modeler" tab in the project settings.
2. In the "Template for scheme" field in the "Attribute exchange" control group, enter the base object which you wish to use for the creation of a new template. The base object must have an attribute with the name "Sche" on the "XML" tab.  
Standard: "PLM > Y > BO > SC"
3. In the "Library for schemes" field, enter the scheme library in which you want to save created schemes. The scheme library has to be a folder.  
Standard: "PLM > LIBSC"

## 2.4.3 Create a template

Depending on the information on a COMOS object which should be displayed in Microstation, you can create different templates.

### Procedure

1. In the base project, open the Attribute Transfer Editor via the menu "Administrator > Base data > Attribute Transfer Editor". See also chapter Attribute Transfer Editor (Page 148).
2. To create a new template, enter a name in the "Template name" field.  
Click the "Create template" button.  
The name is not permitted to contain spaces or symbols, apart from periods and underscores. It is not permitted to begin with a period or a number.  
You have created a base object for a template which you can specify further afterwards. The base object was stored in the library for templates which you defined in the project settings.
3. To edit an existing template, drag&drop the base object of a template from the library to the "Template CDevice" field.  
Click the "Edit" button.  
Only edit templates using the Attribute Transfer Editor.

4. Drag&drop the base object of an object into the empty field of the "COMOS Base object properties" control group.  
The tabs and attributes of the object are displayed in the upper area. If you select an object in the upper area, the available system properties of the object are displayed in the lower area.
5. Drag&drop attributes and system properties into the attribute lists in which the relevant information should be recorded.
6. To delete individual entries from the attribute lists, select an entry and press the <Del> key.
7. To empty an attribute list, click the "Delete" button of the relevant attribute list.
8. To modify the entry description, double-click a field in the column "Description".
9. Confirm with "OK".

## Result

You have created a template which you can now assign to a 3D object. See also chapter Assign template (Page 40).

### 2.4.4 Assign template

#### Procedure

1. In the base project or the engineering project, open the properties of a 3D object under the node "@VIPER > @PPC Pipe part catalogs" or under "PLM > Y > ABO Assignable objects catalog".
2. Open the "Attributes > Plant Modeler components" tab.
3. In the "Default for scheme" field, enter the template which you wish to use for attribute transfer for the engineering objects of this base object.

## Result

Depending on the attributes which you selected in the template for attribute transfer, the corresponding information is transferred.

You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Transferring attributes using a template".



## 2.5 Default settings in Plant Modeler

### 2.5.1 Configuring settings in Plant Modeler

#### Prerequisite

You have started Plant Modeler. Plant Modeler is connected to COMOS.

#### Procedure

To configure settings in Plant Modeler, proceed as follows:

1. Click on the "COMOS Plant Modeler > Administration > Settings" menu.  
The "Settings" window opens.
2. Make the desired settings on the different tabs of the "Settings" window.
3. Click "Save".

#### Result

Your settings are saved and written to the configuration file.

#### See also

"References" tab (Page 149)

"Center line routing" tab (Page 150)

"Data exchange" tab (Page 151)

Default directories and settings in the configuration file (Page 32)

### 2.5.2 Plant Modeler KeyIns

KeyIns are defined in Plant Modeler to provide you with support when drawing 3D pipe plans. These KeyIns supplement the standard KeyIns from Microstation.

#### 2.5.2.1 Available Plant Modeler KeyIns

Two KeyIns are defined in Plant Modeler for use when routing center lines. If center line routing has not yet been activated, these KeyIns have no effect on AccuDraw.

#### lpd KeyIn

The lpd KeyIn is used to determine the construction direction. You can use the KeyIn to align AccuDraw with the direction of the dynamic center line at the mouse cursor.

## sd KeyIn

The sd KeyIn is used to construct a slope in the center line. The slope changes according to the value you entered previously in the "Slope" field in the "Center Line Routing" window, in the direction of the Y axis.

### 2.5.2.2 Defining AccuDraw shortcuts for Plant Modeler KeyIns

Defining AccuDraw shortcuts for KeyIns makes it easier for you to access the corresponding functions.

#### Procedure

1. Activate AccuDraw.
2. Open the "AccuDraw Shortcuts" window by pressing the <?> key with AccuDraw activated.  
All key combinations you have defined thus far are displayed.
3. Click "New".  
The "New Shortcut" window opens.
4. Enter a key combination in the "Shortcut" field.
5. Enter a description of the KeyIn in the "Description" field.
6. In the "Command" field, enter the KeyIn that is to be executed by the key combination.

#### Result

The function is available to you as an AccuDraw shortcut.

### 2.5.3 MicroStation handle

#### Overview

Disable MicroStation handles for the following reason:

Using MicroStation handles to modify cells corrupts COMOS-specific 3D data.

## 2.5.4 The "COMOSPlantModelerDemo.cel" cell library

### Overview

The "COMOSPlantModelerDemo.cel" cell library is located in the "mdlapps" directory. This cell library contains a nozzle, a vessel, and a pump, all with the necessary connectors.

You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Cell management".



## COMOS PDMS Integration

The PDMS software is a product of the AVEVA company and is hereafter referred to simply as PDMS.

### 3.1 Activating COMOS PDMS Integration for COMOS

#### Activating COMOS PDMS Integration for COMOS

In COMOS, you activate the interface on a project-specific basis.

You only need to activate the interface once for each project. The interface remains activated, even in subsequent sessions, until you deactivate it. It only has to be reactivated after it has been deactivated.

To work with COMOS PDMS Integration, you must activate the interface in AVEVA PDMS every time you restart AVEVA PDMS. You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Activating COMOS PDMS Integration for PDMS".

#### Procedure

To activate COMOS PDMS Integration in COMOS, proceed as follows:

1. Start your full version of COMOS.
2. Open the project in which you will work.
3. Open the project properties and go to the "PDMS interface/construction assistant" tab.
4. Click the "Activate" button.

Result:

- COMOS PDMS Integration is activated.
  - A script is saved in your COMOS user profile to ensure that COMOS PDMS Integration is automatically activated as soon as you open this project.
5. Start your work. Whether or not you use COMOS or the construction assistant will depend on the task you are working on:
    - Synchronizing P&ID data and 3D planning data: Construction assistant
    - Configuring the interface operations area: COMOS
    - Catalog and pipe spec management: The "PDMS Interface" plugin or the COMOS PipeSpec Designer

## Deactivating the interface

To deactivate the interface, proceed as follows:

1. Start your full version of COMOS.
2. Open the project for which you want to deactivate the interface.
3. Open the project properties and go to the "PDMS interface/construction assistant" tab.
4. Click "Deactivate".

## 3.2 Cats&Specs

### 3.2.1 General information regarding catalogs (Cats)

#### Link between AVEVA templates and COMOS

An important function component of COMOS PDMS Integration is the link with the AVEVA template database. The link between the COMOS pipe part catalog (PPC) and the AVEVA templates is provided by the COMOS DB.

The AVEVA template database contains templates for the latest pipe part categories (GTypes). The term template designates a catalog template, meaning a parameterized element such as a flange, valve or pipe elbow, for example, which determines the basic appearance of each pipe part. However, their geometry, including the P-points, is only determined by the setting of concrete dimensions and design parameters. Since most of the templates are flexible with regard to their forms of connectors and gaskets, they can be created for welded, plugged-in, bolted or flanged forms of connectors.

A template consists of a point set (PTSE) to specify the individual P-points, a geometry set (GMSE) to specify the geometry and a data set (DTSE) to specify the individual parameters and their designations (DKEY).

When the PPC is exported from COMOS to AVEVA PDMS, the pipe parts it contains are "expanded" to cover all required nominal diameters (as SCOMs). The geometry data from the relevant industrial standards is written from COMOS to the AVEVA templates' parameter lists. They are also available to the AVEVA PDMS user without having to be specially maintained in AVEVA PDMS.

A standard COCO table is used for the generation of connector types in COMOS. COMOS PDMS Integration maps the connector types available in COMOS to the existing user COCO tables.

## 3.2.2 Pipe parts from the pipe part catalog (PPC objects)

### 3.2.2.1 General

#### Expansion of pipe parts

In order to make the pipe parts from the pipe part catalog (PPC) PDMS-capable, they need to be expanded by means of a number of additional attributes. These are located on the "Attributes > External 3D Interface" tab of the PPC objects.

The pipe part catalog (PPC) is located in the base project in node:

"@VIPER > @PPC Pipe Part Components"

The attributes of the "External 3D Interface" tab are inherited from the following tab:

"@PDMS > @Y > @CHP > 03 > 03 Comos PPC object"

### 3.2.2.2 "External 3D Interface" tab

#### Purpose of the tab

The "External 3D Interface" tab is where you make the settings that are necessary for the successful exchange of COMOS PPC objects with AVEVA PDMS. Linking the PPC objects with the AVEVA PDMS templates serves as mapping of the relevant PDMS attributes with COMOS.

You can also create answer attributes for the PDMS selectors in the PPC object. See also chapter Answers for selectors (Page 60).

### 3.2.2.3 Assigning an AVEVA PDMS template

#### Procedure

To assign a template, proceed as follows:

- Set the "GType" filter, and the "Source catalog" filter as an option. Information regarding other filter criteria: see further below.

Result: Only the AVEVA PDMS templates that match the filter criteria are offered in the "Cate" list.

- Select the required AVEVA PDMS template from the "Cate" list:

Information regarding the first-time selection of a Cate: see further below.

A Cate can only be selected if the "Locked" field is not active. Information regarding this: see further below.

## Result

When the template is selected, the AVEVA PDMS attributes and additional information it contains are read in and saved on the "E3D External 3D Interface" tab. See also chapter Import template (Page 50).

If COMOS cannot find a matching template with the specified filter criteria, you can make a setting in the "Fill the Cates list" dialog window to ignore the filters that have been set and display all templates of the XML file.

If you answer with "No", you can select another XML file in a separate dialog in which it is searched for the templates. Information regarding this: see further below.

## Selecting a Cate for the first time

When you click on the dropdown menu of the "Cate" field for the first time, the "Load Cate CAP" dialog window opens.

You have two options:

1. Activate the "Read from PDMS" option and set the code accordingly.

Result: The Cates are read from AVEVA PDMS.

2. Enter the file path to an XML file.

Result: The Cates are read in from the XML file.

---

### Note

You can only confirm your setting in the window with "OK" if the referenced XML file has a valid format.

---

## Reading Cates from the AVEVA PDMS catalog

If communication with AVEVA PDMS is established, you can import all Cates from the catalog specified in the "Source Catalog" field from AVEVA PDMS. The Cates are then available for you in an XML file for subsequent selection in the "Cates" list.

Proceed as follows:

1. Call the context menu at the COMOS object in the Navigator.
2. Select the "Create XML file with Cates from catalog 'Catalog name'" command.
3. In the following window, decide whether only those Cates that match the filter are to be imported, or all Cates. Click "No" for all Cates.

The catalog is stored in the "E3D.Catalog" attribute.



## "GType" list

From the "GType" list, select the AVEVA PDMS GType to which the object corresponds.

---

### Note

This field is a mandatory field.

The following standard table is stored in the "E3D.GTYPE" attribute:

```
"@3D > 00 > PDMS > 04 GTYPE"
```

Additionally filters all PDMS base objects (Cates) during the import to GType.

---

## "Source Catalog" list

Owner of all PDMS base objects (Cates) in AVEVA PDMS. Select the corresponding AVEVA PDMS catalog from the "Source Catalog" list.

The following standard table is stored in the "E3D.Catalog Source Catalog" attribute:

```
"@3D > 00 > PDMS > CATA"
```

The list filters the Cates according to their catalog during the import. This filter is optional. It should be set if an import via the "GType" filter results in too many hits.

Information on how to refresh the catalog selection: see further below.

## Set more filter criteria

In addition to GType and source catalog, which can be set as filters by default, the Cates can also be filtered according to other attributes.

Proceed as follows:

- Implement a script at the "E3D.CATREF Cate" attribute (the "UseScriptBlock1" script function, for example) to filter the Cates according to additional attributes.
- Following applies for the implementation: The name of the attribute on the tab must be identical to that of the corresponding attribute in the XML file.

Example: Filtering the "E3D.Section" attribute

```
Function CatrefFilterItems()
  ReDim arrFilterItems(1)
  arrFilterItems(1) = "E3D.Section"
  CatrefFilterItems= arrFilterItems
End Function
```

## Refresh catalog selection

To update the content of the standard table stored in the "Source catalog" field, proceed as follows:

Select the base object in the Navigator.

- Select "PDMS > Update PDMS catalog (CATA)" from the context menu.

Result: The "Load CATA" window opens.

- "Load CATA" window: Either select the "Read from PDMS" option and set the coding accordingly or else input the path to the corresponding XML file.

---

**Note**

You can only confirm the changes with "OK" if the referenced XML file has a valid format.

---

### Field "Source XML file"

The path to the XML file that is set for the template search is displayed in the "Source XML file" field on the "External 3D Interface" tab and saved in the "E3D.FileName Source XML file" attribute.

If there is no valid file in the "Source XML file" field or no matching Catref is found in the file, click the "..." button to open a file explorer and enter the new file path.

### Locking the "Cate" list

As soon as you have selected a template, COMOS reads it out and stores the information it contains on the "E3D External 3D Interface" tab.

This overwrites the previous assignment settings for the AVEVA PDMS attributes.

To prevent the templates from being changed accidentally, lock the "Cate" list.

Enable the "Locked" option on the right of the "Cate" list.

The blocking of the "Cate" field also creates a unique name for the Cate (value for the "E3D.CATREF" attribute). See also chapter Name generation for Cate (Page 70).

### 3.2.2.4 Import template

#### Importing data

The data of the AVEVA PDMS template is read into the attributes of the "Cate attributes / SCOM references / preconfiguration selectors" control group as soon as you select a template from the "Cate" field.

The following sections exactly describe what happens during this process.

## Example of pipe elbow

Cate attribute / SCOM references / preconfiguration selectors	
Gtype	ELBO - fitting elbow ▼
Source catalog	/VESTemplates ▼
Section	/VESTemplates_ELBO
Purpose	
SKEY	
Ptref_SCOM	/TMPLELB001-PTSE
Gmref_SCOM	/TMPLELB001-GMSE
DtrefOwner_SCOM	/TMPLELB001
STYP	ELBW

## CateOwner

The value is read from the AVEVA PDMS template in the "Source catalog" and "Source section" fields of the "Cate attributes / SCOM references / preconfiguration selectors" control group.

Information on how to add the catalog entered under "Source catalog" to the standard table: See also chapter Assigning an AVEVA PDMS template (Page 47).

## Cate attributes

The following Cate attributes are automatically inserted in the COMOS DB under the "Source section" field:

- "Purpose"
- "Skey"

All other Cate attributes are only read in if a corresponding attribute has been created at the base object in COMOS on the "E3D External 3D Interface" tab. The name of the attribute on the tab must be identical to that of the corresponding attribute in the XML file.

## Attributes/parameters set to SCOMs

All attributes that are not set in the Cate but instead in the SCOM and are the same for all SCOMs are created automatically underneath the Cate attributes. These attributes always end with "\_SCOM".

The SCOM parameters and attributes that are not the same for all SCOMs are listed in the table of the "SCOM mapping" control group and saved in the "V Nominal diameter dependent table" attribute of the "VDM Data sheet (DN1)" tab. See also chapter Mapping AVEVA PDMS attributes to COMOS (Page 52) and section "VDM Data sheet" tab (Page 61).

### User-defined attributes

User-defined attributes (UDAs) created in AVEVA PDMS are only read in if a corresponding attribute has been created at the base object in COMOS on the "E3D External 3D Interface" tab. The name of the attribute on the tab must be identical to that of the corresponding attribute in the XML file.

### Descriptive texts

The descriptive texts from AVEVA PDMS are written to the table of the "Detail text mapping" control group.

See also chapter Mapping AVEVA PDMS attributes to COMOS (Page 52).

### PTSE, GMSE and DTSE

The point set (PTSE), geometry set (GMSE), and data set (DTSE) are imported into the "PDMS sets" control group.

Click the "..." button to view the data in a separate detail window and edit it if required.

---

#### Note

Changes to this data are not checked further for correctness when they are exported by COMOS.

---

### Field "Description"

The attribute for the description is read from AVEVA PDMS into the "TEXT description" field, if it is available. The value of the field is then exported again.

### "Amount" field

This field is not relevant.

## 3.2.2.5 Mapping AVEVA PDMS attributes to COMOS

### The "SCOM mapping" table

After selecting the PDMS template, all SCOM parameters and any SCOM attributes that are not the same for all SCOMs are read in and created on the "VDM Data sheet" tab.

Furthermore, all PDMS attributes are listed in the "SCOM mapping" table on the "External 3D Interface" tab.

SCOM mapping							
	PDMS attribute	Reference attribut / value	PDMS name	Ins...	Unit n...	Uni...	PD...
1	SCOM	=SNew("VDM.V.Catref...	---				
2	nominal bore	=S("VDM.V.VC11")	/*-PA1				
3	connection type	=S("VXC.PCON1")	/*-PA2				
4	outer diameter	=S("GD.V001")	/*-PA3				
5	wall thickness	=S("GD.VS501")	/*-PA4				
6	Dtref	"/CIS_Tube01-DTSE"					
7	Blrfarray	=SBolt("VDM.V.Blrfarr...					
8							

Assign the corresponding COMOS attributes to the PDMS attributes in this table and modify the exact settings to meet your requirements.

To do this, proceed as described below.

The table has the following columns:

Column	Function
"PDMS attribute"	The relation to the PDMS attribute (description or element type)
"Reference attribute / value"	The COMOS attribute or the value calculated from several COMOS attributes which is assigned to the PDMS attribute. Use mathematical operators to link multiple attributes with one another. Configuration details: see further below.
"PDMS name"	The name used for the PDMS attribute in PDMS. The "*" symbol is a placeholder for the in the "Cate" field selected catalog.
"Inspection attribute"	If the value "*" (=1) has been set in this field, COMOS prevents the SCOM from being exported if at least one attribute has a value "0".
"Unit name"	Full name of a unit from the COMOS unit system. The PDMS value is automatically converted into the here specified unit.
"Unit label"	Identifier of the unit that is referenced in the "Unit name" column. Is set automatically by COMOS as soon as you set the "Unit name" field.
"PDMS label for unit"	Unit label which the attribute value should have in PDMS. If no unit is to be used in PDMS, then there is an "-".

### Note

Unused parameters must be manually set to value "0".

### Creating an attribute assignment

To assign a COMOS attribute to a PDMS attribute, proceed as follows:

- Select an attribute to which you wish to assign a COMOS attribute from the "PDMS attribute" column.
- Specify the COMOS attribute in the "Reference attribute" column.

- Use the following format:

```
=S("Tab name.Attribute name")
```

- If you do not enter a tab name, the attribute will be searched on the "GD" tab.
- You can use mathematical operators to link multiple attributes with one another.
- Example:  $=S(X.Y)+S(Y.Z)/2$
- Fixed texts must be enclosed in quotation marks.

Example: `"/TMPLLELB001-DTSE"`

Specialties:

- Nominal diameters:

In the case of the nominal diameter the relation to the "VDM.V.VC11" and "VDM.V.VC21" list attributes must remain set.

- Connector types:

If you address an attribute for a "VC13", "VC23", "VC33", or "VC43" connector type, then the COCO table is also evaluated during the export by COMOS. See also chapter COCO tables (Page 65).

### Entry "SCOM" in the mapping table

Column "PDMS attribute", value = "SCOM": The line defines the rule on the basis of which the SCOM objects are named.

For example, a unique name is created by using the following formula:

```
=SNew("VDM.V.Catref", "%E3D.GTYPE%%UID%%ND1-3%%ND2-3%")
```

The relevant attributes are then located between the "%" character; thus when the formula is evaluated, then also its values such as the description.

The contents of the list attribute in the "Catref" field are complemented by the GType, System UID, and two nominal diameters.

See also chapter Naming conventions (Page 67).

---

#### Note

If the entry in the "SCOM" field refers to the "VDM.V.Catref" list attribute, then the names in the column of the list attribute units need to be modified manually for the individual SCOMs so that they are unique.

---

Since this does not involve a SCOM parameter attribute, the entry in the "PDMS name" column should also remain set to "---".

## Entry "Blrfarray" in the mapping table

Column "PDMS attribute", value = "Blrfarray"

The line generates a column for the bolting sets (BTSE) in the nominal diameter-dependent list attribute of the object which you specified in the formula.

Example:

- Formula: =SBolt("VDM.V.Name of Column")
- The column for the bolting sets is created on the "VM" tab in the "V" list attribute.

The unique name for the bolting sets is generated automatically by COMOS. See also chapter Name generation for bolting sets (BTSE) (Page 72).

## Unit systems

To convert the attribute value unit into another unit during the export, proceed as follows:

- Select the COMOS menu "Administrator > Base data > Unit systems" from the menu bar.

Result: The "Unit systems" tab opens.

You can find additional information on this topic in the "Basic Operation" manual, keyword "Unit systems".

- Take the full name of the COMOS unit from the unit system.
- "Unit name" column: Enter the full name of the COMOS unit into which the value is to be converted in PDMS.
- "PDMS label for unit" column: Enter the name of the unit as it is specified in PDMS.

To suppress the display of the designator for the unit in PDMS, simply input a minus sign in the column. This is also an option for the degrees character, since there are problems displaying it in PDMS.

## Result

- The unit label is automatically displayed in the "Unit label" column.
- During the export, the value of the PDMS attribute is automatically converted into the unit you specified in "Unit name".

When the unit is converted, the value of the attribute is also converted automatically by COMOS during the export operation.

---

### Note

Create a standard table with common units and link this with the attribute for the "Unit name" column.

---

The following standard table can be found in the drop-down menu of the COMOS DB:

"@3D > 00 > PDMS > Unit Selected units"

### Mapping for descriptive texts

The mapping table for the descriptive text is located in the "Detail text mapping" control group:

Detailtext mapping		
Detail	1	2
DetRef	=SNew("VDM.V.Detref", "%E..	
SKEY	=S("VXC.VXSKY")	
RTEXT	=SI("VTX.VST02",1)	
TTEXT	=SI("VTX.VST02",0)	
STEXT	unset	

To configure the table, proceed as follows:

- Use the following format to reference attributes:

```
=S("Tab.Attribute")
```

- Attributes relating to a language are to be referenced with:

```
=SI("Tab.Attribute", country code)
```

The country code corresponds to the values set in the project properties in the "Country code" column on the "Languages" tab.

- In the case of numeric values the decimal delimiter from the language settings will be taken into consideration.
- Texts must be included without quotation marks.
- If you do not want to reference a field, enter the following value:

```
unset
```

- As in the "SCOM" field, you can create a name in the "DTREF" field for each to be generated SCOM in order for it to be unique. Do this by inputting, for example:

```
=SNew("VDM.V.Detref", "%E3D.GTYPE%UID%_D_%ND1-3%%ND2-3%")
```

Also see the "SCOM" entry in the mapping table.

- If a SCOM has multiple descriptive texts for an attribute, then the table has a corresponding number of columns.
- Information regarding the referenced attributes on the "Text modules" tab and the TValues it contains. The TValues have already been set in the COMOS DB.

### "Default mapping set" button

If you click the "Default mapping set" button in the COMOS DB, the mapping of the attributes is reset to a default value predefined in COMOS. This is realized with the help of the "OnClick" script function and is implemented through a special script.

The default values for the mapping of the attributes are stored in the following standard table:

```
"@3D > 00 > PDMS > PDMS_Templates Default Mapping"
```

This standard table is to be adjusted accordingly for an individual mapping.



### 3.2.2.6 Field "Mask for PCOM name generation"

In the "Mask for PCOM name generation" field there is the rule by which the name for the pipe spec element as soon as the PPC object of a pipe spec has been added.




See also chapter Naming conventions (Page 67).

### 3.2.2.7 Field "Material"

#### Purpose

The "Material" field contains a material from the material catalog.

There are three buttons next to the "Material" field:

	Opens a window in which you can select a new material from the available materials.
	Removes the current reference to the material.
	Navigates to the object of the material or its inheritance sources. Alternatively, you can also open the object properties of the material.

#### Attributes of a material object

The material objects have the following attributes:

- XTEXT, YTEXT and ZTEXT
- Cate

References the PDMS object if the material had originally been created in AVEVA PDMS underneath the object.

#### Material catalog

The material catalog is located under the following node in COMOS:

"@PDMS > @SMTE Material Text"

All materials from AVEVA PDMS found when importing the XML files are located below this node. The materials are automatically supplemented.

### 3.2.2.8 Automatically adding additional pipe parts to the PipeSpec

#### Procedure

When you add the PPC object to a pipe spec, you can specify an additional pipe part which is automatically added to the pipe spec and is recognized by AVEVA PDMS as belonging to the PPC object.

To do this, drag the according pipe part from the pipe part catalog in the Navigator to the properties of the PPC object:

Tab "Attributes > External 3D Interface", control group "Attributes for pipe spec element", field "Additional PCL element"

The reference to the pipe part is saved in the "E3D.PCOMPlus1 Additional PCL element" attribute.

### Entering additional PCL elements

To automatically add multiple pipe parts to the pipe spec, you have to create more attributes at the base object. For the second pipe part an attribute called "E3D.PCOMPlus2", for the third pipe part "E3D.PCOMPlus3", and so on.

See also chapter Window for name generation (Page 69).

### 3.2.2.9 Export settings

#### Procedure

Make the following settings for the export operation:

- Field "Target catalog":

Specify the PDMS catalog into which the objects are to be exported.

If the catalog does not already exist in AVEVA PDMS, it is created.

If no value has been input in the field, the objects are exported to the catalog that is stored in the "Source Catalog" field. If this field is also blank, then the following object is evaluated in the PML code.

The target catalog also determines the database in which it is to be exported. It is defined within AVEVA PDMS which database is assigned to which catalog.

- Field "Target Section":

Same as "Target Section" field. If no value is input in the field, the objects are exported into the section that is stored in the "Section" field.

- "COCO Table" field:

– In order to set the reference to the COCO table, click the "..." button.

Result: The "Load CCTA" window opens.

– Enable the option at the bottom of the window and select the XML file for the COCO table.

Result:

All CCTAs that contain the COCO tables are listed in the detail area.

- If an XML file contains several CCTAs: Select the required CCTA from the detail area.

See also chapter COCO tables (Page 65).

The XML file containing the CCTAs can be generated in AVEVA PDMS.

### 3.2.2.10 Bolts

#### Purpose of the "Bolting sets" control group

In the "Bolting sets" control group, you define the bolts belonging to the PPC object for AVEVA PDMS.

For example, flanged pipe parts require a bolt definition.

#### Structure

The "Bolting sets" table has the following structure:

- "Connection point" column:  
Connector number in COMOS  
Up to four connectors are permitted.
- "BThk" column:  
Bolt thickness.  
E.g. for the height of the flange with the value: =S("vc25")
- "BType" column:  
Defines the bolt type.  
The following bolt types are permissible for AVEVA PDMS:
  - MACH (machine bolts)
  - STU (stud bolts)
  - CAP (cap screws)
  - JACK (jacking screws)
  - TAP (tapped holes)
  - BOLT (for general bolt types)

#### Mixed bolting

In the case of a mixed bolting, enter the additional bolt type and the bolt thickness in the extra columns.

### 3.2.2.11 Note "Not checked"

#### Configuration in COMOS DB

The COMOS DB is configured in such a way that the message "Not checked" is displayed when changes are made to the "GType" or "Source catalog" fields on the "E3D External 3D Interface" tab. This is to warn you that the changes may conflict with the rest of the displayed attributes.

**Not checked**

For example, the AVEVA PDMS attributes to be mapped are only updated when you select a new template in the "Cate" dropdown menu and not directly upon conversion of the GType.

#### Warn upon changes

If you want this warning to also be displayed in your database, activate the "On Change" script function in the corresponding attribute with the script.

```
Sub OnChange()  
'After editing the unit or the value  
  Set FillLib = CreateObject("ComosPDMS3D.Lib")  
  FillLib.CheckParam me  
End Sub
```

### 3.2.2.12 Answers for selectors

#### Introduction

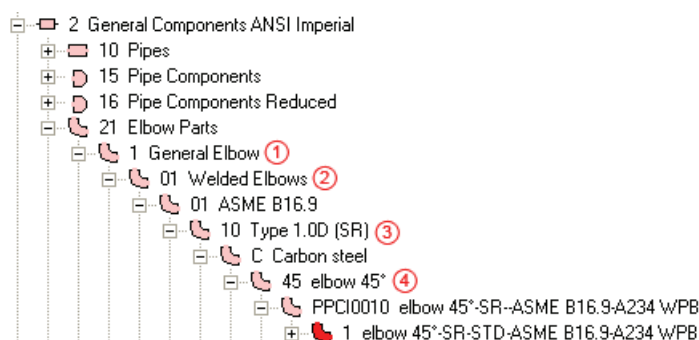
The sequence structures (questions) for the selectors are defined in the pipe spec. Information regarding this: see the link below.

However, the answers for the selectors are either defined in the PCC object or in the pipe spec element. Definitions in the PPC object are inherited automatically by its pipe spec elements.

You must define the answers for each pipe spec element individually in the pipe spec. Definitions of answers at pipe parts in the PPC offer the advantage that they only need to be handled at central points and this information is then inherited.

In order to define the answers for the selectors, create the desired attributes at the PPC object on the "E3D External 3D Interface" tab, on the relevant level in the Navigator structure. The name of the attribute must match the name of the selector.

## Example of pipe bends in the PPC



Level	Definition of answers (attributes) for the selectors
(1)	GType, Catalog and Cate are set for the first time.
(2)	STYP "ELBW" and SKEY "ELBW" are set.
(3)	RADI "1.0" is set.
(4)	ANGL Min "5" and Max "46" are set.

The attributes of the "External 3D Interface" tab that are required for AVEVA PDMS have been set as examples for the pipe parts in the PPC in the COMOSDB.

You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Selectors".

### 3.2.2.13 "VDM Data sheet" tab

#### Purpose

SCOM attribute references which are not the same for all SCOMs of a Cate are stored in the "V Nominal diameter dependent table" list attribute on the "Attributes > VDM Data sheet (DN1)" tab.

See also chapter Import template (Page 50).

A new column is created in the list attribute for each attribute that is not available yet. Each entry for an SCOM is then saved in a row of the list attribute.

#### Note

Since the list attribute can be extremely comprehensive if there are many objects, many entries only become visible upon using the scroll bars.

### 3.2.3 PipeSpec elements

#### 3.2.3.1 General

In the COMOS DB, the pipe spec elements are located in the base project under the node of the relevant pipe spec:

"@VIPER > @SPEC PipeSpec"

#### 3.2.3.2 "External 3D Interface" tab

##### Inheritance from PPC object

The attributes on the "ED3 External 3D Interface" tab have already been defined at the PPC object and are automatically inherited by the pipe spec elements. They should not be over-defined at the pipe spec element itself. Changes are not taken into consideration during the export. See Section "External 3D Interface" tab (Page 47) for information about the attributes.

Exception: Answers for the selectors can be created either at the PPC object or at the pipe spec element on the "E3D External 3D Interface" tab. See also section Answers for selectors (Page 60).

Selector attributes that are newly created at the pipe spec element and deleted by the interface (= "E3D External 3D Interface" tab) remain in the Navigator.

#### 3.2.3.3 "PDMS PipeSpec elements" tab

##### Purpose

The attributes that are specifically for the pipe spec elements are created on the "PDMSName PDMS pipe spec elements" tab and are inherited from the following node:

"@PDMS > @Y > @CHP > 03 > 02 > PDMSName PDMS pipe spec elements"

Among other things, the "PDMS pipe spec elements" tab specifies the name under which the individual pipe spec element from COMOS is to be "rolled out" (exported) to AVEVA PDMS, depending on its nominal diameter. The COMOS pipe spec element and the AVEVA PDMS objects resulting from it have a 1:N relationship.

### "Attributes valid for all elements" control group

The "Attributes valid for all elements" control group has the following attributes:

- "Detref":  
Reference to the detail text.  
Is initially taken from the "E3D" tab.
- "Detref owner":  
Owner of the detail text from the PPC.  
As a rule, that is the original base object (= PPC object). Can be reset via the buttons.
- "Matxt":  
The material text.  
Reference to the material catalog in COMOS.  
Is initially taken from the "E3D" tab. Can be reset via the buttons.
- "BltRef":  
Reference to the bolts.  
Only used for base objects of bolts. Is calculated internally.
- "Cmpref":  
Reference to components such as weight or volume.  
It is managed in AVEVA PDMS.

### "Attributes depending on nominal diameter" control group

The table in the "Attributes depending on nominal width" control group displays the exported, rolled-out pipe spec elements depending on nominal diameter.

Structure of the table:

- "PBOR1" column:  
First nominal diameter of the pipe spec element.
- "PBOR2" column:  
Next nominal diameter of the pipe spec element.
- "Name" column:  
Name of the "rolled-out" pipe spec element (SPCOMs)  
See also chapter Naming conventions (Page 67).
- "Status" column:  
Decides whether a pipe spec element is to be exported. If the field is blank (= 0), then the pipe spec element is exported, and otherwise it is not.
- "BltRef" column:  
Reference to the bolts.  
The columns for all other references are hidden and must be superimposed if required.

"Mask for name generation" field:

- Specify the rule according to which the AVEVA PDMS names of the pipe spec elements are generated. See also chapter Naming conventions (Page 67).
- Click the "Execute" button to apply rule changes. If you click on the "Test" button, you can first test in a separate window how the changes will affect the generation of the names before this is actually done. See also chapter Window for name generation (Page 69).

### "UDAs" control group

The UDAs (user-defined attributes) for the pipe spec elements are managed in the "UDAs" control group table.

Structure:

- "Name" column:

Name of the user-defined attribute. The name must be preceded by a colon, for example: ":Test".

- "Reference attribute / value" column:

See also chapter Mapping AVEVA PDMS attributes to COMOS (Page 52).

You can use the O-method to access the expression of another object:

- Form:
- =O ("Link attribute", "=S( ""Tab.Attribute"" )")
- Example:
- =O ("E3D.ABC", "=S( ""PDMSName.PDMSName.XYZ"" )")

---

#### Note

Take great care to enter the correct number of quotation marks!

---

"Unit name", "Unit label", "PDMS label for unit" columns:

See also chapter Mapping AVEVA PDMS attributes to COMOS (Page 52).

### Other fields

"Selector" field

This is used for the internal administration of selectors and stores the last selector within the selector structure. This way it is possible to distinguish between the various branches within a selector structure. You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Selectors".

This is set when the pipe spec element is created.



## 3.2.4 COCO tables

### 3.2.4.1 General

The CCTAs (connection compatibility tables) in AVEVA PDMS contain the individual COCO tables. The COCO tables themselves contain a list of all compatible connection types for pipe parts. If two connection types conflict with one another, the list is examined to determine how AVEVA PDMS should proceed.

To enable COMOS connection types to be assigned to those from the COCO table, connection types from COMOS must be translated into a form which is understandable for AVEVA PDMS.

Mapping of the connection types from COMOS to those from AVEVA PDMS is carried out either via the interface of AVEVA PDMS or by means of a COMOS standard table. See also chapter Mapping via the standard table (Page 65).

If a valid XML file for the COCO table has been specified on the "E3D External 3D Interface" tab, then COMOS uses this file for connector mapping. If a valid file has not been specified, the mapping is performed via the standard table.

### Connector types of the PPC object

The connector types for the SCOMs are created on the "E3D External 3D Interface" tab of the PPC object.

In the "E3D.PDMSMap.PDMS" list attribute of the "SCOM mapping" options group the connector types can be found under "Connection type PX". They usually contain a reference to the connector type on the "GD 3D-geometry" tab.

The entries for the connector types are replaced when exporting to PDMS.

### 3.2.4.2 Mapping via the standard table

Instead of assigning the COMOS connector types to PDMS connector types via the PDMS interface with the help of a COCO XML file, you can use a standard table in which the relevant assignment settings have been stored and which are automatically implemented when an export operation is carried out.

You find the standard table under:

"@3D > 00 > PDMS > COCO"

The list is generated automatically when the AVEVA PDMS interface is started. If the list does not exist yet, then it is created.

The standard table can be managed via the PDMS interface.

---

#### Note

If you use the standard table for the mapping, you should ensure that the standard table also includes all necessary connection type definitions for AVEVA PDMS. Maintain the standard table accordingly.

---

### Connector types

The table contains all connector types as combinations from all possible connector types and gasket types.

Columns of the tables:

- "Name":

Name of the connector type. Combination of the name of the connector type and the name of the gasket type in COMOS.

Format:

Name Connector type(\_Name Gasket type)

The names of the connector types are taken from the following standard table:

"@3D > 01 > BC > 02 Connection types"

The names of the gasket types are taken from the following standard table:

"@3D > 01 > 06 Contact faces"

- "Description":

Description of the combinations of connector types and gasket types. These are likewise read from the standard tables for the connector types and gasket types.

- "Value1":

Name for the connection type as it is exported to AVEVA PDMS. See also "Exported connector type" for information on the composition of the connector type.

- "Value 2":

Legend text from the AVEVA PDMS interface for the COCO table. This is not written until a save is made in the interface.

- "Value 3":

Coding for the legend color from the AVEVA PDMS interface for the COCO table. This is not written until a save is made in the interface.

### Exported connector type

The value in the "Value 1" column of the "COCO" standard table is composed as follows in the COMOS DB:

- Value of the "Value 6" column in the standard table for the connector type:

"3D > 01 > BC > 02 Connection types"

- Value of the "Value 6" column in the standard table for the gasket type:

"@3D > 01 > 06 Contact faces"

The "\*" character (asterisk) included in the value is always placed at the end. Each "\*" character (asterisk) is replaced during the export operation by the value of the "Value 6" column in the standard table for the nominal pressure of the PPC object:

"@3D > 01 > 04 PN-pressure nominal"

If no nominal pressure has been stored for an "\*" character, then no characters are exported for this character either.

You can also export fixed values for PPC objects, such as "GASK" for gaskets or "TUB" for tubes, for example.

## Example

A PPC object has, among other things, the following connector:

- Connection type: Flange end, contact face C  
-> Standard table "COCO", column "Name": 100\_110
- Pressure stage: PN 25
- Standard table "COCO", column "Value 1":  
Standard table "Connector type", column "Value 6", row "100": "F\*"  
+ Standard table "Contact face", column "Value 6", row "110": "B"  
= "FB\*"  
The "\*" character goes to the end in the value.
- Exported value:  
Without "\*" characters (asterisks) this value would be exported as it is. However, now the value has been supplemented by the value for the nominal pressure.  
-> Standard table "Nominal pressures", column "Value 6", row "PN25": "Q"  
= "FBQ"

## 3.2.5 Naming conventions

### 3.2.5.1 General

In order for referencing to be unique, the names of PDMS objects have to be unique. To achieve this, COMOS uses formulas which generate unique object names.

This applies in particular to the names for:

- Cates  
The formulas are defined on the "E3D External 3D Interface" tab.  
See also section Name generation for Cate (Page 70).
- SCOMs  
The formulas are defined on the "E3D External 3D Interface" tab of the PPC object or of the pipe spec element. The SCOMs that were rolled out are written to the "VDM.V" list attribute.  
If an SCOM attribute is saved in the data sheet, then its reference to the Cate is also saved in the "VDM.V.Catref Catref" list attribute of the data sheet.

- **Detrefs**  
The formulas are defined on the "E3D External 3D Interface" tab of the PPC object or of the pipe spec element. The Detrefs that are not dependent on the nominal diameter are written to the "VDM.V" list attribute.
- **PCOMs (=SPCOMs)**  
The formulas are defined on the "PDMSName PDMS PipeSpec elements" tab of the pipe spec element. The SPCOMs that are dependent on the nominal diameter are written to the "PDMSName.PDMSName" list attribute.
- **BTSE (bolting sets)**  
The formulas are defined on the "E3D External 3D Interface" tab of the PPC object or of the pipe spec element.

**See also**

Name generation for bolting sets (BTSE) (Page 72)

**3.2.5.2 Formula for name generation**

**Formulas which can be extended by users**

You can redefine the formula for name generation as you like, as long as the names that are generated are unique.

The following rules apply:

- Attributes need to be integrated as follows:

`%Tab.Attribute name%`

- Expressions such as Left, Right, Mid or the like are also permissible, and thus for example:

`Left(%Tab.Attribute name%,3)`

- Among others, the following constituent parts can be used for the formula:

Form elements	Description
%*UID%	Foreign SystemUID. 10-digit COMOS object number, unique per database
%UID%	Own SystemUID
%.NP%	Pipe spec pressure
%Name%	Own name
%ND1-0%	First nominal diameter: Description
%ND1-1%	First nominal diameter: Index
%ND1-3%	First nominal diameter: AVEVA PDMS label
%ND2-0%	Further nominal diameter: Description
%ND2-1%	Further nominal diameter: Index
%ND2-3%	Further nominal diameter: AVEVA PDMS label

### Further constituent parts of formulas

To find an overview of other possible constituent parts of a formula, proceed as follows:

1. Open the properties of a pipe spec element and go to the "Attributes > PDMSName PDMS pipe spec. element" tab.

2. Click the "Test" button.

Result: A window opens.

3. In the window, click on the "?" button.

Result: A context menu containing several options is displayed.

4. Select the required option.

### 3.2.5.3 Window for name generation

#### Purpose

You can view the current formula for name generation and modify it if you need to.

#### Opening the window for name generation

To open the name generation window, proceed as follows:

1. Open the properties of a pipe spec element and go to the "Attributes > PDMSName PDMS pipespec. element" tab.

2. Click the "Test" button.

Result: The name generation window opens.

#### Structure of the window

- "Default name" field:

Contains the rule (formula) according to which the pipe spec elements are named. See also chapter "Formula for name generation (Page 68)".

- "?" button:

Offers you additional attributes which you can integrate into the name formula so that the name is unique.

- "%\*UID% - object" field:  
See below. From the Navigator, you can drag the PPC object whose system UID is to be set in the formula for the value "%\*UID%".
- A preview table:  
The table has the following columns:
  - "ND1" column:  
First nominal diameter of the pipe spec element
  - "Current" column:  
Name that is produced by the formula previously used at the pipe spec element
  - "New" column:  
Name that is produced by the formula shown in the "Default name" field in the detail window and which can be modified in the window

### Modifying the formula

- In order to integrate an attribute into the formula, set the mouse pointer in the "Default name" field to the location at which the attribute is to be inserted.  
Then insert the desired formula manually or via the options of the "?" button.  
Ensure when inserting attributes or numbers that the existing parts of the formula are not split at the wrong point and hence possibly also made unusable.
- "%\*UID% - object" field: Drag the PPC object from the Navigator whose system UID is to be set in the formula for the value "%\*UID%".  
If the names of several pipe spec elements need to be as identical as possible, use a third-party UID rather than the own UID. This will be the case, for example, if several pipe spec elements are created although only one PPC object is dragged into the pipe spec:
  - Pipes  
You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Adding Pipe Part Components to the PDMS PipeSpec" and "Special case pipe (TUBE)".
  - Additional pipe spec elements (see section Automatically adding additional pipe parts to the PipeSpec (Page 57)).
- Reset the value in the "Default name" field to the default value by means of the "Reset" button.

#### 3.2.5.4 Name generation for Cate

##### Generate name

A unique name is generated for the Cate as soon as you lock the "Cate" field for the pipe spec element on the "External 3D Interface" tab. See also section Assigning an AVEVA PDMS template (Page 47).

In order to do this, a "GetDisplayValue" script function with the following script is executed at the "E3D.CATREF" attribute:

```
Function GetDisplayValue()
'Script function to determine the output value (DisplayValue) of the
attribute
'   Return value: String
GetDisplayValue = DisplayValue
If GetSpecOwner.SystemType = 13 Then
  If GetSpecOwner.CreateOption = 0 Then
    Set Locked = GetSpecOwner.spec("E3D.Locked")
    If Not Locked Is Nothing Then
      If locked.value = "1" Then
        Set GType = GetSpecOwner.spec("E3D.GTYPE")
        If Not GTYPE Is Nothing Then
          strDisplayValue = GTYPE.value
          If strDisplayValue <> "" Then
            GetDisplayValue = "/" & strDisplayValue &
              GetSpecOwner.SystemUID
          End If
        End If
      End If
    End If
  End If
End If
End Function
```

### Structure of the name

The name is generated from the following constituent parts:

- DisplayValue of the "E3D.CATREF" attribute
- Value of the "E3D.GTYPE" attribute
- System UID of the object

### AVEVA naming conventions

Modify the script according to the default to implement the AVEVA naming conventions for naming Cates. However, this is not necessary to ensure that the functional sequence works for the import and export operations.

### 3.2.5.5 Name generation for bolting sets (BTSE)

#### Algorithm for the name generation of bolting sets

COMOS automatically generates the names for the bolting sets from the following constituent parts:

- System UID or, if set, the "E3D.Name" attribute of the associated table with the flange dimensions, to be found below the following node:  
"@VIPER > @CatStd > PP > 31 Flange dimensions > ..."

---

#### Note

On the "E3D External 3D Interface" tab of the table you can also specify the catalog and the section for the export of the bolting sets, if these differ from the associated PPC object or pipe spec element.

---

- Number of bolts
- Bolt dimensions
- BType (see also section Bolts (Page 59))
- BThk (see also section Bolts (Page 59))

## 3.3 Using COMOS PDMS Integration

### 3.3.1 Installation

There are three installation modes: Local, Citrix server, and Citrix client.

- For the Local and Citrix client installation modes:  
Insert the installation CD and start "setup.exe".
- For installation on the Citrix server:
  - Insert the installation CD, open the control panel, click "Software", then "CD or Disk", and select the path to setup.exe. Then start "setup.exe".
  - The start dialog field of "Comos PDMS Integration 2.1 - InstallShield Wizard" opens.
  - Click "Next".



### 3.3.1.1 Local installation

#### Prerequisite

COMOS and AVEVA PDMS are already installed on the computer.

#### Selection of the installation mode

1. Start by selecting the "Local" installation mode:
2. Click "Next".
3. Specify the AVEVA PDMS directories:  
You have the option of changing the default settings for the AVEVA PDMS directories:
  - "PML-Lib":  
If the "Install PML-Lib" option is activated: The PML-Lib supplied with the installation CD is copied to this folder.
  - "Exchange (optional)":  
The path to the directory that will be used for data exchange. If no entries are made during setup: This directory can also be specified at a later time in COMOS in the project properties.
  - "Communication (optional)":  
The path to the directory that is used to exchange command files. If no entries are made during setup: This directory can also be specified at a later time in COMOS in the project properties.
  - "Executable (optional)":  
The path to the folder where "TalkToComos.exe" is saved. The Windows environment variable PATH should point to this directory. If not, the directory is attached to the Windows environment variable PATH.
  - "Install PML-Lib" option: See above.
4. Click "Next".
5. Start the installation  
Click the "Install" button to start the installation.
6. During installation, entries are made in the registry and the AVEVA PDMS-Lib is installed.
7. Click the "Finish" button to complete the installation.

### 3.3.1.2 Citrix Server

#### Prerequisite

COMOS must already be installed on the Citrix server.

#### Selection of the installation mode

1. Start by selecting the "Citrix server" installation mode:
2. Then click "Next".
3. Determining the AVEVA PDMS directories

In the next step, you can change the default settings for the AVEVA PDMS directories:

- "Exchange (optional)" field:

The path to the directory that will be used for data exchange. Must point to the same folder that is entered during the Client installation.

Input required.

- "Communication (optional)" field:

The path to the directory that is used to exchange command files. Must point to the same folder that is entered during the Client installation.

Input required.

4. Click "Next".
5. Starting the installation:  
Click the "Install" button to start the installation.
6. During the installation entries are made in the registry.
7. Click the "Finish" button to complete the installation.

### 3.3.1.3 Citrix Client

#### Prerequisite

COMOS must already be installed on the client's Citrix server.

#### Selection of the installation mode

1. Start by selecting the "Citrix client" installation mode:
2. Click "Next".
3. Determining the target directory

Determine the directory where the COMOS components on the client side of COMOS/PDMS communication are to be copied (COMOS side = the server side, AVEVA PDMS = the client side):

4. Click "Next".

5. Determining the AVEVA PDMS directories

In the next step, you can change the default settings for the AVEVA PDMS directories on the client computer:

– "PML-Lib":

If the "Install PML-Lib" option is activated: The PML-Lib supplied with the installation CD is copied to this folder.

– "Exchange (optional)":

The path to the directory that will be used for data exchange. Must be accessible from the Citrix server and must point to the same folder that was entered during the server installation.

– "Communication (optional)":

The path to the directory that is used to exchange command files. Must be accessible from the Citrix server and must point to the same folder that was entered during the server installation.

– "Executable (optional)":

The path to the directory where `TalkToComos.exe` is saved. The Windows environment variable PATH should point to this directory. If not, the directory is attached to the Windows environment variable PATH.

– "Install PML-Lib" option: See above.

6. Click "Next".

7. Start the installation

Click the "Install" button to start the installation.

8. During installation, entries are made in the registry and the AVEVA PDMS-Lib is installed.

9. Click the "Finish" button to complete the installation.

### 3.3.2 Preparing the COMOS database

The following section describes the requirements your COMOS database has to meet before you can start to configure the interface and before users can use the interface.

#### Base data

Import the following nodes from the COMOS DB:

- "@PDMS Interface" node

Import the node directly below the project root.

- "@System > @D > @Status > Comos <- XML -> PDMS" node

This node defines the status query which determines the status display after data matching.

- "@J > CA PDMS interface/construction assistant" tab

Copy this tab below the project structures you are using for projects in which you are using COMOS PDMS Integration.

#### Standard tables

Import the following node and all its subnodes from the COMOS DB:

"@3D > 00 > PDMS"

### 3.3.3 Project properties

#### Introduction

Before COMOS PDMS Integration can be used, you have to make and/or check a number of user-specific and, in some cases, project-specific settings in the project properties on the "PDMS interface/construction assistant" tab.

See also chapter "PDMS interface/construction assistant" tab (Page 177).

#### Purpose of the "PDMS interface/construction assistant" tab

The interface is configured in COMOS. To guarantee that COMOS and AVEVA PDMS use the same configuration, the configuration is saved in an initialization file (INI file). COMOS and AVEVA PDMS use this file for communication and data exchange between COMOS and AVEVA PDMS during an interface operation.

See also chapter INI file (Page 158).

## Procedure

To configure the "PDMS interface/construction assistant" tab, proceed as follows:

1. For each user: Click the "Select initialization file" button.
2. In the "Interface configuration profiles" window, set the paths to the individual directories by clicking the button to open the file explorer and selecting the required folder there.  
The path to the INI file is displayed in the "Initialization file" field.
3. For each user and project: Click the "Activate" button.

Result: The interface is activated.

- A script is saved in the COMOS user profile to ensure that COMOS PDMS Integration is activated automatically whenever the user opens the project.  
This means that amongst others the necessary context menus are activated.
- COMOS PDMS Integration remains active for the project until the user deactivates the interface for the project.

### 3.3.4 UDAs

To be able to use the PDMS interface, UDAs must be created in PDMS that contain the connection status, configuration information, and information on the connected COMOS object.

#### List with necessary UDAs

The following UDAs must be available in PDMS:

- ":ComosUID"  
Contains the SystemUID of the corresponding COMOS object.
- ":ComosCRefNo"  
Corresponds to the RefNo of the PDMS object.
- ":ComosStatus"  
Saves the object status.
- ":ComosName"  
This is the name of the corresponding COMOS object.
- ":ComosBaseOb"  
Saves the SystemFullName of the base object of the corresponding COMOS object.  
During an Export to Comos operation, this base object is used to create the corresponding COMOS object.
- ":ComosSClass"  
The name of the subclass to which this object belongs.

### Create the UDAs

Create the UDAs in a project-related dictionary database. If no appropriate database is available, create the database with the help of the "!!IFCreateDictDB" macro. The following parameters must be passed to this macro in the form of a string:

- "MDB": Name of the MDB
- "TEAM": Name of the TEAM
- "DictDB": Name of the to be created database
- "DBNumber": Database number (must be specified by the administrator)

This macro does not create an MDB, but a TEAM, insofar this is necessary. Once the database has been created, the macro sets this. If you do not create the database with the macro, set the specified dictionary database to currently active to subsequently create the UDAs in the Lexicon module.

Once you have switched to the Lexicon module, create the UDAs with the !!ITSetComosUDAs macro underneath the UWRL.

### 3.3.5 Creating an INI file

An INI file contains all the information required to run the AVEVA PDMS interface.

1. Specify the storage location for this file in an environment variable in the evars.bat file of your AVEVA PDMS installation. Define the environment variable as follows:

```
set COMOSPDMSINTERFACECONFIGURATION=<path>/<name of ini file>
```

2. Create an INI file containing at least the mandatory information. See also chapter INI file (Page 158).
3. Store the INI file in the directory you specified in the environment variable.

## 3.4 Configuration basics for interface operations

### 3.4.1 Interface objects

#### Definition

Interface objects are engineering objects which can be processed by the interface operations.

An object counts as an interface object if it has the following properties:

- In COMOS:
  - Assignment to a class
  - Assignment to a subclass
  - PDMS element type
- In AVEVA PDMS: UDA "":ComosSClass"

If the devices of your device catalog have these properties, they are processed by COMOS PDMS Integration.

### See also

Configuring COMOS interface objects (Page 137)

## 3.4.2 Classes and subclasses

### Definition

There are two groups of classes and subclasses:

1. Classes and subclasses which categorize interface objects
2. Classes and subclasses which are used to configure the execution of an interface operation

### Group 1

The following classes and their subclasses belong to this group:

- "Pipe":

Interface objects with this class are used for example as:

  - Pipe or pipe branch
  - HVAC
  - DUCT
  - CABLES
- "TaggedItem":

Interface objects with this class are equivalent to PDMS objects which can be positioned in 3D space.

The subclasses allow you to differentiate the behavior of interface objects of the same class. The subclass determines how an interface object is processed during an interface operation.

## Group 2

The following classes and their subclasses belong to this group:

- "Query" class and its subclasses: They are used to configure COMOS queries and make them accessible and executable in PDMS.
- "Document" class and its subclasses: A DocLink is generated for the interface object.

## Properties of classes

Classes have the following properties:

- Classes structure the subclasses.
- The number and name of classes are fixed. They are specified by the following standard table:  
"@3D > 00 > PDMS > 08 Classes"
- A class can have any number of subclasses.

## Properties of the subclasses

Subclasses have the following properties:

- The function performed by a subclass is determined by the class to which it belongs. See above.
- The number and names of subclasses are specified by the subclass definition objects and are dynamically saved in the following standard table:  
"@3D > 00 > PDMS > 08 > 01 Subclasses"

## See also

Class definition objects (Page 80)

Subclass definition objects of the "Pipe" and "TaggedItem" classes (Page 81)

Subclass definition objects of the "Query" class (Page 82)

Subclass definition objects of the "Document" class (Page 83)

### 3.4.3 Class definition objects

#### Function

Class definition objects are engineering objects which are located directly below the "@PDMSMAP" folder.



Class definition objects have the following function:

- They represent a class from the following standard table:  
"@3D > 00 > PDMS > 08 Classes"
- They structure the subclass definition objects.

## See also

Creating a class definition object (Page 105)

Standard tables for classes and subclasses (Page 99)

### 3.4.4 Subclass definition objects of the "Pipe" and "TaggedItem" classes

#### Function

The subclass definition objects for the "Pipe" and "TaggedItem" classes fulfill the following functions:

- They represent the subclasses of these classes in the engineering view.
- They are used to save and configure the properties of the corresponding members of this subclass.

#### Defining interface object properties

One subclass definition object defines the following properties for all interface objects of this subclass:

- Which PDMS element types is the interface object permitted to have?
- Which base object is used to create the interface object?

The user defines two base objects:

- One is used when the interface object is created in the unit tree
- One is used when the interface object is created in the location tree
- What are the structural characteristics of the interface object?
  - Is the PDMS interface object connected to a COMOS object from the unit tree or from the location tree?
  - Is the COMOS interface object to have a pointer to another COMOS object and is this object located in the unit tree or in the location tree?
- Regarding name mapping: How is the PDMS name of the interface object generated?
- Regarding name mapping: How does the interface in the interface operations area find the interface object in the COMOS tree structure?
- Regarding the name directory: Which values are evaluated by the interface operations when you process the interface object?

### 3.4 Configuration basics for interface operations

- Regarding the name directory: Which PDMS attribute is assigned to which COMOS attribute?
- Regarding the name directory: Which String Parameters are available in this subclass?
- Regarding the owner restriction rules: What conditions does the owner under which the interface object is created in PDMS have to meet?
- Which 3D model does the interface object use and how is the object initialized during the export if "Function" is set as design model?

#### See also

Classes and subclasses (Page 79)

Creating subclass definition objects (Page 105)

Configuring subclass definition objects of the "Pipe" and "TaggedItem" classes (Page 106)

Standard tables for classes and subclasses (Page 99)

### 3.4.5 Subclass definition objects of the "Query" class

#### Working with COMOS queries in PDMS

The interface allows users to work with COMOS queries in PDMS. Users have the following options:

- They select a query in PDMS from a range of predefined COMOS queries.
- Optionally: They define filters for the query.
- Optionally: They define the start object.
- They start the query from within PDMS.
- They see the result of the query in PDMS or in an external program such as Excel.

This functionality is implemented with the help of COMOS queries and subclass definition objects of the "Query" class.

The subclass definition objects of the "Query" class define the following:

- Which queries are available in PDMS?
- Which base objects do the objects in the result list have?
- Which is the predefined start object?
- In which output format is the result list is transferred?

**See also**

- Classes and subclasses (Page 79)
- Creating subclass definition objects (Page 105)
- Configuring COMOS queries from PDMS (Page 128)
- Standard tables for classes and subclasses (Page 99)

**3.4.6 Subclass definition objects of the "Document" class****Aim**

The interface allows users to create DocObjs for draft objects in COMOS.

Requirement:

- The draft objects have a DDNM attribute.
- The DDNM attribute references a design object which is connected to a COMOS object.
- A COMOS document under which the DocObjs are imported has been specified in the draft structure.

**Function**

In the subclass definition objects of the "Document" class you define all settings which are necessary to create DocObjs for draft objects.

These include:

- Where in the PDMS draft structure do you specify the COMOS document for which the DocObjs are created?  
As you are able create any number of subclass definition objects, you can configure the interface in such a way that "Import DocLinks" can be started at different levels of the draft structure according to the requirements.
- Are there filter criteria which restrict the COMOS document for which the DocObjs can be created?
- Are DocObjs only created for draft objects with names or also for draft objects without names?
- Are there owner restriction rules for the draft objects for which DocObjs are created?

**See also**

- Classes and subclasses (Page 79)
- Creating subclass definition objects (Page 105)
- Configuring "Import DocLinks" (Page 132)
- Standard tables for classes and subclasses (Page 99)

### 3.4.7 "@PDMSMAP" folder

#### Function

The "@PDMSMAP" folder fulfills the following functions:

- The "@PDMSMAP" folder is where you make general settings which apply for the entire interface and not only for a specific class or subclass.
- The class and subclass definition objects are managed below the folder.
- Use the context menu of the folder to save the configuration of the subclass definition objects in a file and refresh the standard table used to save the subclasses.

#### Overview of the general interface settings

The following general interface settings are configured in the "@PDMSMAP" folder:

- After which interface operations are feedback signals displayed to users?
- Is site mapping and zone mapping used, and how are they configured?
- How is character mapping configured?
- How is units mapping configured?
- Have pre/post executables been registered?
- Which global String Parameters are there?

#### See also

Creating the "@PDMSMAP" folder (Page 99)

Configuring the "@PDMSMAP" folder (Page 100)

Synchronizing settings (Page 135)

Operation messages (Page 93)

Site mapping and zone mapping (Page 93)

Character mapping (Page 92)

Units mapping (Page 92)

Pre/Post executables (Page 92)

String Parameters (Page 89)

### 3.4.8 Name mapping

#### Function

Name mapping fulfills the following functions:

- It generates a PDMS name for a COMOS interface object.  
The interface uses the PDMS name to identify the PDMS interface object to which the COMOS object is connected.
- It uses the PDMS name of a PDMS interface object to ascertain the path to the COMOS interface object to which the PDMS interface object is connected.
- If the structural behavior of the subclass definition object prescribes that a COMOS interface object references another COMOS object, name mapping derives the referenced object from the PDMS name.
- Optionally: If the COMOS interface object which is to be connected to a PDMS object does not exist yet, name mapping determines the base object used to create it.
- If the owner structure of the COMOS interface object which is to be connected to a PDMS object is incomplete or missing, name mapping specifies which owners are created.

#### Implementing name mapping

Name mapping is defined for the subclass definition objects in two tables: Unit mapping and Location mapping.

You find information how the structure of the tables, their configuration, and the algorithm on which name mapping is based in section Configuring unit and location mapping in name mapping (Page 119).

#### Unit mapping and location mapping

The unit tree and the location tree do not usually share the same object structure. This is why it is necessary to manage two mapping tables:

- Unit mapping: Table which defines name mapping in the unit tree
- Location mapping: Table which defines name mapping in the location tree

#### See also

Structural behavior (Page 87)

Level rules of name mapping (Page 86)

### 3.4.9 Level rules of name mapping

#### Purpose

You can define deviations from the standard evaluation model for unit mapping and location mapping.

These deviations are particularly required in the following cases:

- If your project works with an alias structure
- If your project works with different hierarchy depths

#### Rule "Take alias rather than object"

If this rule is activated, the alias structure is used to generate the PDMS names.

If no alias is available, the name or description of the object is used.

#### Rule "Skip level if data part empty"

If this rule is activated, you can map hierarchies of different depths in name mapping. The maximum number of mapping entries in the name mapping is determined by the hierarchy with the deepest level.

If the evaluation of the data item or fixed name of a mapping entry returns an empty string, the process continues with the next mapping entry.

Application example: Your equipment is located in the "Equipment" folder. Only pumps are not directly located in the folder; these are to be found in its "Pumps" subfolder.

#### Rule "Skip level if unable to create object"

If a COMOS object cannot be created because of its configuration - because the potential owner only permits certain elements as subobjects, for example - name mapping jumps to the next mapping entry rather than aborting the process. The entry is processed with the next PDMS name part.

Result: This rule allows you to use name mapping in COMOS to create object structures with hierarchy levels of different depths.

#### Rule "Use label for fixed name"

If this rule is activated, the label of a COMOS object rather than its name is used in the "Fixed name" range.

#### See also

Configuring level rules (Page 127)

Name mapping (Page 85)

### 3.4.10 Structural behavior

#### Purpose

The structural behavior of a subclass definition object defines the following points:

- Creation mode: Is the COMOS interface object to be connected to the PDMS object located in the unit tree or in the location tree?
- Assign mode: Does the COMOS interface object reference another COMOS object and is the referenced object located in the unit tree or in the location tree?

Depending on how you define the structural behavior, either unit mapping or location mapping is evaluated during name mapping to fulfill one of the following tasks:

- Find or create an interface object in order to connect it with a PDMS object.
- Find or create the COMOS object referenced by the COMOS interface object.

#### See also

Name mapping (Page 85)

LocObj (Page 87)

Defining structural behavior (Page 107)

### 3.4.11 LocObj

#### Definition

A LocObj is an object which is generated whenever an interface operation processes an object whose subclass has the following structural behavior:

- Creation mode: "Create in location tree"
- Assign mode: "Assign to object in unit tree"

The aim of this setting is to map multiple PDMS objects to a single COMOS object. For this purpose, the LocObjs level is inserted between the PDMS objects and the COMOS object.

#### Application case

A pipe runs across different zones in PDMS, meaning that there are several pipe objects, but logically these represent the same pipe. There is only one pipe object in the COMOS unit tree.

LocObjs are created for the PDMS pipes in the location tree and connected to the PDMS pipes. The LocObjs get a reference to the same COMOS pipe object.

## Properties

LocObjs have the following properties:

- The LocObj is connected to the PDMS object.
- The LocObj gets a unit pointer to an interface object from the unit tree.
- The full name of the corresponding PDMS object is written to the "Name" property of the LocObj.

In contrast, only the last name part of the PDMS name is written to the name of the referenced unit object. Depending on how the name mapping is configured, the name part can even be truncated before it is written to the "Name" property.

- The data exchange regulated by the String Parameter is always carried out from connected object to connected object, hence, between LocObj and PDMS object.

## See also

Structural behavior (Page 87)

### 3.4.12 Name directory

#### Definition

The name directory is a table in which each line defines a String Parameter.

Section String Parameters (Page 89) contains detailed information on String Parameters.

#### Evaluation

The name directory is evaluated in the following cases:

- During an interface operation if values are compared, exported or imported. These are usually attribute values.
- If a String Parameter is used in a subclass definition object on the "Name mapping" tab to generate one of the following components of a PDMS name part:
  - Prefix
  - Postfix
  - Data item
  - Base object
- If a String Parameter is used in a subclass definition object on the "Name mapping" tab to generate a fixed component of a name.
- If objects are filtered in the "Export to Comos" PDMS mask.
- If a String Parameter is transferred to a PML function as a parameter. For example, if a String Parameter is used in an initialization function in a subclass definition object on the "Model" tab.



### 3.4.13 String Parameters

#### Definition

A String Parameter is a variable which returns or processes a string:

- It reads a COMOS value or a fixed string. If the String Parameter is configured accordingly, it writes the value to PDMS.
- It reads out a PDMS value or a fixed string. If the String Parameter is configured accordingly, it writes the value to PDMS.

#### Availability

There are two types of String Parameters. They differ with regard to their availability:

- Global String Parameters:

These String Parameters are available everywhere in the interface operations area. They are defined in the "@PDMSMAP" folder.

In PDMS, global String Parameters are evaluated independently of the current element.

In COMOS, they are evaluated at the following objects:

- In the project
- In the workset
- In the "@PDMSMAP" folder

- Local String Parameters:

These String Parameters are only available per subclass. They are defined in the subclass definition objects and evaluated at the interface object.

#### Purpose

Global String Parameters are used for the following purpose:

- To define variables which are globally available in the interface operations area.
- To prevent interface operations from being executed if a COMOS value and a PDMS value defined by the String Parameter do not match.

Local String Parameters are used for the following purpose:

- To define variables which are locally available in a subclass
- To perform one of the following actions during the interface operations:
  - Export: To write a COMOS value to PDMS
  - Import: To write a PDMS value to COMOS
  - Check: To compare a COMOS value with a PDMS value

Without these actions it is not possible to make an assignment between the attributes of an interface object in COMOS and PDMS.

- Optionally: To stop objects from being created for the export if a COMOS value you select at random and a PDMS value you select at random do not match.

### See also

Name directory (Page 88)

Working with String Parameters in the name directory (Page 112)

Defining global variables (Page 100)

## 3.4.14 Owner restriction rules

### Definition

You can use the owner restriction rules to define company internal or project-specific rules at the subclass definition objects which determine which owners are permitted in PDMS for objects of this subclass.

The interface evaluates the owner restriction rules before it exports an object to PDMS or makes an assignment between a COMOS object and a PDMS object. If an owner violates one of the rules, the object is not exported or no assignment is made.

### Example

The owner restriction rules allow the following rules to be put in place, for example:

- In PDMS, the exported interface objects are only created under zones whose purpose attribute is set to "TAG".
- In PDMS, pipes and branches are only created under zones whose purpose attribute is set to "PIP".
- In PDMS, the equipment is only created under zones whose purpose attribute is set to "EQU".

### See also

Configuring owner restriction rules (Page 109)

### 3.4.15 Model information

#### Four model types

The following model types are available for interface objects:

- Design template:
  - The model is defined via a design template. The template has been imported from PDMS to COMOS.
  - The user defines one main template and up to five templates for secondary equipment.
- PDMS element:

The model of an existing PDMS element is used.
- Function:

The model is defined via a PML function.
- None:

A box is created.

#### Possible sources

There are three ways of defining which model type an interface object uses:

- The model is inherited from the base object
- The model is taken from the subclass definition object
- The settings inherited from the base object or the subclass definition object are over-defined in the properties of the interface object.

#### Export

When an interface object is exported to PDMS, the model information defined in COMOS is also exported.

The object is displayed in PDMS with the corresponding model.

#### See also

Configuring the model (Page 111)

Overwriting the inherited model information (Page 138)

### 3.4.16 Pre/Post executables

#### Definition

Pre/post executables are user-defined PML functions which are called prior to starting or after completion of an interface operation.

- Pre executables execute steps serving the purpose of initialization and have an effect on the entire interface operation.
- Post executables execute concluding steps.

#### See also

Using pre/post executables (Page 101)

### 3.4.17 Character mapping

#### Maintaining the naming convention

Some characters, such as blanks, are not allowed to form part of a PDMS name.

If name mapping generates a PDMS name from within COMOS, the resulting string may contain one or more characters which are not permitted in PDMS.

You define which characters are to replace these illegal characters in the character mapping settings in the "@PDMSMAP" folder.

#### See also

Using character mapping (Page 102)

### 3.4.18 Units mapping

#### Unit conversion

It may be that an attribute was assigned a different unit in COMOS than in PDMS.

Units mapping enables you to map these units. When an attribute of this type is exported or imported, the attribute value is converted as defined in the units mapping settings.

#### See also

Using units mapping (Page 103)

### 3.4.19 Site mapping and zone mapping

#### Initial situation

Via the owner restriction rules you have the option to specify that interface objects of a subclass in PDMS are only allowed to be located below sites or zones.

When the "Export to CE" interface operation is called, the interface exports the objects of this subclass under the current element.

When the "Export" interface operation is called, the interface does not initially know the zone and site under which these objects are to be exported.

#### Definition

Site mapping and zone mapping involve two mapping tables:

- Site mapping returns a string: the PDMS name of a site
- Zone mapping returns a string: the PDMS name of a zone

The interface carries out site and zone mapping in order to determine the site and zone under which the objects in the selection set are to be exported. If the interface cannot find the site and the zone in PDMS, the interface object is not exported.

#### Condition

Site mapping and zone mapping are only carried out in the following case:

- Site mapping and zone mapping are activated.
- The "Export" interface operation is called.
- The subclass definition object of the interface object which is being processed stipulates that the object has a zone or a site as its owner in PDMS.

#### See also

Using site mapping and zone mapping (Page 104)

### 3.4.20 Operation messages

#### Definition

Short operation messages are available for the following interface operations:

- "Assign"
- "Export" from PDMS and COMOS
- "Refresh"
- "Unassign"

If the administrator activates an operation message for these operations, a window opens once the interface operation is complete which contains information regarding the course of the operation for the user.

**See also**

Activating operation messages (Page 100)

**3.4.21 Connection information**

**Definition**

A COMOS interface object and a PDMS interface object are connected when their connection information matches.

COMOS object		PDMS object
"SystemUID"	=	":ComosUID"
"E3D.PDMSRefNo"	=	"RefNo"

COMOS object		COMOS object
"SystemUID"	=	"E3D.ComosCheckUID"

PDMS object		PDMS object
"RefNo"	=	":ComosCRefNo"

These properties are set during the following interface operations:

- PDMS and COMOS: "Export"
- COMOS: "Export to CE"
- COMOS: "Assign"

**Name matching**

The fact that name mapping of a COMOS object generates a name which matches the name of a PDMS object is not sufficient for the two objects to count as connected.

Still, name matching is the assignment criterion for the "Assign > Match Names" interface operation.

## 3.5 Configuring interface operations

### 3.5.1 Workflow

#### Two approaches

Either of the two following approaches outlined below are possible for the administration of the interface operations area:

- Task-based administration:
  - You start by only making the settings which are necessary for a specific task.
  - Then you check your entries by executing the corresponding interface operations.
  - After this you configure the next task, and so on.
  - You should use task-based administration if you do not have extensive experience of interface administration.
  - The following tables describe the task-based approach.
- Object-based administration:
  - You start by creating the "@PDMSMAP" folder and, underneath it, the class definition objects and the subclass definition objects.  
The interface objects are or have been created by users during project planning.
  - Afterwards, you configure the objects: First the "@PDMSMAP" folder, then the subclass definition objects, then the base objects of the interface objects.
  - You should only use object-based administration if you have extensive experience of interface administration.
  - The structure of the "Administration" section focuses on object-based administration.

#### Workflow for task-oriented administration

The following tables describe the workflow for task-based administration of the interface operations area:

- Table 1: The basic configuration (configuration of the "@PDMSMAP" folder, the subclass definition objects of the "Pipe" and "TaggedItem" classes, and the interface objects)
- Table 2: Use of COMOS queries in PDMS
- Table 3: Use of pre/post executables
- Table 4: Importing DocLinks
- Table 5: Importing AVEVA templates

Steps marked with "\*" must be executed at this point at the latest, but can also be performed beforehand.

Table 3- 1 Table 1: Workflow for the basic configuration

Step	To be configured element	Description
1	Interface objects	Add the "E3D" tab to the base data.
2	"PDMS element types" standard table	Optionally Enter more element types.
3	"@PDMSMAP" folder	Create the "@PDMSMAP" folder.
4	Class definition objects	Create the class definition objects.
5	Subclass definition objects	For each class definition object: Create the subclass definition objects.
6	Subclass definition objects of the "Pipe" and "TaggedItem" classes	For each subclass definition object: <ul style="list-style-type: none"> <li>• Defining the structural behavior</li> <li>• Defining the PDMS element types</li> <li>• Owner restriction rules:                             <ul style="list-style-type: none"> <li>– Define the element type of the owner</li> <li>– Optionally: Define more restriction rules</li> </ul> </li> <li>• Optionally: Define local variables in the name directory</li> <li>• Define unit mapping and location mapping</li> <li>• Optionally: Define the level rules</li> <li>• Define the model information</li> </ul>
7	Interface objects	Configure the "E3D" tab in the base data
8*	"@PDMSMAP" folder	This can already be done after step 3 <ul style="list-style-type: none"> <li>• Define character mapping</li> <li>• If you plan to use "Export", not only from "Export to CE": Define site mapping and zone mapping</li> <li>• Optionally: Defining global variables</li> <li>• Optionally: Activate the operation messages</li> </ul>
9	"@PDMSMAP" folder	Synchronize the settings between COMOS and PDMS
10	Test settings: <ul style="list-style-type: none"> <li>• In COMOS:                             <p>Run "PDMS &gt; 3D-View &gt; Select &gt; By Name":</p> <ul style="list-style-type: none"> <li>– If the object is found in PDMS: Run "PDMS &gt; Assign&gt; Selected object".</li> <li>– If the object is not found in PDMS: Run "PDMS &gt; Export" and check the generated name in PDMS.</li> </ul> </li> <li>• In PDMS:                             <p>Run "Comos &gt; CE &gt; Navigate".</p> </li> </ul>	
11*	"@PDMSMAP" folder	This can even be carried out after step 3. Define units mapping.
12*	Subclass definition objects of the "Pipe" and "TaggedItem" classes	This can already be done after step 5 In the name directory, define how values are mapped to one another in COMOS and PDMS.
13	"@PDMSMAP" folder	Synchronize the settings between COMOS and PDMS.



Step	To be configured element	Description
14	Use the following interface operations to test the settings in COMOS: <ul style="list-style-type: none"> <li>• "Refresh"</li> <li>• "Custom Refresh"</li> <li>• "Check status"</li> <li>• "Export" (with attribute values)</li> </ul>	
17*	Subclass definition objects of the "Pipe" and "TaggedItem" classes	This can already be done after step 6. "General" tab: Specify the base objects for creating objects.
18	"@PDMSMAP" folder	Synchronize the settings between COMOS and PDMS.
19		Use the "Export" interface operation to test the settings in PDMS.

Table 3-2 Table 2: Workflow for the use of COMOS queries in PDMS:

Step	To be configured element	Description
After step 5 in the basic configuration or later		
1	Queries	Create and configure the main queries that are called from within PDMS. If applicable, create and configure start queries for the main queries.
2	Subclass definition objects of the "Query" class	Configure the subclass definition objects.
3	"@PDMSMAP" folder	Synchronize the settings between COMOS and PDMS.
4		Test the settings in PDMS in the "Query Comos data" window.

Table 3-3 Table 3: Workflow for the use of pre/post executables

Step	To be configured element	Description
After step 3 and prior to step 10 of the basic configuration		
1	PML functions	Declare and implement PML functions.
2	"@PDMSMAP" folder	Register PML functions from step 1 as pre/post executables.
3	"@PDMSMAP" folder	Synchronize the settings between COMOS and PDMS.
4		Test the settings by executing the interface operation for which you registered a PML function as a pre/post executable.

Table 3- 4 Table 4: Importing DocLinks

Step	To be configured element	Description
After step 10 of basic configuration, since the connection between COMOS objects and PDMS objects exists at this point in time.		
1	Draft structure in PDMS	Specify the attribute or the UDA for the name of the COMOS document.
2	Subclass definition objects of the "Document" class	Configure
3	Test the settings as follows: <ul style="list-style-type: none"> <li>• Create a document in COMOS</li> <li>• Enter the name of the document in PDMS in the attribute defined in step 1 or the UDA</li> <li>• Run the "Import DocLinks" operation in PDMS</li> </ul>	

Table 3- 5 Table 5: Importing AVEVA templates

Step	To be configured element	Description
If you are using design templates to generate the model: Before step 6 of basic configuration.		
1	Base object node "@PDMS > @TPL"	Call the "Update design templates" command from the context menu of the node.
2	GType folder	Assign the GType folder references created in the base data in step 1 to the device catalog.

### 3.5.2 Maintaining the standard table for PDMS element types

#### Purpose

The following standard table is supplied with the COMOS DB:

"@3D > 00 > PDMS > 14 PDMS element types"

The most important PDMS element types are listed in the standard table.

You can expand the standard table:

- By adding missing PDMS element types
- PDMS Version 12 and higher: By means of user-defined PDMS element types

#### See also

Structure of the standard table for PDMS element types (Page 164)

### 3.5.3 Standard tables for classes and subclasses

#### System-internal management

The standard tables in which classes and subclasses are listed are managed system-internally in the engineering projects. This means that you are not permitted to edit the standard tables in the engineering project.

You may edit the standard table in the base project.

#### Standard table for classes

You find the standard table in the following node:

"@3D > 00 > PDMS > 08 Classes"

#### Standard table for subclasses

You find the standard table in the following node:

"@3D > 00 > PDMS > 08 > 01 Subclasses"

The standard table is dynamically updated: When you call the "PDMS > Save subclasses" context command at the "@PDMSMAP" folder, COMOS synchronizes the structure of the subclass definition objects underneath the class definition objects with the entries of the standard table.

#### See also

Structure of the standard table for classes (Page 165)

Structure of the standard table for subclasses (Page 165)

### 3.5.4 Creating the "@PDMSMAP" folder

#### Procedure

Create the "@PDMSMAP" folder on the "Units" tab in the Navigator.

The folder is usually created directly underneath the project root. Depending on the structure of your project, you might also be able to create the folder further down in the tree structure.

Depending on how your project structure is configured, create the folder by selecting the "New" command from the context menu or dragging it from the base data to the "Units" tab.

If you create the folder using drag&drop from the base data, enter the following value in the "Name" property: "@PDMSMAP"

Write the name in capital letters.

### Base object in the base data

You find the base object of the "@PDMSMAP" folder in the following location in the COMOS DB: "@PDMS > PDMSMAP Folder for class/subclass definitions"

### See also

Configuring the "@PDMSMAP" folder (Page 100)

"@PDMSMAP" folder (Page 84)

## 3.5.5 Configuring the "@PDMSMAP" folder

### 3.5.5.1 Activating operation messages

#### Procedure

To activate/deactivate operation messages, proceed as follows:

1. Open the properties of the "@PDMSMAP" folder.
2. Select the "Attributes > General" tab.
3. Search for the "Operation messages" control group.

The control group contains a table with the following columns:

- "Operation" column: Interface operations for which operation messages are available
- "Show information" column To activate/deactivate operation messages

4. Right-click a row in the "Show information" column.

A list is displayed.

Possible values:

- "Yes": Activates the operation message for the corresponding operation based on the row you selected.
- "No": The operation message is deactivated.

### See also

Operation messages (Page 93)

### 3.5.5.2 Defining global variables

#### Structure of the "Name directory" table

The global variables are defined in the "Name directory" table. The table structure is virtually identical to the one for the subclass definition objects.

Differences:

- No COMOS unit
- No SetFunction
- Only one flag: "Rule"
- Name of the table: "GlobNameDir"

As no exchange of values is defined in the name directory of the "@PDMSMAP" folder, only global variables, there is no SetFunction and only the "Rule" flag.

## Procedure

You define global variables in the "@PDMSMAP" folder, on the "Name directory" tab.

When defining global variables, proceed in exactly the same way as when defining a local variable in a String Parameter at the subclass definition objects.

## Using the "Rule" flag

If you activate the "Rule" flag, you have to configure the String Parameter in such a way that it returns both a COMOS value and a PDMS value.

Result: The interface compares the two values before every interface operation and only executes the operation if the values are identical.

## See also

"Name directory" tab (Page 167)

Working with String Parameters in the name directory (Page 112)

### 3.5.5.3 Using pre/post executables

## Procedure

To execute a PML function before an interface operation is executed or after it has been completed, proceed as follows:

1. Declare the PML function.
  - Signature:

```
!!NamePMLFunktion(InterfaceOp is String, arg1 is String, arg2 is String, ...)
```
  - Parameter `InterfaceOp`: The name of the interface operation before or after which the PML function is executed
  - Parameter `arg1, arg2, ...`: Name of any number of global String Parameters, optional
2. Implement the PML function.
3. Open the properties of the "@PDMSMAP" folder.

4. Go to the "Attributes > Pre/Post executables" tab.
5. If the PML function is executed before the start of the interface operation, register it in the "Pre executables" table.

If the PML function is executed after completion of the interface operation, register it in the "Post executables" table.

To register a PML function for an interface operation, proceed as follows:

- "Executable" column:  
Enter the name of the PML function from step 1.
- "Query" column:  
Select "Yes" if the PML function is to execute a COMOS query. The function must have been implemented accordingly.  
Select "No" if the PML function does not use a COMOS query.
- "Export", "Assign", "Refresh", "Check status", "Import", "MTO import" columns:  
Select "Yes" to register the PML function for these interface operations.

You can find additional information on this topic in the "3D Integration Operation" manual, keyword "List of interface operations and internal names".

#### See also

"Pre/post executables" tab (Page 168)

Pre/Post executables (Page 92)

#### 3.5.5.4 Using character mapping

##### Procedure

To define those characters which are permitted in COMOS but are illegal in PDMS and are replaced during name mapping, proceed as follows:

1. Open the properties of the "@PDMSMAP" folder.
2. Select the "Attributes > Character mapping" tab.
3. Configure the table which you find on this tab.

For each line, define an illegal character along with the character which is to replace the illegal character in PDMS. Proceed as follows:

- "Comos" column: Enter the illegal character.
- "PDMS" column: Enter the character which is to replace the illegal character.
- Enclose the characters in single quotation marks.

## See also

"Character mapping" tab (Page 168)  
Character mapping (Page 92)

### 3.5.5.5 Using units mapping

#### Procedure

To define how units are mapped to one another in PDMS and COMOS if a different unit has been assigned to an attribute in COMOS than in PDMS, proceed as follows:

1. Open the properties of the "@PDMSMAP" folder.
2. Select the "Attributes > Character mapping" tab.
3. Configure the table which you find on this tab:
  - "Name" column: The name of the mapping entry  
The name is displayed in the name directory in the "Unit" column.
  - "Unit in PDMS" column: The PDMS unit which the value has or is to have in PDMS.  
During the import/export there is no explicit check whether the attribute actually uses the unit specified in the units mapping in PDMS. The interface assumes that the correct unit has been entered or transferred.
  - "Unit group in Comos" column: The unit group to which the unit entered in "Unit in Comos" belongs.
  - "Unit in Comos" column: The COMOS unit which the value has or is to have in COMOS.

#### Result

- During the import the interface converts the value transferred from PDMS from the unit specified in the "Unit in PDMS" column into the unit specified in the "Unit group in Comos" and "Unit in Comos" columns.
- During the export the interface converts the value transferred from COMOS from the unit specified in the "Unit group in COMOS" and "Unit in COMOS" columns into the unit specified in the "Unit in PDMS" column.
- The mapping entries defined here are available in the name directory.

## See also

"Units mapping" tab (Page 168)  
Units mapping (Page 92)

### 3.5.5.6 Using site mapping and zone mapping

#### Procedure

To configure site mapping and zone mapping, proceed as follows:

1. Open the properties of the "@PDMSMAP" folder and select the "Attributes > General" tab.
2. Enable the "Use site and zone mapping" option in the "General properties" control group.

---

#### Note

##### Deactivation locks the "Export" menu

If you deactivate the option and the subclass of an interface object prescribes that the object is located under a site or a zone in PDMS, the "PDMS > Export" command will be locked in the context menu of the COMOS interface object.

---

3. Site mapping: Switch to the "Site mapping" tab.
4. Zone mapping: Switch to the "Zone mapping" tab.
5. Configure the mapping table. The configuration is virtually identical to that for unit mapping and location mapping at the subclass definition object.

There are the following differences:

- Configure the table in such a way that the mapping entries generate the name of a site or zone
- Mapping is only performed for one direction: It generates the PDMS name of a site or zone.
- There are no level rules.
- The structural behavior is irrelevant.

#### See also

"Site mapping" and "Zone mapping" tabs (Page 169)

Configuring unit mapping and location mapping (Page 124)

Site mapping and zone mapping (Page 93)



### 3.5.6 Creating a class definition object

You can only create one class definition object for each class.

#### Procedure

To create a class definition object, proceed as follows:

1. Click the "Units" tab in the Navigator.
2. Select the "@PDMSMAP" folder in the Navigator.
3. Select the "New" command from the context menu and select the required class.

If you have already created a class definition object for the class, the class will no longer be available in the context menu.

#### Assign a class to the class definition object

A class definition object is assigned to a class via the name of the class definition object.

When creating the object, COMOS automatically enters a name - depending on which class you selected under "New" in the context menu.

You are not permitted to change the name of the class definition object.

#### Defining the properties of the class

You define the subclass for the class by creating subclass definition objects under a class definition object.

### 3.5.7 Creating subclass definition objects

#### Procedure

To create a subclass definition object, proceed as follows:

1. Click the "Units" tab in the Navigator.
2. Open the "@PDMSMAP" folder in the Navigator.
3. Select the class definition object whose name matches the name of the class which is to have the subclass definition object.
4. Select the "New > SUBCLASS\_..." command from the context menu.
5. Open the properties of the subclass definition object.
6. In the "Name" property, enter the name of the subclass which represents the subclass definition object.
7. Enter a meaningful description in the "Description" property.

## Result

- The subclass definition object is created.
- The subclass definition object is automatically assigned to the class that is entered as the name of its class definition object.

## Adding a subclass to a standard table

Subclasses which are represented by the subclass definition object can be added to the standard table for subclasses. See also section Standard tables for classes and subclasses (Page 99).

## 3.5.8 Configuring subclass definition objects of the "Pipe" and "TaggedItem" classes

### 3.5.8.1 Defining PDMS element types

#### Procedure

To define which PDMS element types are permitted for interface objects of this subclass, proceed as follows:

1. Open the properties of the subclass definition object.
2. Switch to the "Attributes > General" tab.
3. Table "PDMS element types", column "Name":
  - Click on a field in the "Name" column.  
Result: A list is displayed in the field.
  - Open the list and select the required element type from the list.  
Result: The type is entered in the field. COMOS automatically adds a new line to the bottom of the table.
4. Repeat step 3 until you have entered all required element types.

#### Result

Only PDMS element types defined in this way are available to the user in the interface objects of this subclass.

### 3.5.8.2 Defining structural behavior

#### Procedure

To define the structural behavior of an interface object of this subclass, proceed as follows:

1. Open the properties of the subclass definition object.
2. Switch to the "Attributes > General" tab.
3. "Structural behavior" control group, "Creation mode" list:

Define whether the COMOS object connected to the PDMS interface object is located in the unit tree or in the location tree.

Possible values:

- "Creation in unit tree": Unit mapping is performed to find or create the COMOS interface object.
- "Creation in location tree": Location mapping is performed to find or create the COMOS interface object.
- "None": You find information on this in the reference part of this document, section "General" tab (Page 166).

4. "Structural behavior" control group, "Assign mode" list:

Define whether the COMOS interface object references another COMOS object and if the referenced object is located in the unit tree or in the location tree.

The values that are offered to you are determined by which "Creation mode" you have set.

Possible values:

- "Assign to object in location tree" The COMOS interface object gets a location reference. The referenced object is determined by means of location mapping.
- "Assign to object in unit tree": The COMOS interface object gets a unit reference. The referenced object is determined by means of unit mapping.
- "None": The interface object gets no reference.

#### See also

LocObj (Page 87)

### 3.5.8.3 Defining the base object for the creation of interface objects

#### Possible alternatives

There are three ways of defining which base object is used to create an interface object in COMOS:

1. Specify the base object on the "General" tab of the corresponding subclass definition object.
2. Specify the base object by mapping the name of the corresponding subclass definition object.

This is the method used in this document.

3. Enter the SystemFullName of the base object in the following UDA ":ComosBaseOb" of the PDMS interface object.

This method offers you the most flexibility but it is also the most complex, since the UDA is set manually for each PDMS interface object.

#### Procedure

To define which base object is used to create the interface objects of this subclass in COMOS, proceed as follows:

1. If PDMS interface objects of this subclass already exist: Make sure that the ":ComosBaseOb" UDA of the PDMS interface object is empty. If not, the interface will use the base object that is entered there.
2. Open the properties of the subclass definition object.
3. Switch to the "Attributes > Name mapping" tab.

Make sure that no base object has been defined for the mapping entry on the far right in the unit mapping or location mapping.

4. Switch to the "Attributes > General" tab.

5. "Base objects for creating devices" control group, "Creation in unit tree" reference:  
Using drag&drop, specify which base object is used during unit mapping to create the interface object in COMOS.  
You have to set a base object reference if one of the following values is set in the "Structural behavior" control group:
  - Creation mode: "Create in unit tree" or
  - Assign mode: "Assign to object in unit tree"
6. "Base objects for creating devices" control group, "Creation in location tree" reference:  
Using drag&drop, specify which base object is used during location mapping to create the interface object in COMOS.  
Has to be set if one of the following values is set in the "Structural behavior" control group:
  - Creation mode: "Create in location tree" or
  - Assign mode: "Assign to location in location tree"

## See also

Algorithm for generating a COMOS object through name mapping (Page 123)

### 3.5.8.4 Configuring owner restriction rules

#### Types

There are three types of rules:

- The PDMS element type which the owner of an interface object has to have in PDMS.
- An expression which is evaluated by the owner and must be true.
- A filter which checks whether a given owner attribute contains a specific occur string.

#### Procedure

To define the owner restriction rules, proceed as follows:

1. Open the properties of the subclass definition object.
2. Switch to the "Attributes > Owner restriction rules" tab.
3. "Element type of owner" list:

Select the PDMS element type, which the owner of the interface object has to have in PDMS.

4. "Conditions" control group:

Optionally

Define any number of attributes and/or filters which are applied to the potential owner. To do this, proceed as described below.

## Result

If the potential owner violates one or more rules, the object is not exported or assigned.

## "Expressions" table

To define an expression, proceed as follows:

1. Click in the "Expression" column.
2. Enter an expression for which the following applies:
  - It is compatible with a PDMS block object.
  - It returns a boolean type value.
3. Press <ENTER> to confirm your entry.

## Result

- The expression is executed in PDMS at the potential owner.
  - Return value "True": The owner meets the condition.
  - Return value "False": The owner does not meet the condition.
- A line for another expression is added to the table.

Example:

```
name.Substring(1,2).eq('EP')
```

## "Filter" table

To define a filter, proceed as follows:

1. Click in the "Attribute" column.
2. Enter the name of the attribute as it is defined in PDMS.
3. Click in the "Occur string" column.
4. Enter the search string.
5. Press <ENTER> to confirm your entry.

## Result

- The filter is executed in PDMS at the potential owner. It will only return "True" if the search string is an occur string of the specified attribute.
- A line for another filter is added to the table.

## See also

"Owner restriction rules" tab (Page 171)

Owner restriction rules (Page 90)

### 3.5.8.5 Configuring the model

#### Requirement

If you are using design templates: The design templates have been imported from PDMS. You find information about importing design templates in section Importing AVEVA design templates (Page 136).

#### Inheritance of model information to interface objects

The COMOS DB is configured in such a way that the interface objects take the model information from their subclass.

If the interface objects are to use different model information from what is specified by their subclass, you can define different model information in the base objects of the interface objects.

As a user, you can overwrite the settings taken from the subclass or the base objects in the engineering view.

#### Procedure

To define model information at the subclass definition object, proceed as follows:

1. Open the properties of the subclass definition object and switch to the "Attributes > Model" tab.
2. "GTypes for filtering design templates" field:

If the geometry is based on a design template: Enter the GTypes. Filter the GTypes which are available for selection in design templates.

Delimiter: ";"

3. "Mode" list:

Define how the model is generated. Select one of the following modes:

Mode	Subclass of the "Pipe" class	Subclass of the "Tagged Item" class	Result
"Design template"	/	Available for subclasses with the "EQUI" PDMS element type	The geometry is based on a design template.
"PDMS element"	/	Available	The geometry is copied from an existing PDMS element.
"Function"	Available	Available	The geometry is generated by a function which is also performs other initializations.
"None"	Available	Available	A box is generated.

4. Depending on which mode you have selected, you now need to define the details for the generation of the model. Proceed exactly as described in section Overwriting the inherited model information (Page 138) for the configuration of the interface object.

See also

"Model" tab (Page 171)

3.5.9 Working with String Parameters in the name directory

3.5.9.1 Structure of a String Parameter

String Parameters are defined in the mapping table on the "Name directory" tab. Each line defines a String Parameter.

A String Parameter has the following constituent parts:

- A name: "Name" column  
This is the name used to address the String Parameter.
- A value of the string data type:  
The value is calculated/set by evaluating one of the following columns:
  - "Comos attribute"
  - "PDMS attribute/expression"
  - "GetFunction"
  - "SetFunction"

See below for details.



- Flags which define how the String Parameter is processed by interface operations  
 The flags are set in the following columns:
  - "Check"
  - "Export"
  - "Import"
  - "Admin"
  - "Rule"
- An entry from the units mapping in the "@PDMSMAP" folder.  
 This tells the interface operations which unit COMOS and PDMS expect or return and they convert the value accordingly.
- The data type transferred to the PLM function in PDMS and into which the string of the String Parameter is converted in PDMS.  
 If no PLM function is used: The PDMS attribute type.

**The value of the string data type**

The string has one of the following values:

<b>Value comes from COMOS</b>	<b>Value comes from PDMS</b>
An attribute value of the interface object which is processed by the same interface operation which evaluates the name directory.	An attribute value of the PDMS interface object that is set as the current element.
Any other property of an object other than the interface object, such as the name or the label.	A value which is returned by a PDMS expression. The PDMS expression is evaluated after the currently view PDMS interface object was set as the current element.
A fixed string.	A fixed string.
Special case: An attribute value of an object other than the interface object.	/

### 3.5.9.2 Configuring String Parameters

#### Configuration

To configure a String Parameter, proceed as follows:

1. Open the properties of the subclass definition object.
2. Switch to the "Attributes > Name directory" tab.
3. Configure the "Name directory" table:

- "Name" column:

Enter the name used to address the String Parameter.

- Remaining columns:

Configure the other columns of the String Parameter depending on which tasks the String Parameter is to fulfill. The individual configuration options can be combined so that a String Parameter can fulfill various tasks.

#### Tasks

A String Parameter can fulfill the following tasks:

1. It assigns a COMOS value to a variable.\*
2. It assigns a PDMS value to a variable.\*
3. It writes a PDMS value to a COMOS attribute (import).
4. It writes a COMOS value to a PDMS object (export).
5. It generates a component of a mapping entry in name mapping (variables definition).
6. An interface operation compares the value of a COMOS interface object returned by the String Parameter with the value of the connected PDMS object returned by the String Parameter (check).
7. The "Export" interface operation uses the String Parameters to check whether it is permissible to create an object under a specific owner.

The tasks marked with "\*" provide the basis for the remaining tasks.

#### Assign a COMOS value to a variable (task 1)

- Configure the "Comos attribute" column.  
Information regarding configuration: See the end of the section.
- "PDMS type" column: Select any data type. The software expects a value.

#### Assign a PDMS value to a variable (task 2)

- Configure the "PDMS attribute/expression" or "GetFunction" column.  
Information regarding configuration: See the end of the section.
- "PDMS type" column: Select any data type. The software expects a value.

### Import (task 3)

Configure the following columns so that the String Parameter writes a PDMS value to a COMOS attribute:

- "Comos attribute" column:  
Only attribute name permitted. Information regarding configuration: See the end of the section.
- "PDMS attribute/expression" or "GetFunction" column:  
Information regarding configuration: See the end of the section.
- "Unit" column:  
You are offered the entries from the units mapping in the "@PDMSMAP" folder. Choose a suitable entry.
- "PDMS type" column:  
Specify the data type that is to be transferred to the PML function.
- "Import" column: Select "Yes".

### Export (task 4)

To enable the String Parameter to write a COMOS value to a PDMS attribute, proceed as follows:

- "Comos attribute" column:  
Information regarding configuration: See the end of the section.
- "SetFunction" or "PDMS attribute/expression" column:  
Information regarding configuration: See the end of the section.  
If both columns were configured, SetFunction has a higher priority.
- "Unit" column:  
You are offered the entries from the units mapping in the "@PDMSMAP" folder. Choose a suitable entry.
- "PDMS type" column:  
Specify the data type that is to be transferred to the PML function.
- "Export" column:  
Select "Yes".

### Generate the name mapping part (task 5)

Proceed as described below so that the String Parameter generates one of the following components of a mapping entry in name mapping.

- "Data item"
- "Prefix"

- "Postfix"
- "Fixed name"

Configure the following columns of the String Parameter:

- "Admin" column:  
Select "Yes".
- "Comos attribute" or "PDMS attribute/expression" or "GetFunction" column:  
Define which string the String Parameter returns.  
If more than one column is set, following priority applies: "Comos attribute" > "GetFunction" > "PDMS attribute/expression"  
Information regarding configuration: See the end of the section.
- "PDMS type" column:  
Select any data type. The software expects a value.

#### Check (task 6)

Configure the following columns so that an interface operation compares the values of a COMOS object returned by the String Parameter with those of the PDMS object to which it is connected:

- "Check" column:  
Select "Yes".
- "Comos attribute" and "PDMS attribute/expression" or "GetFunction" column:  
Access a COMOS value and a PDMS value.  
Information regarding configuration: See the end of the section.
- "PDMS type" column: Select any data type. The software expects a value.

#### Check owner (task 7)

Configure the following columns so that the "Export" interface operation uses the String Parameters to check whether it is permissible to create an object under a specific owner:

- "Rule" column:  
Select "Yes".
- "Comos attribute" and "PDMS attribute/expression" or "GetFunction" column:  
Access a COMOS value and a PDMS value.  
The PDMS expression GetFunction is called at the potential owner of the object that is to be created in PDMS.  
Information regarding configuration: See the end of the section.
- "PDMS type" column: Select any data type. The software expects a value.
- The object is only created if the values are the same.

## Miscellaneous

Please note the following points:

- If the "Export" and "Import" columns are activated, the "Export" column takes higher priority. The "Import" column is not evaluated.
- "PDMS type" column: Always mandatory, even if no value is transferred to a PML function.
- "SetFunction" and "PDMS attribute/expression" columns are set:  
SetFunction has higher priority.
- "GetFunction" and "PDMS attribute/expression" columns are set:  
GetFunction has higher priority.

## See also

Configuring the "Comos attribute", "PDMS attribute/expression", "GetFunction", and "SetFunction" columns (Page 117)

### 3.5.9.3 Configuring the "Comos attribute", "PDMS attribute/expression", "GetFunction", and "SetFunction" columns

Which of these columns you configure and which entries are permitted depends on the purpose for which you are using the String Parameter.

## Procedure

Step 1: Define which task the String Parameter fulfills. This derives the action it executes.

Purpose	Action
Variable definition	The value is fetched from PDMS or COMOS.
Export	The value is fetched from COMOS and written to PDMS.
Import	The value is fetched from PDMS and written to COMOS.
Check	A value is fetched from COMOS and PDMS in order to compare the two values with each other.

Step 2: The actions determine which columns you configure:

Action	Column	Permissible entries
Fetch value from COMOS	Column: "Comos attribute":	<ul style="list-style-type: none"> <li>• Attribute name</li> <li>• Fixed string</li> <li>• Call of an owner method</li> </ul>
Set value in COMOS	Column: "Comos attribute":	<ul style="list-style-type: none"> <li>• Attribute name</li> </ul>

Action	Column	Permissible entries
Fetch value from PDMS	"PDMS attribute/expression" column: or	<ul style="list-style-type: none"> <li>• Attribute name</li> <li>• Expression</li> <li>• Fixed string</li> </ul>
	"GetFunction" column:	<ul style="list-style-type: none"> <li>• Name of a GetFunction</li> </ul>
Set value in PDMS	Column: "PDMS attribute/expression": or	<ul style="list-style-type: none"> <li>• Attribute name</li> </ul>
	"SetFunction" column:	<ul style="list-style-type: none"> <li>• Name of a SetFunction</li> </ul>

**Notation**

Use the following notation:

Column	Notation
"Comos attribute"	<ul style="list-style-type: none"> <li>• Attribute name: Use the NestedName: "&lt;Tab name&gt;.&lt;Attribute name&gt;" The String Parameter returns the DisplayValue of the attribute.</li> <li>• Fixed string: Set the fixed string inside double quotation marks.</li> <li>• Call of an owner method: Enter a command in the field which returns the required value. Use the key word <code>owner</code>. The same functions and properties are available as in the COMOS Object Debugger. Example: The String Parameter should return the label of the owner of the currently processed interface object: "Owner.Label"</li> </ul>
"PDMS attribute/expression"	<ul style="list-style-type: none"> <li>• Attribute name: Enter the name of the PDMS attribute.</li> <li>• Expression: Enter the PDMS expression.</li> <li>• Fixed string: Set the fixed string inside single quotation marks.</li> </ul>
"GetFunction"	Enter the name of the GetFunction which is to be executed in PDMS. The GetFunction returns a PDMS value. Example: "!!GetPurpose()"
"SetFunction"	Enter the name of the SetFunction which is to be executed in PDMS. The SetFunction writes a value to PDMS. Example: "!!SetPurpose()"

**See also**

Configuring String Parameters (Page 114)

## 3.5.10 Configuring unit and location mapping in name mapping

### 3.5.10.1 Structure of the tables for unit and location mapping

You find the tables for unit mapping and location mapping at the subclass definition objects on the "Name mapping" tab.

- The tables for unit mapping and location mapping have the same structure.
- Each column in the table defines a mapping entry.
- Each mapping entry contains the rows listed in the table below. These rows save all information that is required to execute one of the following actions:
  - Generate the name of the PDMS object to which a COMOS interface object is connected
  - Derive the path to the COMOS interface object to which the object is connected from the name of a PDMS object.
  - Create the COMOS interface object at the position in the tree structure stipulated by the PDMS name.
  - Complete the owner structure of the COMOS interface object identified by the PDMS name.
  - Assign a reference from a COMOS interface object to another COMOS object.

Row name	Function	Comment
"Prefix"	The prefix of a PDMS name part.	Used to generate/break down the PDMS name part of the mapping entry.
"Data item"	The data item of a PDMS name part.	
"Start index"	The point where a PDMS name part is inserted in the full PDMS name.	
"Number of characters"	The length of a PDMS data item.	
"Postfix"	The postfix of a PDMS name part.	
"Fixed name"	The name of a COMOS object, used to structure the COMOS data	Only used to generate a path to a COMOS object
"Base object"	Base object used to create an object.	Only used to complete the owner structure of the COMOS interface object or to create the interface object in COMOS.
"Base object from structure"	Flag whether the engineering structure linked to the project is used to create an object.	

### 3.5.10.2 Algorithm for generating a PDMS name

#### Introduction

A PDMS name comprises a number of name parts. Each mapping entry in name mapping returns a name part. When put together, the mapping entries produce a full PDMS name.

#### Elements of a PDMS name part

The PDMS name part of a mapping entry is defined by the following elements:

- Optionally: A prefix
- A data item
- A position within the full name
- A length
- Optionally: A postfix

#### Content of the data item

The data item of a mapping entry contains either COMOS data or PDMS data. The following data is permitted:

COMOS data	PDMS data
<ul style="list-style-type: none"><li>• Return value of a String Parameter which returns COMOS data.</li></ul>	Return value of a String Parameter which returns PDMS data.
<ul style="list-style-type: none"><li>• An expression which returns the name or label of an object in the unit tree.</li></ul>	/
<ul style="list-style-type: none"><li>• An expression which returns the name or label of an object in the location tree.</li></ul>	/



## Algorithm

To generate the name of the PDMS object to which a COMOS interface object is to be connected, the interface proceeds as follows:

1. It checks which subclass the COMOS object has and evaluates the creation mode of the subclass definition object:
  - "Create in unit tree": Unit mapping is performed.
  - "Create in location tree": Location mapping is performed.
2. The mapping entries are evaluated. The evaluation is performed from left to right. The following algorithm is executed for each mapping entry:
  - The data item is evaluated.
  - If no data entry was defined, the name part of this mapping entry is not processed. The algorithm continues with the next mapping entry.
  - If the data item is longer than is permitted by the "Number of characters" cell, it is truncated.
  - Prefix (if necessary truncated), data item and postfix are combined to form a character string - the PDMS name part.
3. The PDMS name parts are combined to form a string which cannot be modified. Each name part is copied to the position in the string defined by its start index.
4. If the index ranges of two name parts overlap, the name part of the later evaluated mapping overwrites the first evaluated mapping.

### 3.5.10.3 Algorithm for generating a COMOS path name

#### Introduction

To find the COMOS object to which a PDMS object is to be connected, the interface breaks down the name of the PDMS object into its name parts. The interface then derives the path to the correct node in the tree structure from the name parts.

#### Algorithm

The interface proceeds as follows:

1. It checks the which subclass the PDMS interface object has. To do this, it evaluates the following UDA: ":ComosSClass"
2. It evaluates the creation mode of the corresponding subclass definition object: "General" tab, "Structural behavior" control group, "Creation mode" list:
  - "Create in unit tree": Unit mapping is performed.
  - "Create in location tree": Location mapping is performed.

The execution of the mapping tables involves the following steps:

1. Starting with the last PDMS name part, the interface extracts the PDMS name parts from the PDMS name. To do this, it evaluates the start index and the number of characters of the corresponding mapping entry.

For the last name part, the mapping entry on the far right is used; for the penultimate name part, the second mapping entry from the right is used, and so on.

2. The interface processes the individual mapping entries starting with the mapping entry on the far right.

The following algorithm is executed for each mapping entry:

1. The entry in the "Data item" row is checked:
  - Name of a String Parameter:  
Consequence: A corresponding COMOS node does not exist for the name part.  
Continue with step 2.
  - Name or label of a COMOS object:  
If there is one: The prefix and postfix are removed from the PDMS name part.  
The remainder of the string represents the name or the label of a COMOS object. It is copied to the path list.
  - Nothing entered: Continue with step 2.
2. If the "Data item" row is empty or contains a String Parameter, a check is made whether the "Fixed name" row has been configured:
  - "Fixed name" not set: The processing of the mapping table is aborted for the object.
  - "Fixed name" set: The entered or resulting string is written to the path list.

Once all mapping entries have been processed, the sequence of the entries in the path list is reversed.

## Result

- The string written to the path list produces the path to the COMOS interface object you are looking for.
- If the object does not exist yet, the mapping table returns all needed information in order to create it. The name part of the last mapping entry is written to the "Name" property of the COMOS object.
- If the interface detects that the owner structure of the object you are looking for is incomplete: The mapping table returns all information required to complete the owner structure.

### 3.5.10.4 Algorithm for generating a COMOS object through name mapping

#### Algorithm

To create an interface object in COMOS and, if necessary, to complete its owner structure, the interface has to know which base objects are used.

To create the owner structure, COMOS evaluates the "Base object" or "Base object from structure" row of the name mapping.

To create the interface object which is actually searched for, the interface starts by evaluating the UDS ":ComosBaseOb" of the PDMS interface object. If the UDA is empty, it evaluates the "Base object" or "Base object from structure" row in the name mapping. If no entry has been made here, the interface evaluates the "General" tab of the subclass definition object.

This documentation assumes that this setting is made via the "General" tab.

If you define the base object via name mapping, the algorithm described below is used.

#### Evaluation of the "Base object" row

The here specified base object is used.

#### Evaluation of the "Base object from structure" row

Only activate the "Base object from structure" setting if the project is linked to a project structure.

If the setting is activated, the interface proceeds as follows:

- It navigates to the project structure level corresponding to the current mapping entry.
- In a project structure, several objects with different base objects can be prepared for a single level. The interface compares the data item or fixed name of the current mapping entry with the name mask of the possible base objects.
- The object is created with the base object whose name mask matches the data item or fixed name of the mapping entry.
- The data item or fixed name is entered in the "Name" property of the new object.

#### See also

Defining the base object for the creation of interface objects (Page 108)

### 3.5.10.5 Configuring unit mapping and location mapping

#### Requirement

You have defined the structural behavior.

#### Structural behavior determines configuration

The structural behavior of the subclass determines which of the mapping tables you can configure on the "Name mapping" tab:

Creation mode	Assignment mode	Mapping
"Create in unit tree"	"Assign to location in location tree"	Unit mapping and location mapping
"Create in unit tree"	"None"	Unit mapping
"Create in location tree"	"Assign to location in unit tree"	Unit mapping and location mapping
"Create in location tree"	"None"	Location mapping

#### Procedure

To configure unit or location mapping, proceed as follows:

1. Open the properties of the subclass definition object.
2. Select the "Attributes > Name mapping" tab.
3. Configure the "Unit mapping" and "Location mapping" tabs as described in the following.
4. Identify where objects of the subclass are located or need to be created in your COMOS object structure. Split the overall path into its individual nodes.
5. Analyze how the PDMS name for objects of the subclass is structured and split it into its name parts.

A PDMS name part consists of the following constituent parts:

- (Optional) prefix
- Data item
- (Optional) postfix

6. Analyze which COMOS nodes and PDMS name parts can be mapped directly to one another.

Configure a shared mapping entry for these nodes and name parts. Information regarding configuration: See the table below.

7. Analyze which COMOS nodes do not have a corresponding PDMS name part and vice versa.

Configure a separate mapping entry for each of these nodes and name parts. Information regarding configuration: See the table below.

Examples of COMOS nodes which do not have a corresponding PDMS name part:

- Folder
- The PDMS root object on the "Locations" tab
- Other objects that are only used to structure the COMOS data

8. Configure the mapping entries as described in the table below.

The sequence of the mapping entries is the same as the sequence of the COMOS nodes from the overall path, although if necessary it may be interrupted by mapping entries for PDMS name parts for which there are no nodes.

This makes it possible to generate a PDMS-compatible names from the object path which is unique throughout PDMS.

Information regarding notation: See below.

Row	COMOS node without a corresponding PDMS name part	COMOS node with a corresponding PDMS name part	PDMS name part without a corresponding COMOS node
"Prefix"	/	Optionally	
"Postfix"	/	Optionally	
"Data item"	/	<ul style="list-style-type: none"> <li>• Mandatory</li> <li>• Only an expression which returns the name or label of an object from the unit tree or location tree is permitted</li> </ul>	<ul style="list-style-type: none"> <li>• Mandatory</li> <li>• Only String Parameter is permitted</li> </ul>
"Start index"	/	<ul style="list-style-type: none"> <li>• Mandatory</li> </ul>	
"Number of characters"	/	<ul style="list-style-type: none"> <li>• Mandatory</li> </ul>	
"Base object"	<ul style="list-style-type: none"> <li>• Mandatory</li> </ul>	<ul style="list-style-type: none"> <li>• Mandatory</li> </ul>	/
"Base object from structure"	<ul style="list-style-type: none"> <li>• Only one of the two</li> <li>• See below</li> </ul>	<ul style="list-style-type: none"> <li>• Only one of the two</li> <li>• See below</li> </ul>	/
"Fixed name"	<ul style="list-style-type: none"> <li>• Mandatory</li> <li>• String Parameter which evaluates COMOS data or</li> <li>• Fixed string</li> </ul>	/	

**"Base object" and "Base object from structure" rows**

- Mapping entry on the far right:  
 In this field you can specify which base object uses the interface object you are actually looking for.  
 This documentation assumes that the base object is defined via the "General" tab of the subclass definition object, in the "Base objects for creating objects" control group.
- Remaining mapping entries:  
 Define which base objects are used by the individual levels of the owner structure.

**Notation**

Use the following notation:

- "Prefix" row: Any number of characters
- "Postfix" row: Any number of characters
- "Data item" row:

A String Parameter or an expression composed of a number of elements:

String Parameter notation	Notation of the composed expression
"!<Name of a local String Parameter>" example: "!Area"	First element: <ul style="list-style-type: none"> <li>• In unit mapping: "U"</li> <li>• In location mapping: "L"</li> </ul>
"!!<Name of a global String Parameter>"	Second element <ul style="list-style-type: none"> <li>• "&lt;Number&gt;": Structure level in the tree on which the object is located                              In unit mapping: Starting from the project root                              In location mapping: Starting from the "@PDMS" folder</li> </ul>
	Third element: <ul style="list-style-type: none"> <li>• "N" or "L": Use the name or label of the object</li> </ul>
	Example: "U2L": Use the label of an object on the second level underneath the project root in the unit tree

- "Start index" row:  
 The start index  
 If nothing is entered, the name part is appended to the up to this point composed name parts.
- "Number of characters" row:
  - Number
  - Any number of characters permitted

- "Base object" row:
  - SystemFullName of the base object
  - Use "\" to separate the individual nodes.
- "Base object from structure" row: Flag
  - "Yes": If the project is linked to a project structure, the project structure is used for the generation of the COMOS objects.
  - "No": Even if a project structure is available and linked, it is not used.
- "Fixed name" row:

One of the following values:

  - Local String Parameter: "!<Name of a local String Parameter>"
  - Global String Parameter: "!!<Name of a global String Parameter>"
  - Fixed string: Without quotation marks

### 3.5.10.6 Configuring level rules

#### Procedure

To define a deviation from the standard evaluation of unit mapping and location mapping, proceed as follows:

1. Open the properties of the subclass definition object.
2. Activate the "Attributes > Name mapping" tab.
3. Right-click a field in the right-hand column of the "Level rules" table.

Result: The field is activated and highlighted in color.
4. Right-click the field a second time with the mouse.

Result: A list is displayed in the field.
5. To activate the rule, select "Yes".

To deactivate the rule, select "No".

#### See also

Level rules of name mapping (Page 86)

## 3.5.11 Configuring COMOS queries from PDMS

### 3.5.11.1 Overview

#### Workflow

Users who wish to execute and view COMOS queries in PDMS first have to complete the following preparatory steps:

- Create however many COMOS queries the user is to execute in PDMS.  
These queries are called main queries in the interface.
- Configure the main queries in the usual COMOS way.
- Create a class definition object for the "Query" class.
- For each main query: Create a subclass definition object.
- Configure the subclass definition objects.
- If the user should have the option to select the start object of a main query from a list of objects in PDMS:
  - Create a COMOS query whose result list returns the possible start objects  
This query is called the start object query in the interface.
  - Configure the start object query.
  - Configure the access to the start object query for the subclass definition object.

### 3.5.11.2 Creating and configuring COMOS queries

#### Procedure

- For each query that is started from and viewed in PDMS, create a main query and configure it in the usual COMOS way.  
If you are using an already existing query as the main query, this step is not necessary.
- If the user should have the option to select the start object of a main query from a list of objects in PDMS:  
Create a start object query and configure it in such a way that it returns useful start objects for the main query.  
If you are using an already existing query as the start object query, this step is not necessary.



### 3.5.11.3 Creating definition objects for queries

#### Procedure

1. Create the class definition object for queries as described below.
2. For each main query you have defined, create a subclass definition object as described below.

#### Creating the class definition object

To create the class definition object for queries, proceed as follows:

1. Click the "Units" tab in the Navigator.
2. Select the "@PDMSMAP" folder.
3. Select the "New > Query definition for class queries" command from the context menu.

#### Creating the subclass definition object

To create a subclass definition object you use to manage a main query and its start object query, proceed as follows:

1. Select the "Query" class definition object in the Navigator.
2. Select the "New > SUBCLASS\_QUERY base object for query subclass definitions" command from the context menu.
3. Open the properties of the subclass definition object.
4. "PDMS settings" control group:
  - "Name" field:  
Enter a name. The name is offered to the user for selection in PDMS: "Query Comos" window, "Query" tab, "Query" list
  - "Description" field:  
Enter a description. If the user has selected the query in PDMS, the description is displayed there: "Query Comos" window, "Query" tab, "Description" field

### 3.5.11.4 Configuring a subclass definition object for queries

#### Procedure

To configure the subclass definition object you use to manage the main query and the start object query, proceed as follows:

1. Open the properties of the subclass definition object.
2. Switch to the "Attributes > General" tab.

3. "Output format" control group:

- "Output format" control group:

Select the output format in which the result list of the query will be transferred to PDMS.

Possible output formats: CSV and XML.

- "Delimiter" field:

If CSV is set as the output format: Define the separator for columns. The separator character is arbitrary.

4. "Main query" control group:

- "Name" reference:

Drag&drop the main query from the Navigator into the field.

- "Start object" reference:

The default start object.

The name of the here referenced start object is displayed to the user in PDMS: "Comos Query" window, "Query" tab, "Start object" field

If you are using an already existing query as the main query: You can define a start object other than the one set in the main query.

Drag&drop the start object from the Navigator into the field.

- "Base objects" list:

The base objects according to which the query filters the result list.

If you are using an already existing query as the main query: You can filter the query in PDMS on the basis of base objects other than those defined by the main query in COMOS.

Drag&drop the base objects from the Navigator into the list.

5. "Query for start object" control group:

Only configure this group if the user should to be able to select the start object of a main query from a list of objects in PDMS:

- "Name" reference:

The start object query which returns the list of possible start objects in PDMS.

Drag&drop the start object query from the Navigator into the field.

- "Start object" reference:

If you are using an already existing query as the start object query: You can define a start object other than the one set in the start object query.

- "Base objects" list:

The base objects according to which the start object query filters the result list.

If you are using an already existing query as the start object query: You can filter the query in PDMS on the basis of base objects other than those defined by the main query in COMOS.

Drag&drop the base objects from the Navigator into the list.

### Evaluation priority for the start object

You can define the start object of the query in various places. The following table lists these places.

If a start object has been defined in various places, the places which appear further down the table overwrite those above them.

Start object of the main query	Start object of the start object query
The start object which is set in the COMOS query	The start object which is set in the COMOS query
The start object which was set at the subclass definition object	The start object which was set at the subclass definition object
The start object which the user defined in PDMS: "Comos Query" window, "Execute" tab	/

**See also**

The "General" tab (Page 172)

**3.5.12 Configuring "Import DocLinks"**

**3.5.12.1 Defining the entry for the target document in PDMS**

**Introduction**

To import draft objects into COMOS as DocObjs, the interface has to know the COMOS document for which the DocObjs are being created.

For this purpose, you should define an attribute or a UDA in which the user will subsequently enter the name or the SystemUID of the target document.

You decide on which level or levels of the draft structure target documents may be specified.

**Defining target documents on different levels of the draft structure**

You can specify a target document on more than one level of the draft structure. This enables the user to start the DocLink import as required on different levels of the draft structure, e.g. for all draft objects in a region, for an individual view, or for an individual drawing.

To do this, repeat the below described procedure for all levels for which the "Import DocLinks" operation needs to be available.

**Procedure**

To define where the target document is specified, proceed as follows:

1. Define whether the user enters the name of the target document in an attribute or in a UDA and which attribute or UDA is used.
2. Define the level of the draft structure on which the user specifies the name of the target document.
3. If you choose to use a UDA: Create the UDA.

### 3.5.12.2 Configuring "Import DocLinks" in COMOS

#### Procedure

To configure the "Import DocLinks" operation in COMOS, proceed as follows:

1. Create a subclass definition object of the "Document" class.
2. In the subclass definition object name directory, define a String Parameter which addresses the attribute or UDA in which the name or SystemUID of the target document is entered.
3. Configure the general properties of the subclass definition object.
4. Optionally: In the owner restriction rules, specify whether the owner of the draft objects for which DocObjs are being generated has to meet certain conditions.

#### See also

Defining the String Parameter for the document name (Page 133)

Configuring general settings (Page 134)

### 3.5.12.3 Defining the String Parameter for the document name

#### Aim

Define a String Parameter which, from PDMS, reads out the name of the COMOS document under which the DocObjs are imported.

#### Procedure

To define the String Parameter, proceed as follows:

1. Open the properties of the subclass definition object and switch to the "Name directory" tab.
2. Configure the "Name directory" table:
  - "Name" column: Enter the name of the String Parameter.
  - "PDMS attribute/expression" or "GetFunction" column: Access the value of the attribute or the UDA in which the document name or the SystemUID is saved in PDMS. See also section Working with String Parameters in the name directory (Page 112).
  - "PDMS type" column: Enter "String".

### 3.5.12.4 Configuring general settings

#### Procedure

To define the general settings of a subclass definition object of the "Document" class, proceed as follows:

1. Open the properties of the subclass definition object and switch to the "General" tab.
2. "Document name" field:
  - Enter the name of the String Parameter which reads out the document name from PDMS.
  - Do not precede the name of the String Parameter with a "!" character.
3. Optionally: "Base object for document search" control group:

Define filters.

Result: During the search, the interface restricts itself to documents which meet the criteria defined here. This speeds up the search.

The following filters are possible:

- "Document type" list: Select the document type of the document.
  - "Base object" reference: Select the base object the document uses.
  - "Start object" reference: Select the node under which the document is searched.
4. Optionally: "Filter unnamed objects" option:

Activated: In PDMS, only objects with a name are displayed in the result list in the "Export to Comos" window.
  5. "PDMS element types" list:

Enter the PDMS element types of the draft objects with a DDNM attribute which references a design object that is connected to a COMOS object.

### 3.5.12.5 Defining owner restriction rules for draft objects

#### Procedure

You can define owner restriction rules which the draft objects have to meet to be imported as DocObjs.

Proceed as when defining owner restriction rules at subclass definition objects of the "Pipe" and "TaggedItem" classes.

#### See also

Configuring owner restriction rules (Page 109)

### 3.5.13 Synchronizing settings

#### Initial situation

To ensure that COMOS and PDMS use the same settings, the interface does not directly access the settings for the "@PDMSMAP" folder and for the class and subclass definition objects. Instead, the settings are written to an XML file in the exchange directory. COMOS and PDMS load the settings from the file to the working memory of your computer.

When an interface operation is called, the current version number of the "@PDMSMAP" folder is transferred. The operation checks whether the number is identical to the version number of the settings stored in the working memory.

- Yes: The settings stored in the working memory are used.
- No: The file is loaded to the working memory again. The settings are synchronized.

#### Synchronization successful

Make sure that the current settings in the file in the exchange directory are up to date.

Update the file whenever you have performed one of the following actions:

- You changed the settings of the "@PDMSMAP" folder.
- You created a new class or subclass definition object.
- You changed the settings of a subclass definition object.

#### Procedure

To update the file in which the settings for the "@PDMSMAP" folder and the class and subclass definition objects are saved, proceed as follows:

1. Select the "@PDMSMAP" folder in the Navigator.
2. Select the following command from the context menu: "PDMS > Save subclasses".

#### Result

- The version number in the "@PDMSMAP" folder is incremented.
- The standard table for subclasses is updated.
- The settings for the "@PDMSMAP" folder and for the class and subclass definition objects are written to the file in the exchange directory.

### 3.5.14 Importing AVEVA design templates

#### Initial situation

You can assign an AVEVA design template to an interface object as a model.  
To do this, you need to import the design templates from PDMS into COMOS.

#### Procedure

To import the AVEVA design templates from PDMS into COMOS, proceed as follows:

1. Switch to the "Base objects" tab in the Navigator.
2. Select the following node: "@PDMS > @TPL Design templates".
3. In the context menu, select the following command: "PDMS > Update design templates".

#### Result

- The GTypes of all templates are evaluated in PDMS.
- The templates are imported from PDMS to COMOS. During this process, the interface proceeds as described below.

#### Managing the design templates in the base data

The design templates are managed under the following node in the base data:

"@PDMS > @TPL"

The design templates are managed sorted by GType:

- Nodes which represent the GTypes of the templates are located on the first level below "@PDMS > @TPL".
- The nodes for the templates are located on the second level. A node is created for each template.

#### Algorithm for importing the design templates into the base data

- For each GType found in PDMS, a search is performed below "@PDMS > @TPL" for a node whose name matches the name of the GType.
- If a node of this type does not exist yet, it is created. The name of the GType is entered in the "Name" property of the node.
- A node for the design template is created underneath the GType node.



- If the design template already exists, it is updated. No data is deleted during the update operation.

Example:

- A sub template was deleted in PDMS.
- The sub template remains available in COMOS even after the update.
- The attributes of the template node are automatically set when the template node is created or during the update.

## 3.5.15 Configuring COMOS interface objects

### 3.5.15.1 Assigning the class, subclass, and PDMS element type

#### Procedure

To assign a class, a subclass, and PDMS element types to an interface object, proceed as follows:

1. Click on the "Base objects" tab in the Navigator.
2. Navigate to an interface base object.
3. Open the properties of the interface base object.
4. Switch to the "Attributes > External 3D Interface" tab.
5. Configure the tab as described below.
6. "Class" list:
  - Select a class.
  - Do not select the "Undef" value. Otherwise, the object will not be recognized as an interface object.
7. "Subclass" list:

Select a subclass. Only subclasses permitted for the selected class are offered.
8. "PDMS element type" list:

Select a PDMS element type. Only those element types entered for the selected subclass are offered.

#### Overwriting the preselection in the engineering view

In the COMOS DB, the edit mode of these attributes is configured in such a way that users can overwrite the inherited settings in the engineering view, if this is necessary.

#### See also

Attributes of the "External 3D Interface" tab (Page 175)

### 3.5.15.2 Overwriting the inherited model information

#### Requirements

- A class, a subclass, and a PDMS element type have already been assigned to the interface object.
- AVEVA templates have been imported.

#### Inheritance from subclass

The COMOS DB is configured in such a way that the interface objects take the model information settings from their subclass. For this purpose, the model attributes of the interface objects are linked to those of the subclass definition objects.

Link type: Static link via the "GetLinkedSpecification" script function.

The COMOS DB is configured in such a way that you can overwrite the settings taken from the subclass at the base objects of the interface objects.

#### Procedure

To define the model information at the base object of the interface object, proceed as follows:

1. Click on the "Base objects" tab in the Navigator.
2. Navigate to an interface base object.
3. Open the properties of the interface base object and switch to the "Attributes > External 3D Interface" tab.
4. Configure the "Model" control group as described in the following.

5. "Mode" list:

Define how the model is generated. Select one of the following modes:

Mode	Interface has subclass of the "Pipe" class	Interface object has subclass of the "Tagged Item" class	Result
"Design template"	/	Available if the subclass has the "EQUI" PDMS element type	The geometry is based on a design template.
"PDMS element"	/	Available	The geometry is copied from an existing PDMS element.
"Function"	Available	Available	The geometry is generated by a function which is also performs other initializations.
"None"	Available	Available	A box is generated.

6. Depending on which mode you have selected, you now need to define the details for the generation of the model:

- "Function" mode:

Enter the function call in the "Function" field.

- "PDMS element" mode:

Enter the name of the PDMS element whose geometry is being copied in the "Element" field.

- "Design template" mode:

Define the main template in the "Design template" control group, along with the used sub templates which, if applicable. Further information: See below.

### Defining design templates

If the following conditions are met, you can select design templates as the basis for your model:

- The subclass of the interface object has the "Tagged Item" class.
- The subclass has the "EQUI" PDMS element type.

If you select a design template as the basis for your model, you need to configure the "Design template" control group.

Proceed as follows:

1. "Main template" list:
  - Select the main template.
  - Only those templates whose GType has been entered in the subclass in the "GTypes for filtering design templates" field are listed for selection.
  - If the the "EQUI" PDMS element type is missing at the subclass, the list remains empty.
2. "Sub template 1" to "Sub template 5" lists:
  - You can define up to five templates for secondary equipment.
  - Which templates are offered depends on the main template you selected.

### Design templates in the engineering view

When COMOS creates an interface object in the engineering view, it automatically checks whether the interface object uses a design template. If it does, Comos creates the main template underneath the interface object and the sub templates underneath that.

COMOS also creates the template objects when you assign a design template to an existing interface object.

- Name of the template object: "@TPL"
- Description of the template object: As defined at the base object of the template object

You can change the design parameters of the template objects in the engineering view: Properties of the template object, "Attributes > External 3D Interface" tab, "PDMSDP01" through "PDMSDP10" attributes.

### Overwriting the base object settings in the engineering view

Users can overwrite both the settings inherited from the subclass and the settings predefined at the base object in the engineering view.

### Restoring the default settings

The settings inherited from the subclass can be restored in the engineering view using the "Refresh static links on tab" context menu.

### See also

Attributes of the "External 3D Interface" tab (Page 175)

## COMOS 3D viewing

### 4.1 Requirements

The 3D viewer application must support the COMOS 3D view communication system. The 3D data must be linked to the corresponding COMOS systemUIDs in the 3D model.

### 4.2 Script adjustments

#### Requirement

To adapt a script, you need a suitable COMOS project database and an interface-compatible 3D viewer, such as Walkinside from VRcontext.

#### Procedure

The `Sub OnProjectOpen(Project)` script must be adapted in order for the COMOS 3D viewer to start in the database.

1. Open the base project.
2. Select the "Base objects" tab in the Navigator.
3. Open the properties of the required project type under the "@J Project" node.
4. Switch to the "Script" tab.
5. Save the script that then appears to the `Sub OnProjectOpen(Project)` function. See also chapter `Sub OnProjectOpen(Project)` (Page 179).
6. To check whether the creation procedure was successful, comment the `'msgbox` entries which have been commented out back in again.

Create an exchange directory to enable COMOS to communicate with the 3D viewer.

1. Open the "<COMOS installation path>\config" directory.
2. Save the script that then appears to the "comos.C3DView.config" file. See also chapter Exchange directory (Page 180).



# COMOS NX - Routing Mechanical interface

## 5.1 Overview

- The interface addresses the NX Routing Mechanical module and is supported by NX version NX 7.5.2 or higher.
- The routing technology used, which is based on COMOS P&ID, is integrated into the NX 3D CAD tool.
- P&ID pipe topologies and the required process data are transferred between COMOS and NX by exchanging XML files.
- COMOS NX exchanges object information which enables Teamcenter components to be selected.

## 5.2 Default settings in COMOS

### Script adjustment

The `Sub OnProjectOpen (Project)` script must be adapted in order for the COMOS NX - Routing Mechanical interface to start.

1. Open the base project.
2. Select the "Base objects" tab in the Navigator.
3. Open the properties of the required project type under the "@J Project" node.
4. Switch to the "Script" tab.
5. Save the script that then appears to the `Sub OnProjectOpen (Project)` function. See also chapter `Sub OnProjectOpen (Project)` (Page 181).
6. To check whether the creation procedure was successful, comment the `!msgbox` entries which have been commented out back in again.

### Configuration

Create an exchange directory to enable COMOS to communicate with NX.

1. Open the "<COMOS installation path>\config" directory.
2. Save the configuration file "Comos.nxviper.config". See also chapter COMOS exchange directory (Page 182).

The following information is stored in the configuration file:

- Exchange directory for XML files
- Start status of the interface

### 5.3 Default settings in NX

- Debug status
- Path to an export directory for NX ptb files

#### P&ID data

See also chapter P&ID data (Page 21).

#### Linking P&ID and PPC

See also chapter Linking between P&ID and PPC (Page 22).

#### COMOSDB

The COMOSDB requires the following tab in order to export objects as a PTB file:

- "NXV010 NX 3D object mapping -> @10|NXV|1|NXV010 NX 3D object mapping"

Copy the tab from "Base project > @10 @Y Attribute catalog > NXV @Y NX Viper > 2 @Y Tabs + attributes".

Each object that you export to NX needs this tab. See also chapter "NX 3D object mapping" tab (Page 184).

## 5.3 Default settings in NX

### Configuration

Create an exchange directory to enable NX to communicate with COMOS.

1. Start NX.
2. Click on the "File > Utilities > Customer Defaults..." menu.
3. In the "Customer Defaults" window, click on the "Routing > Extras" menu.
4. Specify the exchange directory on the "Schematics Integration" tab.

The path specification must be identical to that of the COMOS exchange directory. See also chapter COMOS exchange directory (Page 182).



## References

### 6.1 Plant Modeler

#### 6.1.1 User interface reference

##### 6.1.1.1 Control elements on the "Plant Modeler" tab

###### "Storage settings" control group

Field	Description
"Default location for pipes"	The default location for pipes. All 3D objects that you unassign from cells in MicroStation are moved to the node specified here.
"Default location for equipment"	The default location for equipment. All 3D objects that you unassign from cells in MicroStation are moved to the specified node.

###### "Database settings" control group

Field	Description
"Plant Modeler start node"	References the "PLM" node in the base data, providing you with the opportunity to move the node if necessary.
"Pipe spec limit"	References structure objects. If you do not make any of your own settings here, default objects are displayed automatically.
"Part unit structure"	
"KKS system count"	
"PPC start node"	References the "PPC" node in the base data, providing you with the opportunity to move the node if necessary.
"Cell library start node"	References the start node underneath which the cell libraries are located in the base data, providing you with the opportunity to move the node if necessary.

**"Document management settings" control group**




Field	Description
"Default seed file"	3D standard seed file The standard seed file is imported into the ComosDB.
"Auto backup on exit"	Specifies that the last displayed status is stored in the backup folder when the Plant Modeler is closed. You can also activate this option in the Plant Modeler in the "Document management" window. To do so, select the "Auto backup on exit" command in the "Options" menu. You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Document management in the Plant Modeler".
"Auto check-in of working DGN on exit"	When you have started the Plant Modeler through the checking-out and editing of a DGN file, COMOS tries to check the DGN file back in when the Plant Modeler is closed. You can also activate this option in the Plant Modeler in the "Document management" window. To do so, select the "Auto check-in of working DGN on exit" command in the "Options" menu. You can find additional information on this topic in the "3D Integration Operation" manual, keyword "Document management in the Plant Modeler".

**"Microstation settings" control group**

Field	Description
"Startup file commands"	Reference to the start file "ComosPlantModeler.Startup" of the Plant Modeler. Example: C:\Program Files\Bentley\Microstation V8i\MicroStation\mdlapps\ComosPlantModeler.Startup
"Interface settings"	Interface with which the Microstation program is opened. Default: "comosplantmodeler"
Configuration file (optional)"	Configuration file with which the Microstation program is opened.

**"Plant Modeler" tab**

The following table describes the buttons that are available in the project properties on the "Plant Modeler" tab:

Button	Tooltip	Description
	"Set Pointer"	Opens a window where you select locations or other references.
	"Remove Pointer"	Deletes the pointer.
	"Navigate, Properties"	Depending on your selection, navigates: <ul style="list-style-type: none"> <li>• To the object</li> <li>• To the object properties</li> <li>• To the base object</li> <li>• To the base object in the base project</li> <li>• To uses</li> </ul>
"Reinitialize"		Updates the default settings for the "Default location for pipes" and "Default location for equipment" fields. Also restarts Plant Modeler and reads out the configuration file again.

**6.1.1.2 "MessageSystem <mode> as <node>" window**

This window is a debug tool for communication between Microstation and COMOS. It contains information about the communication mode.

It is opened from within COMOS and/or Microstation when you enter the value `true` for the `CommunicationWindowEnabled` key in the corresponding configuration file. See also Section Keys for communication mode (Page 36).

The name of the window is determined by the program from which it was opened and the communication mode set in the corresponding configuration file.

Wildcard	Parameters
<Mode>	<ul style="list-style-type: none"> <li>• FILE</li> <li>• TCP</li> </ul>
<Node>	<ul style="list-style-type: none"> <li>• Client (Microstation)</li> <li>• Server (COMOS)</li> </ul>

The most important elements of this window are described below.

## Control elements

Control element	Description
"Connected/Disconnected" field	<ul style="list-style-type: none"> <li>Connected A connection with the other program exists.</li> <li>Disconnected A connection with the other program does not exist.</li> </ul>
"Save" button	Saves all information from the window with the exception of the "Manual Message" tab in the cpm_debug_output.xmlfile.
"Load" button	Loads a saved XML file and opens a new window for the debug display.
"Running/Paused" button	Pauses message transmission.

## "Info" tab

The configuration settings for the corresponding program are displayed here.

## 6.1.1.3 Attribute Transfer Editor

## "Transfer Template Information" control group

Control element	Description
"Template CDevice" field	You drag an existing template that you wish to edit to this field. If you create a new template, the base object of the template is created automatically and shown in this field.
"Template Name" field	Enter a name when you create a new template.
"Version number" field	Version number of the template. Is updated automatically when you edit and save an existing template.
"Edit" button	Releases a template for editing. Only visible when you drag an existing template to the "Template CDevice" field.
"Create template" button	Creates a new template when you have specified the name of the template.
"Delete template" button	Clears the contents of the template that is currently open.

**"COMOS Base object properties" control group**

Control element	Description
Field for base object	You drag to this field a base object from whose properties you wish to use attributes for the template.
Upper area	The tabs and attributes of the object which you dragged to the field above are shown here.
Lower area	The available system properties for the object which you selected in the first area are shown here.

**"Attribute lists of the template" control group**

Control element	Description
"Own object" attribute list	You drag attributes from the "COMOS Base object properties" control group to this field. The attributes are evaluated based on the 3D object to which you assign the template
"Owner" attribute list	You drag attributes from the "COMOS Base object properties" control group to this field. The attributes are evaluated based on the owner of the 3D object to which you assign the template
"Request object" attribute list	You drag attributes from the "COMOS Base object properties" control group to this field. The attributes are evaluated based on the request object of the 3D object to which you assign the template
"Delete" button	Deletes the contents of the corresponding attribute list.

If an object does not possess an attribute that you specified in one of the attribute lists, no value is shown for this attribute in the Plant Modeler The field remains blank.

**6.1.1.4 "References" tab****Control elements**

The control elements available on the "References" tab in the "Settings" window are described in the following table:

Control element	Description
"Use path for reference file" option	Enabling this option allows you to enter a directory for the reference file in the "Default reference file" field.
"Default reference directory" field	This field is only available if you have enabled the "Use path for reference file" option. To enter a path for the reference file in this field, click "...".
"Use environment variable for reference file path" option	Enable this option if you prefer to use the MicroStation environment variable as a directory for the reference file instead of a reference file path.

Control element	Description
"Use environment variable for reference file path" field	This entry field is only active if you have enabled the "Use environment variable for reference file path" option. The environment variable is generally set during the MicroStation project or user configuration.
"Default seed file" field	Click "..." next to this field to specify a directory for the seed file.
"Change references automatically" option	By enabling this option, you are specify that changes to cells in referenced files can be made and will be saved automatically.

### 6.1.1.5 "Center line routing" tab

#### Options

The following table describes the options that are available on the "Center line routing" tab in the "Settings" window:

Option	Description
"Accept connector directly"	When you enable this option, the snap mode is hidden automatically. Centerline routing is carried out based on the known standard MicroStation procedures.
"Force implementation order"	When you enable the "Force implementation order" option, implementations are only released for installation in the order in which they were created on the P&ID diagram.
"Display implementation name"	When you enable this option, the names of the implementations are displayed.
"Display implementation label"	When you enable this option, the labels of the implementations are displayed.
"Display implementations of current branch"	When you activate this option, only those implementations you can actually install in the current branch are displayed. When you disable this option, all objects COMOS finds are displayed in the pipe. In this case, routing stops when there are pipe dividing components. This means that not all components that are displayed can actually be mounted.

### 6.1.1.6 "Data exchange" tab

#### Fields

The following table describes the fields that are available on the "Data exchange" tab of the "Settings" window:

Field	Description
"Path+file for incoming messages"	The path to the exchange directory is displayed in this field. The file name for messages from COMOS is "CMessage.xml". The following path is set by default: "C:\Temp\CMessage.xml"
"Path+file for outgoing messages"	The path to the exchange directory is displayed in this field. The file name for messages from Plant Modeler is "MSMMessage.xml". The following path is set by default: "C:\Temp\MSMMessage.xml"

## 6.1.2 Base data reference

### 6.1.2.1 Structure of the "PLM" node in the base data

In the base project, the "Base objects" tab in the Navigator must include the "PLM Microstation" node.

#### "PLM" node

The following table describes the structure of the "PLM" node:

Subordinate node	Call on the "Base objects" tab	Description
"CELLIB Cell libraries"	"PLM > CELLIB"	This node is required in order to be able to create new COMOS base objects, based on the MicroStation cell library. needed so you can create new COMOS objects in Plant Modeler based on the MicroStation cell library.
"Y Microstation catalog"	"PLM > Y"	MicroStation catalogs for all objects required for Plant Modeler are located underneath this node.

**"CELLIB" node**

The following table describes the structure of the "PLM > CELLIB" node:

Subordinate node	Call on the "Base objects" tab	Description
"CEL Plant modeler"	"PLM > CELLIB > CEL"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > CEL" node below.

**"Y" node**

The following table describes the structure of the "PLM > Y" node:

Subordinate node	Call on the "Base objects" tab	Description
"ABO Assignable objects catalog"	"PLM > Y > ABO"	Catalog for objects you can assign.
"BO Microstation Catalog"	"PLM > Y > BO"	MicroStation catalog for base objects required by Plant Modeler
"CATT Microstation catalog attributes"	"PLM > Y > CATT"	Microstation catalog attributes
"CTAB Microstation catalog tabs"	"PLM > Y > CTAB"	MicroStation catalog for tabs.

**"ABO" node**

You create "ABO" objects yourself. You can define these objects as placeholders for MicroStation cells. The objects you create underneath the "ABO" node inherit the tabs of the "ABO" node.

The following table describes the tabs of the objects you create under the "ABO" node:

Tab	Call on the "Base objects" tab	Description
"ABOS Connector Mapping"	"PLM > Y > ABO > <ABO Object> > ABOS"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > ABOS" node below.
"CM3D Plant modeler devices"	"PLM > Y > ABO > <ABO Object> > CM3D"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > CM3D" node below.
"GD Geometry"	"PLM > Y > ABO > <ABO Object> > GD"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > GD" node below.



Tab	Call on the "Base objects" tab	Description
"M3D Plant modeler"	"PLM > Y > ABO > <ABO Object > > M3D"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > M3D" node below.
"SYSISO System Information"	"PLM > Y > ABO > <ABO Object> > SYSISO"	Referenced tab of the "@VIPER > @Y > CHP > PP > A0 > SYSISO > SYSISO" node This tab is used for information about the isometric drawing.

## "BO" node

The following table describes the structure of the "PLM > Y > BO" node:

Subordinate node	Call on the "Base objects" tab	Description
"CELMASER cell"	"PLM > Y > BO > CELMASTER"	Base object for MicroStation cell
"DU 3D object"	"PLM > Y > BO > DU"	Base object for 3D placeholder object required for equipment.
"HS Hanger and support"	"PLM > Y > BO > HS"	Base object for hangers and supports.
"LIB Cell library"	"PLM > Y > BO > LIB"	Base object for MicroStation cell

## "DU" node

The following table describes the tabs of the "PLM > Y > BO > DU" node:

Tab	Call on the "Base objects" tab	Description
"GD Geometry"	"PLM > Y > BO > DU > GD"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > GD" node below.

**"HS" node**

The following table describes the tabs of the "PLM > Y > BO > HS" node:

Tab	Call on the "Base objects" tab	Description
"GD Geometry"	"PLM > Y > BO > HS > GD"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > GD" node below.
"M3D Plant modeler"	"PLM > Y > BO > HS > M3D"	Referenced tab For more information, refer to the description of the "PLM > Y > CTAB > M3D" node below.
"VXC InterfaceCodes"	"PLM > Y > BO > HS > VXC"	Referenced tab of the "@Viper > @Y > CHP > PP > A0 > VXC > VXC" node This tab is used for the interface codes of an object in the pipe parts catalog.

**"CTAB" node**

The following table describes the tabs of the "PLM > Y > CTAB" node:

Tab	Call on the "Base objects" tab	Description
"ABOS connector mapping"	"PLM > Y > CTAB > ABOS"	Catalog tab for base objects you can assign. This tab is used for connector mapping and allows you to reorganize the base objects.
"CEL Plant modeler"	"PLM > Y > CTAB > CEL"	Catalog tab for MicroStation cell objects. This tab stores the name of the cell library and the name of the MicroStation model.
"CM3D Plant modeler devices"	"PLM > Y > CTAB > CM3D"	Catalog tab for Plant Modeler objects. This tab stores the location of the objects in the MicroStation environment. This tab must be linked to the pipe parts catalog so that it can be used by all 3D objects.
"GD Geometry"	"PLM > Y > CTAB > GD"	Catalog tab for the geometry of placeholders
"GD1 Geometry hanger & supports geometry"	"PLM > Y > CTAB > GD1"	Catalog tab for the geometry of hangers and supports.
"M3D Plant modeler"	"PLM > Y > CTAB > M3D"	Plant Modeler objects are identified based on this catalog tab. This catalog tab is linked to the pipe parts catalog.
"M3DP Plant modeler"	"PLM > Y > CTAB > M3DP"	Catalog tab for Plant Modeler objects, especially pipes. This tab is linked to the pipe parts from the pipe parts catalog: "@01 > PID > 01 > 03 > 01"
"MSI Plant Modeler"	"PLM > Y > CTAB > MSI1"	Catalog tab that is linked to the project. This tab is checked every time a user starts Plant Modeler. The tab is displayed in the project properties.

### Attributes of the "ABOS" tab

The following table describes the attributes of the "PLM > Y > CTAB > ABOS" tab:

Attribute	Call on the "Base objects" tab	Description
"Conn connectors"	"PLM > Y > CTAB > ABOS > Conn"	This attribute has the "Frame" display type and has the "Text" type. It is used only as the frame title.
"CX1 CX1"	"PLM > Y > CTAB > ABOS > CX1"	These are "Edit field" display type attributes. All four attributes have the same properties.
"CX2 CX2"	"PLM > Y > CTAB > ABOS > CX2"	
"CX3 CX3"	"PLM > Y > CTAB > ABOS > CX3"	
"CX4 CX4"	"PLM > Y > CTAB > ABOS > CX4"	

### Attributes of the "CEL" tab

The following table describes the attributes of the "PLM > Y > CTAB > CEL" tab:

Attribute	Call on the "Base objects" tab	Description
"E002 Cell library"	"PLM > Y > CTAB > CEL > E002"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the name of the cell library.
"E003 Model"	"PLM > Y > CTAB > CEL > E003"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the name of the model.

### Attributes of the "CM3D" tab

The following table describes the attributes of the "PLM > Y > CTAB > CM3D" tab:

Attribute	Call on the "Base objects" tab	Description
"DGN DGN"	"PLM > Y > CTAB > CM3D > DGN"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the name of the DGN file.
"ID ID"	"PLM > Y > CTAB > CM3D > ID"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the object ID.
"Model Model"	"PLM > Y > CTAB > CM3D > Model"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the name of the model.

### Attributes of the "GD" tab

The following table describes the attributes of the "PLM > Y > CTAB > GD" tab:

Attribute	Call on the "Base objects" tab	Description
"V002 Length"	"PLM > Y > CTAB > GD > V002"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the length of the object in millimeters. The default value is "100".
"V012 Height"	"PLM > Y > CTAB > GD > V012"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the height of the object in millimeters. The default value is "100".
"V050 Width"	"PLM > Y > CTAB > GD > V050"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the width of the object in millimeters. The default value is "100".
"VGEO Geo Type"	"PLM > Y > CTAB > GD > VGEO"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the geo type of the object. The default value is "Prim.Box".

### Attributes of the "GD1" tab

The following table describes the attributes of the "PLM > Y > CTAB > GD1" tab:

Attribute	Call on the "Base objects" tab	Description
"V002 Length"	"PLM > Y > CTAB > GD1 > V002"	For additional information, refer to the description of the "GD" tab attributes.
"V012 Height"	"PLM > Y > CTAB > GD1 > V012"	
"V050 Width"	"PLM > Y > CTAB > GD1 > V050"	
"VGEO Geo Type"	"PLM > Y > CTAB > GD1 > VGEO"	
"VC11 DN1"	"PLM > Y > CTAB > GD1 > VC11"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the diameter of the object.
"VFCD Function code"	"PLM > Y > CTAB > GD1 > VFCD"	This is an "Alphanumeric" type, "Edit field" display type attribute. It is linked to the standard table for function codes.
"VSTD Standards system"	"PLM > Y > CTAB > GD1 > VSTD"	This is an "Alphanumeric" type, "Edit field" display type attribute. It is linked to the standard table for standard systems.
"VSUI"	"PLM > Y > CTAB > GD1 > VSUI"	This is an "Alphanumeric" type, "Edit field" display type attribute. The default value is "VPCL; VC11".

### Attributes of the "M3D" tab

The following table describes the attributes of the "PLM > Y > CTAB > M3D" tab:

Attribute	Call on the "Base objects" tab	Description
"E0003 Label"	"PLM > Y > CTAB > M3D > E003"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the object label. This attribute is read out in MicroStation.
"ST0001 Level"	"PLM > Y > CTAB > M3D > ST0001"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the object level. This attribute ensures that objects in MicroStation are used at the correct level.

### Attributes of the "M3DP" tab

The following table describes the attributes of the "PLM > Y > CTAB > M3DP" tab:

Attribute	Call on the "Base objects" tab	Description
"E0003 Label"	"PLM > Y > CTAB > M3DP > E003"	For additional information, refer to the description of the "M3D" tab attributes.
"ST001 Level"	"PLM > Y > CTAB > M3DP > ST001"	
"E0004 Reference name"	"PLM > Y > CTAB > M3DP > E0004"	This is an "Alphanumeric" type, "Edit field" display type attribute. It specifies the name of the Ustation reference. This attribute contains the information in which reference the object is searched.

### Attributes of the "MSI1" tab

The following table describes the attributes of the "PLM > Y > CTAB > MSI1" tab:

Attribute	Call on the "Base objects" tab	Description
"B0001 Re-initialize"	"PLM > Y > CTAB > MSI1 > B0001"	This is a "Text" type, "Button" display type attribute. This button updates the Plant Modeler module.
"CPM0001 Plant Modeler start node"	"PLM > Y > CTAB > MSI1 > CPM0001"	This is an "Alphanumeric" type, "Pointer" display type attribute. This field references the "PLM" node in the base data.
"CPM002 pipe spec limit"	"PLM > Y > CTAB > MSI1 > CPM0002"	This is an "Alphanumeric" type, "Pointer" display type attribute. The field references a structure object.
"CPM0003 part unit structure"	"PLM > Y > CTAB > MSI1 > CPM0003"	This is an "Alphanumeric" type, "Pointer" display type attribute. The field references a structure object.

Attribute	Call on the "Base objects" tab	Description
"CPM0004 KKS system count"	"PLM > Y > CTAB > MS11 > CPM0004"	This is an "Alphanumeric" type, "Pointer" display type attribute. The field references a structure object.
"CPM0005 PPC start node"	"PLM > Y > CTAB > MS11 > CPM0005"	This is an "Alphanumeric" type, "Pointer" display type attribute. This field references the "PPC" node in the base data.
"CPM0006 cell library start node"	"PLM > Y > CTAB > MS11 > CPM0005"	This is an "Alphanumeric" type, "Pointer" display type attribute. The field references the start node underneath which the cell libraries are located in the base data.
"L0002 Default location for pipes"	"PLM > Y > CTAB > MS11 > L0002"	This is an "Alphanumeric" type, "Pointer" display type attribute. The reference indicates the default location for pipes. If this location is not set, a selection window appears if a user tries to route pipes without an owner.
"L0003 Default location for equipment"	"PLM > Y > CTAB > MS11 > L003"	This is an "Alphanumeric" type, "Pointer" display type attribute. This button is used to specify the default location for equipment in the project properties.

## 6.2 COMOS PDMS Integration

### 6.2.1 INI file

The complete INI file is in XML format and has the following structure:

```
<ComosPDMSInterfaceConfiguration>
  <CommunicationFolder path="<path>"/>
  <ExchangeFolder path="<path>"/>
  <SubclassesFile path="<path>"/>
  <TalkToComos path="<path>"/>
  <Logfile path="<path>"/>
  <DocumentFolder path="<path>"/>
  <MTOTransferConfigurationFile path="<path>"/>
  <StartupFunctionsFile path="<path>"/>
  <ClientCommandService path="<path>"/>
  <StartClientCommandService value="<Boolean>"/>
</ComosPDMSInterfaceConfiguration>
```

The following nodes are included:

Node	Description
CommunicationFolder	<ul style="list-style-type: none"> <li>• Mandatory information</li> <li>• Directory containing the XML files for command exchange. Created automatically, if not already present.</li> </ul>
ExchangeFolder	<ul style="list-style-type: none"> <li>• Mandatory information</li> <li>• Directory containing the XML files that are used for data exchange. Created automatically, if not already present.</li> </ul>
SubclassesFile	<ul style="list-style-type: none"> <li>• Mandatory information</li> <li>• Path to the Subclasses.xml configuration file</li> </ul>
TalkToComos	<ul style="list-style-type: none"> <li>• Mandatory information</li> <li>• Path to talkToComos.exe</li> </ul>
Logfile	<ul style="list-style-type: none"> <li>• Mandatory information</li> <li>• Path to the log file</li> </ul>
DocumentFolder	<p>Path to the document directory</p> <p>If this path is not specified, you cannot export any documents from the draft module. Created automatically, if not already present.</p>
MTOTransferConfigurationFile	<p>Path to file MTOExportSettings.dat. The file name here is an example only; you can choose any file name.</p> <p>If this path is not specified, you cannot transfer any MTO data to COMOS.</p> <p>You can find additional information on this topic in the "COMOS Material Management Administration" manual, keyword "MTOExportSettings.dat".</p>
StartupFunctionsFile	Path to the file containing the StartupFunctions
ClientCommandService	Path to ClientCommandService.exe
StartClientCommandService	<p>Flag which indicates whether ClientCommandService.exe is to be executed when AVEVA PDMS starts</p> <p>Possible values:</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>

## 6.2.2 Communication process

### 6.2.2.1 Communication modes

#### Directed communication process

How COMOS and PDMS communicate with one another depends on the application in which the communication process starts:

- From PDMS to COMOS
- From COMOS to PDMS

#### Direct communication and file pipe communication

If communication starts in COMOS, the communication process is also determined by the installation modes of PDMS and COMOS.

- Direct communication: Both applications run locally on the same workstation.
- File pipe communication: COMOS runs on a Citrix server.

### 6.2.2.2 COMOS to PDMS

#### General rules

COMOS uses commands to communicate with PDMS. These commands are sent to the PDMS command line via the Windows Messaging Service.

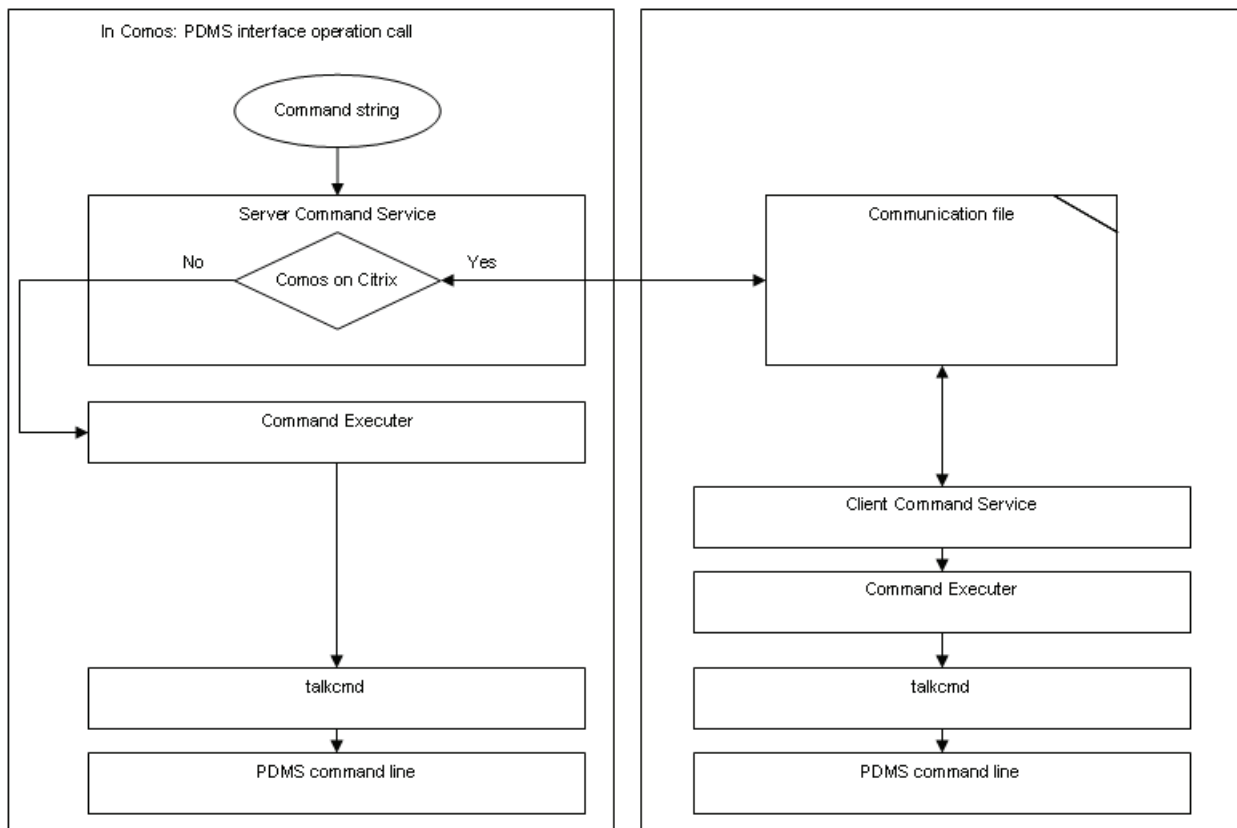
All commands which go from COMOS to PDMS call a function in PDMS. They generate no PML command that are directly send to PDMS.

This procedure has the advantage that COMOS does not work with PML. Changes to PML by AVEVA do not affect the COMOS side of the interface.



## Process

The following diagram provides you with an overview of how the communication process works when it starts in COMOS. The diagram maps the process for direct communication and file pipe communication:



## Setup details

The following requirements apply for direct communication and file pipe communication:

- The "talkcmd" window is active in PDMS.
- In the wait state, the window is called "Comos Command Interface".

The following requirements also apply for file pipe communication:

- The same value has been entered in the "PDMSCmdFile" system environment variable on the client and the server.

The variable contains the path name and the name of the file COMOS and PDMS use for communication, and which is monitored by the COMOS Server Command Service and the Client Command Service.

Example:

- Server: "\\client\c\$\filename"
- Clients: "c:\filename"

- The Client Command Service is running on the client.

Name: "PDMSToCitrix.exe"

Name of window: "Comos Command Service"

- In PDMS:

The PDMS environment variable "COMOSEXCHANGEPATH" has been created.

The variable specifies the default path to the exchange file.

The same value was entered in the variable as in the following COMOS project property:

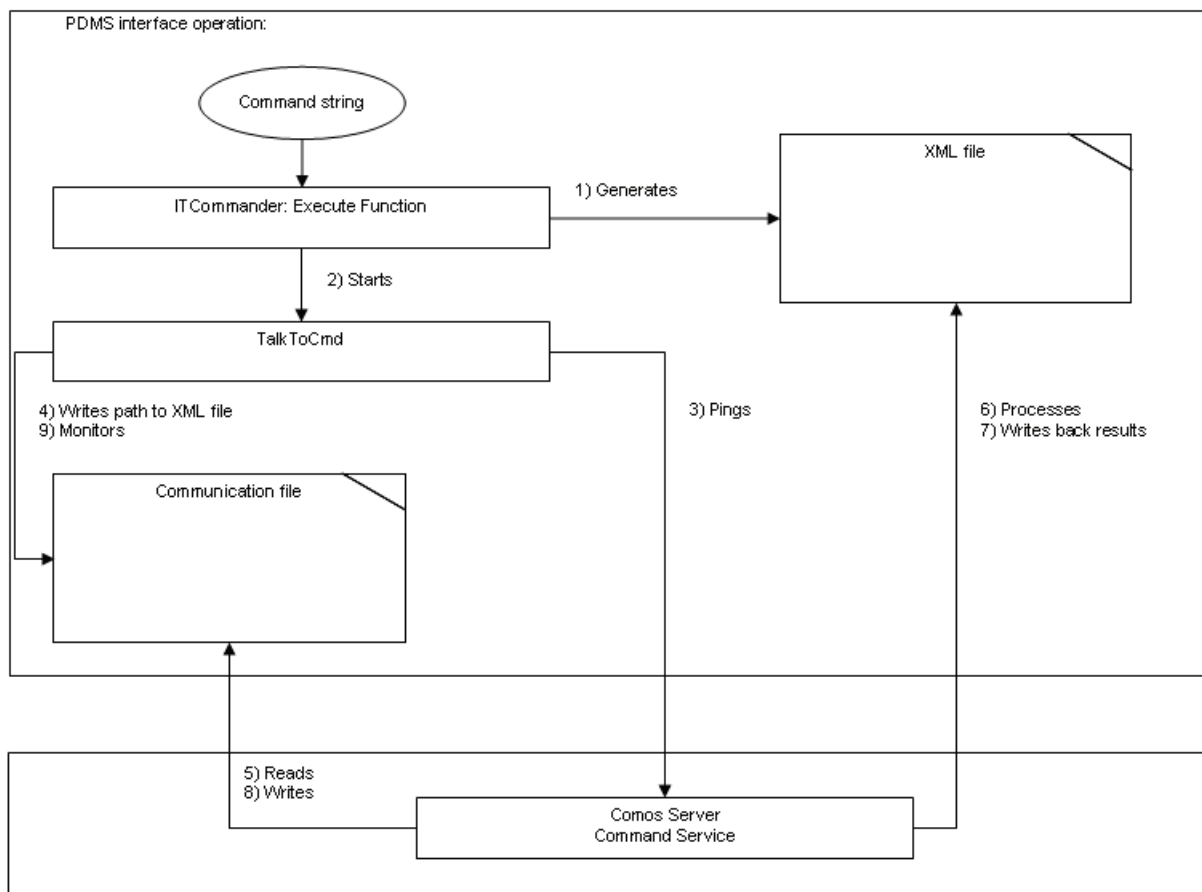
Project properties, "PDMS interface/construction assistant" tab, "Exchange directory" field

### 6.2.2.3 PDMS to COMOS

#### Process

If communication starts in PDMS, the interface does distinguish between direct communication and file pipe communication.

The following diagram provides you with an overview of the communication process if it starts in PDMS:



- When an interface operation is called in PDMS, the ITCommander creates an XML file and stores it in the exchange path.
- It also creates a communication file and saves it in the communication path. The communication path is continuously monitored by both applications. This tells COMOS that there is an XML file in the exchange path.
- COMOS executes the interface operation and processes the XML file.
- COMOS writes the results of the interface operation to the XML file.
- COMOS writes an answer to the communication file. This tells PDMS that the results of the interface operation are available for further processing.

### Setup details

The following applies for the workstation on which PDMS is installed:

- TalkToCmd has been installed on the same workstation as PDMS.
- The instance of the ITCommander knows the path to "TalkToCmd.exe": The path has been specified in the ITCommander's "ExePath" variable.

6.2 COMOS PDMS Integration

- The PDMS environment variable "COMOEXCHANGE\_PATH" has been created. The variable specifies the default path for the exchange file.  
The same value was entered in the variable as in the following COMOS project property:  
Project properties, "PDMS interface/construction assistant" tab, "Exchange path" field
- You can overwrite the default path for the exchange file generated by the ITC Commander:
  - Default path: As specified in the PDMS environment variable "COMOEXCHANGE\_PATH".
  - New path and name: Enter the new path and the new name in the "FileName" variable of the ITC Commander.
  - Requirement: You have write rights.
- The "PDMSCmdFile" system environment variable has been set. The variable tells the COMOS Server Command Service the file via which COMOS and PDMS communicate.  
The following value has been written to the variable:
  - COMOS runs on a Citrix server: The variable points to the same file as on the client: "\\client\c\$\filename".
  - COMOS is running on the same workstation as PDMS: The variable has already been set in the context of TalkToCmd
- The COMOS Server Command Service is running.

6.2.3 Database

6.2.3.1 Structure of the standard table for PDMS element types

Structure of the "@3D > 00 > PDMS > 14 PDMS element types" standard table

Column name	Function
"Name"	<ul style="list-style-type: none"> <li>• Index number</li> <li>• Used internally</li> <li>• Starts with 1</li> </ul>
"Description"	Is displayed as a value in the interface for attributes to which the standard table has been assigned.
"Value 1"	The PDMS element type Notation as in PDMS
"Value 2"	PDMS module in which the PDMS element type is used. Possible values: <ul style="list-style-type: none"> <li>• "Design"</li> <li>• "Draft"</li> </ul>

**See also**

Maintaining the standard table for PDMS element types (Page 98)

**6.2.3.2 Structure of the standard table for classes****Structure of the "@3D > 00 > PDMS > 08 Classes" standard table**

Column name	Function
"Name"	<ul style="list-style-type: none"> <li>• Index number</li> <li>• Used internally</li> <li>• Starts with 1</li> </ul>
"Description"	Is displayed as a value in the interface for attributes to which the standard table has been assigned.
"Value 1"	<ul style="list-style-type: none"> <li>• Class name</li> <li>• Used system-internally</li> </ul>
"Value 2"	Letter code: The letter entered here is used in the standard table for subclasses to assign a class to a subclass.

**See also**

Standard tables for classes and subclasses (Page 99)

**6.2.3.3 Structure of the standard table for subclasses****Structure of the "@3D > 00 > PDMS > 08 > 01 Subclasses" standard table**

Column name	Function
"Name"	<ul style="list-style-type: none"> <li>• Index number</li> <li>• Used internally</li> <li>• Starts with 1</li> </ul>
"Description"	Is displayed as a value in the interface for attributes to which the standard table has been assigned.
"Value 1"	<ul style="list-style-type: none"> <li>• Subclass name</li> <li>• Used system-internally</li> </ul>
"Value 2"	Letter code which assigns a class to the subclass. Only letters entered in the class standard table in the "Value 2" column are permitted.

**See also**

Standard tables for classes and subclasses (Page 99)

**6.2.3.4 Attribute properties****No configuration necessary**

The properties of the attributes listed in this reference have already been fully configured in the COMOS DB.

Only change the preconfigured settings if the documentation explicitly indicates that such a change is permitted or necessary.

In particular:

- Do not change or modify any scripts.
- Do not modify or delete any links.
- Do not delete standard table assignments.

**6.2.3.5 Attributes of the "@PDMSMAP" folder****"General" tab**

Name of tab	Description
"MAP001"	"General"

**Content**

The tab contains the following attributes:

Name	Description	Purpose
"FileVersion"	"Class/subclass version"	<ul style="list-style-type: none"> <li>• Used system-internally</li> <li>• Do not configure or delete</li> <li>• Saves the current version number of the PDMSMAP folder and of the class and subclass definition objects.</li> </ul> <p>If the user has selected the following command in the context menu of the "@PDMSMAP" folder, the number is incremented and written to an exchange file together with the configuration settings:</p> <p>"PDMSMAP &gt; Save Subclasses"</p>
"MAPITEM001"	"Use site and zone mapping"	<p>Option:</p> <p>Activates site and zone mapping.</p>

Name	Description	Purpose
"MAPITEM002"	"Subclass for pipe"	Field: Application case: COMOS works with a two-stage KKS structure which uses pipe branches and segments. Enter the subclass which is used to create or find the pipes in PMS under which the branches are located. Configure the here entered subclass as follows: <ul style="list-style-type: none"> <li>• Creation mode and assign mode: "None"</li> </ul> or <ul style="list-style-type: none"> <li>• Creation mode: "Create in location tree"</li> </ul> and Assign mode of the subclass for pipe branches: "Assign to object in location tree" and Location mapping of the subclass for pipe branches: Addresses objects of the subclass entered in the "Subclass for pipes" subclass in the object tree
"MTORevision"	"Use MTO revisioning"	Do not use
"OperationInfo"	"Operation messages"	Option: Activates or deactivates operation messages.

**See also**

Synchronizing settings (Page 135)

**"Name directory" tab**

Name of tab	Description
"MAP006"	"Name directory"

**Content**

The tab contains the following table:

Name of the table	Description of the table	Purpose
"GlobNameDir"	"Name directory"	Each line in the table defines a String Parameter. The String Parameters define variables which are globally available in the interface operations area. You configure an assignment between COMOS attributes and PDMS attributes in the subclass definition objects, not at the "@PDMSMAP" folder.

**See also**

Defining global variables (Page 100)

**"Pre/post executables" tab**

Name of tab	Description
"MAP007"	"Pre/post executables"

**Content**

The tab contains the following tables:

Name of the table	Description of the table	Purpose
"PreExecutable"	"Pre executables"	Registers the PML functions which are called before an interface operation is executed.
"PostExecutable"	"Post executables"	Registers the PML functions which are called after an interface operation has been executed.

You find information on the structure of the table and the configuration of the tab in section Using pre/post executables (Page 101).

**"Character mapping" tab**

Name of tab	Description
"MAP002"	"Character mapping"

**Content**

The tab contains the following table:

Name of the table	Description of the table	Purpose
"MAPLIST"	/	Defines which characters are permitted in COMOS but are illegal in PDMS and are replaced during name mapping.

You find information on the structure of the table and the configuration of the tab in section Using character mapping (Page 102).

**"Units mapping" tab**

Name of tab	Description
"MAP005"	"Units mapping"



## Content

The tab contains the following table:

Name of the table	Description of the table	Purpose
"UNITSMAP"	/	Defines how units in PDMS and COMOS are mapped to one another if a different unit has been assigned to an attribute in COMOS than in PDMS.

You find information on the structure of the table and the configuration of the tab in section Using units mapping (Page 103).

## "Site mapping" and "Zone mapping" tabs

Name of tab	Description
"MAP003"	"Site mapping"
"MAP004"	"Zone mapping"

## Content

The tabs contain the following tables:

Tab	Name of the table	Description of the table
"Site mapping"	"SITEMAP"	"Mapping"
"Zone mapping"	"ZONEMAP"	"Mapping"

You find information on the structure of the tables and the configuration of the tabs in section Using site mapping and zone mapping (Page 104).

### 6.2.3.6 Attributes of the subclass definition objects in the "Pipe" and "TaggedItem" classes

#### "General" tab

Name of tab	Description
"MAP001"	"Name directory"

#### System-internal attributes

The following attributes are used system-internally. Do not configure or delete them.

- "PDMSModule PDMS module usage"
- "PDMS002 class"
- "PDMSSubClass PDMS subclass"

## Remaining attributes

The following sections describe the attributes of the tab and its configuration:

- Section Defining PDMS element types (Page 106)
- Section Defining structural behavior (Page 107)
- Section Defining the base object for the creation of interface objects (Page 108)

## See also

Algorithm for generating a COMOS object through name mapping (Page 123)

## "Name directory" tab

Name of tab	Description
"MAP002"	"Name directory"

## Content

The tab contains the following table:

Name of the table	Description of the table	Purpose
"MapAttributes"	"Name directory"	Definition of String Parameters

You find information on the structure of the table and the configuration of the tab in section Working with String Parameters in the name directory (Page 112).

## "Name mapping" tab

Name of tab	Description
"MAP003"	"Name mapping"

## Content

The tab contains the following tables:

Name of the table	Description of the table	Purpose
"UNITMAP"	"Mapping"	Table which defines unit mapping.
"LOCMAP"	"Mapping"	Table which defines location mapping.
"LevelRules"	"Level rules"	Table which defines deviations from the standard procedures for unit mapping and location mapping.

Information on the structure and configuration of the tables: See the links below.

**See also**

Name mapping (Page 85)

Configuring level rules (Page 127)

Configuring unit and location mapping in name mapping (Page 119)

**"Owner restriction rules" tab**

Name of tab	Description
"MAP004"	"Owner restriction rules"

**Content**

The tab contains the following attributes:

Name	Description	Purpose
"PDMSOwnerType"	"Element type of owner"	List in which you define the element type of the owner of the connected PDMS object.
"FILTER001"	"Expressions"	Table in which you define expressions which have to be evaluated to "True" at the owner.
"FILTER002"	"Filter"	Table in which you define filters which check whether a given owner attribute contains a specific occur string.

You find information on the structure of both tables and the configuration of the tab in section Configuring owner restriction rules (Page 109).

**"Model" tab**

Name of tab	Description
"MAP005"	"Model"

**Content**

The tab contains the following attributes:

Name	Description	Purpose
"PDMSModel"	"Mode"	List in which you specify the model type.
"PDMSFunction"	"Function"	Field in which you enter the function call to generate the model.
"PDMSElement"	"Element"	Field in which you enter the name of the PDMS element whose model is copied.

Name	Description	Purpose
"PDMS004"	Main template	List from which you select a design template for the main equipment. The entries in the list are filtered based on the entries in "TMPFILTER". The list is empty if the entry "EQUI" is missing in "TMPFILTER".
"PDMS005" through "PDMS009"	"Sub template"	The design templates for the secondary equipment. The entries in the list are determined by the main template set in "PDMS004".
"TMPLFILTER"	"GTypes for filtering design templates"	Field in which you enter the GTypes of the design templates which are offered for this subclass Delimiter: ";"

**See also**

Configuring the model (Page 111)

**6.2.3.7 Attributes of the subclass definition objects of the "Query" class****The "General" tab**

Name of tab	Description
"MAP001"	"General"

**Content**

The tab contains the following attributes:

Name	Description	Purpose
"PDMS settings" control group		
"PDMSQueryName"	"Name"	Field for the name which users will see when the query is offered for execution in PDMS.
"PDMSQueryDesc"	"Description"	Field for the description with which the query is displayed in PDMS once the user has selected it for execution.
"Output format" control group		
"OutputFormat"	"Output format"	List for the output format in which the result list of the query is transferred to PDMS.
"OutputDelimiter"	"Separator"	Field for the separator. Only set if the output format is "CSV". The separator can be chosen at will.
"Main query" control group		

Name	Description	Purpose
"MainQuery"	"Name"	Reference to the query object whose content is to be displayed in PDMS.
"StartObjectMQ"	"Start object"	Optionally: Reference to a start object. Overwrites the start object set in the query referenced in the "Name" field.
"BaseObjMQ001"	/	Only in the Navigator. Used system-internally
"BaseObjectsMQ"	"Base objects"	Optionally: References to the base objects according to which the query filters the result list. You can set other base objects than those set in the query referenced in the "Name" field.
"Start object query" control group		
"StartObjectQuery"	"Name"	Reference to the query returned by the list of possible start objects for the main query in PDMS.
"StartObjectSQ"	"Start object"	Optionally: Reference to a start object for the start object query. Overwrites the start object set in the query referenced in the "Name" field.
"BaseObjSQ001"	/	Only in the Navigator. Used system-internally
"BaseObjectsSQ"	"Base objects"	Optionally: References to the base objects according to which the query filters the result list. See "BaseObjectsMQ".
Miscellaneous attributes		
"PDMSModule"	"PDMS module usage"	Used system-internally

**See also**

Configuring a subclass definition object for queries (Page 129)

**6.2.3.8 Attributes of the subclass definition objects of the "Document" class****"General" tab**

Name of tab	Description
"MAP001"	"General"

## Content

The tab contains the following attributes:

Name	Description	Purpose
"PDMSDocID"	"Document name"	Field for the name of the String Parameter which, from PDMS, reads out the name of the COMOS document under which the DocLinks are imported.
"Base object for document search" control group		
"PDMSDocType"	"Document type"	List of document types. Used as a filter to speed up the search for the COMOS document.
"PDMSBaseObject"	"Base object"	Reference to the base object of the COMOS document. Used as a filter to speed up the search for the COMOS document.
"PDMSStartObject"	"Start object"	Reference to the start object under which the search for the COMOS document is carried out. Used as a filter to speed up the search for the COMOS document.
Miscellaneous attributes		
"PDMSFilterUnnamed"	"Filter unnamed objects"	Option to filter the result list in the "Export to Comos" window. Activated: The result list only contains objects with names.
"PDMSIdentifyOwner"	"Identify owner"	Do not use.
"PDMSElementTypes"	"PDMS element types"	List of PDMS element types permitted for the draft objects for which DocLinks are created.
System-internal attributes		
"PDMSModule"	"PDMS module usage"	Used system-internally
"PDMSSubclass"	/	Used system-internally

## See also

Configuring "Import DocLinks" (Page 132)

## "Name directory" tab

Name of tab	Description
"MAP002"	"Name directory"

## Content

The content of the tab is exactly the same as that of the tab of the same name for the subclass definition objects of the "Pipe" and "TaggedItem" classes.

**"Owner restriction rules" tab**

Name of tab	Description
"MAP004"	"Owner restriction rules"

**Content**

The content of the tab is exactly the same as that of the tab of the same name for the subclass definition objects of the "Pipe" and "TaggedItem" classes.

**6.2.3.9 Attributes of the interface objects****Attributes of the "External 3D Interface" tab**

Name of tab	Description
"E3D"	"External 3D Interface"

**Function**

All COMOS interface objects need the "External 3D Interface" tab.

COMOS PDMS Integration evaluates this tab whenever it processes interface objects. Without this tab, the object is not processed.

**COMOS DB**

In the COMOS DB, tabs have been prepared for the following interface objects:

- Objects of the "TaggedItem" class
- Objects of the "Pipe" class
- For nozzles

The tabs for these objects are located in the following catalog: "@PDMS > @Y > @CHP > 01 PI object tabs"

From here they are inherited by the PID device catalog in the COMOS DB.

**Customer database**

Import the tabs into your database and add them to the device catalog.

### System-internal attributes

The following attributes are used system-internally. Do not configure or delete them:

- "ComosCheckUID Check SystemUID"
- "PDMSNAME Name of the corresponding object in PDMS"
- "PDMSRefNo Reference number"

### Remaining attributes

Define the following properties of the interface object on the "External 3D Interface" tab:

- The class and subclass
- The PDMS element type
- The model of the object

Details regarding the corresponding attributes and their configuration: See the links below.

### See also

Assigning the class, subclass, and PDMS element type (Page 137)

Overwriting the inherited model information (Page 138)

### 6.2.3.10 Attributes of the design template objects

#### Attributes of the "External 3D Interface" tab

Name of tab	Description
"E3D"	"External 3D Interface"

### Content

The tab contains the following attributes:

Name	Description	Purpose
"PDMSGTYPE"	"GTYPE"	Field in which the GType of the design template is entered.
"PDMSPurpose"	"PURPOSE"	Field in which the purpose of the design template is entered.
"Design parameters" control group		
"PDMSDP01" through "PDMSDP10"	"01:" through "10:!"	Field in which the value of the corresponding design parameter is entered. Can be modified by the user in the engineering data.



Name	Description	Purpose
"PDMSDPName01" through "PDMSDPName10"	/	Field for the description of the design parameter from PDMS.
The following attributes are dynamically created while the template object is being created in the engineering data:		
"PDMSSUPP<x>_<y>"	/	Is only generated if one or more design templates for secondary equipment were specified for an interface object: <ul style="list-style-type: none"> <li>"&lt;x&gt;": Defines which of the five possible sub templates the current sub template belongs to.</li> <li>"&lt;y&gt;": Several templates are available for selection for each item of secondary equipment. "&lt;y&gt;" designates the template to which the attribute refers.</li> <li>Value: The Catref of the sub template selected at the interface object.</li> </ul>

### 6.2.3.11 Project parameters

#### "PDMS interface/construction assistant" tab

Name of tab	Description
"CA"	"PDMS interface/construction assistant"

#### Content

The tab defines the following project parameters:

Name	Description	Purpose
"Project and profile folders" control group		
"IniFile" field	"Initialization file"	Directory containing the INI file.
"CaProjectFolders"	"Select initialization file"	Button Opens the "Interface configuration profiles" window in which you can enter the directory of the INI file.
"Directory interfaces" control group		
"CAExchangePath"	"Exchange directory"	These fields cannot be edited. Their content is taken from the configuration file.
"CAComPath"	"Communication directory"	
"CADocPath"	"Document directory"	
"Object class interfaces" control group		

Name	Description	Purpose
"CAConfigPath"	"Configuration path"	The configuration path cannot be edited. It is taken from the configuration file.
Status of the interface		
"Acti"	"Activate"	Button to activate Comos PDMS Integration.
"Deac"	"Deactivate"	Button to activate COMOS-PDMS Integration.

**See also**

Project properties (Page 76)

## 6.3 COMOS 3D viewing

### 6.3.1 Sample scripts

#### 6.3.1.1 Sub OnProjectOpen(Project)

```
Sub OnProjectOpen(Project)

'Check if Comos3DView already is running
If Workset.Globals.ItemExist("Comos3DView") Then
    If IsObject(Workset.Globals.Comos3DView) Then
        Set c3d = Workset.Globals.Comos3DView
        'msgbox "3D View already present"
    End If
End If

'Check if Comos3DView has to be created new
bCreatec3d = False
bCreatec3d = isEmpty(c3d )

If Not bCreatec3d Then
    If IsObject(c3d ) Then
        If c3d Is Nothing Then bCreatec3d = true
    End If
End If

'Create new Comos3DView
If bCreatec3d Then
    Set c3d = Createobject("Comos.C3DView.Comos3DView")
    If Not c3d Is Nothing Then
        c3d.Init(Workset)
        Workset.Globals.Comos3DView = c3d
        'msgbox "New 3D View present"
    End If
Else
    'msgbox "No 3D View present"
End If

End Sub
```

#### See also

Script adjustments (Page 141)

### 6.3.1.2 Exchange directory

```
<configuration>  
  <appSettings>  
    <add key="WorkingDirectory" value="C:\exchange\" />  
    <add key="DEBUG" value="1" />  
  </appSettings>  
</configuration>
```

#### Parameter description

Parameter	Description
WorkingDirectory	Specifies the directory in which communication with the 3D application is to take place. The specified directory must already exist.
DEBUG	Specifies whether an extended debug output is to be performed. The default value is 0, which indicates no debug output is to be performed. If you select the value 1, the XML messages being exchanged between COMOS and the 3D application are written to the debug output.

#### See also

Script adjustments (Page 141)

## 6.4 COMOS NX - Routing Mechanical interface

### 6.4.1 Sample scripts

#### 6.4.1.1 Sub OnProjectOpen(Project)

```
Sub OnProjectOpen(Project)

'Is called on opening a project
On Error Resume Next

If Workset.Globals.ItemExist("NXViper") Then
    If IsObject(Workset.Globals.NXViper) Then
        Set NXV = Workset.Globals.NXViper
    End If
End If

bCreateNXV = False
bCreateNXV = isEmpty(NXV)
If Not bCreateNXV Then
    If IsObject(NXV) Then
        If NXV Is Nothing Then bCreateNXV = true
    End If
End If

If bCreateNXV Then
    Set NXV = CreateObject("Comos.NXViper.ComosNXRoutingMechanical")
    If Not NXV Is Nothing Then
        NXV.Init(Workset)
Workset.Globals.NXViper=NXV
        'Output "created new NXV"
    End If
Else
    'Invoke "Disconnect" if, NXV is already existing.
    NXV.Disconnect
    'Output "use existing NXV "
End If

End Sub
```

#### See also

Default settings in COMOS (Page 143)

## 6.4.1.2 COMOS exchange directory

## Sample script

```

<configuration>
  <appSettings>
    <add key="StartInterface" value="1" />
    <add key="WorkingDirectory"
value="C:\temp\ComosExchangeDir\" />
    <add key="DEBUG" value="true" />
    <add key="ExportDirectory" value="C:\exportToNX\" />
    <add key="FilterEmptyExportValues" value="false" />
    <add key="ExportVersion" value="120" />
  </appSettings>
</configuration>

```

## Parameter description

Parameter	Description
WorkingDirectory	Specifies the directory in which communication with NX is to take place. The specified directory must already exist.
DEBUG	Specifies whether an extended debug output is to be performed. The default value is <code>false</code> , which indicates no debug output is to be performed. If you select the value <code>true</code> , the XML messages being exchanged between COMOS and the 3D application are written to the debug output. Error messages which occur in the module are displayed.
ExportDirectory	Specifies where the PTB file is to be saved. If the specified path does not exist, COMOS attempts to create it. If no path has been specified, the PTB files are saved in the exchange directory.
FilterEmptyExportValues	If the parameter is set to <code>false</code> , the geometric data is written, even if no valid value could be read out of the COMOS catalog. The string "n.n." is written, rather than an empty value. If the key is missing, the default value is <code>false</code> . This means that none of the rows of nominal diameters which contain an undefined value are transferred to the export file. This may result in an empty geometry table.
ExportVersion	A string written in the header of the PTB file. It sets the version of the export file. The default is "120".

## Exchange directory

Values	<p>You can freely select the path to the exchange directory. The file names are predefined:</p> <ul style="list-style-type: none"><li>• File name for messages from COMOS: "CMessage.xml"</li><li>• File name for messages from NX: "NXMessage.xml"</li></ul>
--------	---

## See also

Default settings in COMOS (Page 143)

## 6.4.2 User interface reference

## 6.4.2.1 "NX 3D object mapping" tab

**"General part information" control group**

Control element	Description
"NXNameGen" field	<p>Member name of the component being exported. Specify how the name will be generated in the export file. The individual parts of the name are separated by "+".</p> <p>Example: "VL"+NPS+OutsideD1+Standard"</p> <ul style="list-style-type: none"> <li>• "xxx"</li> </ul> <p>Everything within the quotation marks is recognized as a string. The string is not changed and is transferred exactly as it appears in the attribute for the purposes of generating the member name.</p> <ul style="list-style-type: none"> <li>• "NPS"</li> </ul> <p>Constant value for the nominal diameter for the path created.</p> <ul style="list-style-type: none"> <li>• "NPS_2"</li> </ul> <p>Constant value for the second nominal diameter (branch or deviating value)</p> <ul style="list-style-type: none"> <li>• "NXAttributes"</li> </ul> <p>The "NX Viper geometry mapping" table contains several NX attributes, along with their COMOS assignment. Every assigned NX attribute can be used for name generation purposes. You simply add the string from the "NX attributes" column.</p>
"Part Number" field	<p>Part number of the exported component. This is usually the same as the member name. Structure with the same key words as NXNameGen.</p>



Control element	Description
"Part Name" field	<p>Name of the component and of the entire PTB file. Is not dependent on the nominal diameter. Can be adapted with the following values:</p> <ul style="list-style-type: none"> <li>"xxx"</li> </ul> <p>Everything within the quotation marks is recognized as a string. The string is not changed and is transferred exactly as it appears in the attribute for the purposes of generating the part name.</p> <ul style="list-style-type: none"> <li>"Systemuid"</li> </ul> <p>SystemUID for the part to be exported</p> <ul style="list-style-type: none"> <li>"Systemfullname"</li> </ul> <p>SystemFullName for the part to be exported</p>
"NXMotherprt" field	Only one string can be specified in this version.

### "Texts and properties" control group

Both tables are read out and written to the PTB file.

The "NXTextMap" table shows the COMOS text attributes in the various languages.

The "NXPropMap" table shows the object properties without a geometric reference.

### "NX Viper geometry mapping" control group

Control element	Description
"NX attributes" column	Attribute name
"Check" column	Object must be checked. Value: "true" or "false"
"Hide" column	Object must be hidden. Value: "true" or "false"
"Units name" column	Name of the unit. Example values: "string", "real"

These parameters are evaluated and written to the "COLUMNS" area of the PTB file.

### PTB file

The following values are always written to the PTB file:

Value in the PTB file	Description
MEMBER_NAME	Member name
NPS/NPS_BRANCH	Nominal diameter
PART_NUMBER	Part number
PART_NAME	Part name

If mapping between NX attributes and COMOS attributes is used, the geometric values are evaluated.

COMOS attributes are evaluated for every nominal diameter and written to the DATA area.

**See also**

Default settings in COMOS (Page 143)