

COMOS

Platform iDB Administration

Programming and Operating Manual

Objective and requirements	1
Searching for objects and data in the iDB	2
Standardized designations for folder objects	3
Standardized designations for base objects	4
Generally-applicable writing rules in standardized descriptions	5
Localization rules for iDB texts	6
COMOS fundamentals of base objects	7
Administering "@10 Requirement objects"	8
"@20 General base objects": Editing basic base objects	9
"@30 Structures": Editing module-specific structures	10
"@40 @Y Attribute catalog": Editing attributes and tabs	11
"@50 Manufacturer devices": Editing the manufacturer catalogs	12
"@99 System settings": Editing default lists and settings	13
Administration of standard tables	14
Administering units	15
Administration of the labeling system and ALIAS labeling system	16

Platform iDB Administration

Programming and Operating Manual

Continued

<u>Classifying objects</u>	17
<u>Administration of documents in the iDB</u>	18
<u>Developing scripts for the iDB</u>	19
<u>Editing project properties</u>	20
<u>Customer-specific additions</u>	21
<u>iDB-specific rules for using external files</u>	22
<u>Using case variants as design cases or editing them</u>	23
<u>Administration of templates</u>	24
<u>Database update</u>	25

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.

 DANGER

indicates that death or severe personal injury will result if proper precautions are not taken.
--

 WARNING
--

indicates that death or severe personal injury may result if proper precautions are not taken.

 CAUTION
--

indicates that minor personal injury can result if proper precautions are not taken.
--

NOTICE

indicates that property damage can result if proper precautions are not taken.
--

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:

 WARNING
--

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

1	Objective and requirements	13
2	Searching for objects and data in the iDB	15
2.1	Overview: Search techniques for the iDB	15
2.2	Using "Attribute descriptions DE/EN" query	17
2.3	Finding attributes with "(@40) @Y Attributes and tabs"	18
2.4	Using the "Tab descriptions DE/EN" query	19
3	Standardized designations for folder objects	21
3.1	Names and meaning of the main levels in the iDB	21
3.2	Names of additional levels	23
3.3	Special sub-branches with Y prefix	24
3.4	The @Local branch for local base objects	24
4	Standardized designations for base objects	27
4.1	Overview of the naming convention in the iDB	27
4.2	Uniqueness of names	28
4.3	System characters and special characters	28
4.4	Names of base object elements	29
4.5	Names in the labeling system	29
4.6	Special cases when using letters	30
4.7	Base objects with fixed names	31
4.8	Entering norm-specific text masks	32
5	Generally-applicable writing rules in standardized descriptions	33
6	Localization rules for iDB texts	35
7	COMOS fundamentals of base objects	37
7.1	"Object locking" and "System object" state	37
7.2	Creation option for base objects	39
7.3	Adding tabs to base objects	39
7.4	Design guidelines for tabs	40
7.5	Creating elements at base objects	40
7.5.1	Administering properties of elements in the base object	40
7.5.2	Constructing context menus via elements	43
7.5.3	Constructing context menus via subelements	44
7.6	Forming a labeling system with elements	45

7.7	Prohibited change for the target of cross-inheritances.....	45
7.8	Inserting customer-specific base objects in the iDB.....	46
7.9	Controlling base object links using dereferencing.....	46
7.10	Standard icon of an object class.....	47
7.11	Setting the object icon manually.....	47
7.12	Overview for administration of symbols.....	50
7.13	Drawing type-specific symbol display.....	50
7.14	Using generic symbol with variable number of connectors.....	51
7.15	Preparing symbols for AutoCAD export.....	53
7.16	Connector types in the iDB.....	53
7.16.1	Purpose of the connector types.....	53
7.16.2	Reference: Names of the connector types.....	54
7.16.3	Connectors on EI&C reports.....	58
7.17	General object class.....	58
7.18	Deleting objects.....	59
7.19	Note on obsolete base data.....	59
8	Administering "@10 Requirement objects".....	61
8.1	Purpose of the "@10" node.....	61
8.2	Structure of the "@10" node.....	61
8.3	Editing options of the "@10" node.....	62
8.4	Use/expansion of the "@10" node.....	63
8.5	Notes on expansion of tabs.....	63
8.6	Example for use of the "@10" node.....	64
8.7	Notes on multiple use of "@10" objects.....	65
9	"@20 General base objects": Editing basic base objects.....	67
9.1	Purpose of @20.....	67
9.2	Interface objects for base data import (dynamic objects).....	68
9.3	Reference: "@20 General base objects".....	70
9.3.1	@20 > A10 Units.....	70
9.3.2	@20 > A15 Categories.....	71
9.3.3	@20 > A20 Locations.....	71
9.3.4	@20 > A25 Positions.....	72
9.3.5	@20 > A30 Functions.....	73
9.3.6	@20 > A40 Signals.....	73
9.3.7	@20 > A50 Project settings.....	73
9.3.8	@20 > A60 Documents.....	74
9.3.9	@20 > A70 Queries.....	74
9.3.10	@20 > A80 Scripts.....	75
9.3.11	@20 > A90 Standard import objects.....	76
9.3.12	@20 > B10 Decision tables.....	76

9.3.13	@20 > B20 Tasks.....	76
9.3.14	@20 > B30 User administration.....	78
9.3.15	@20 > B40 Workflows.....	78
9.3.16	@20 > B50 Expenses.....	78
9.3.17	@20 > B60 Configuration and mapping objects.....	79
9.3.18	@20 > B70 Installation.....	79
9.3.19	@20 > B80 Routing.....	79
9.3.20	@20 > B90 Risk-, safety evaluation.....	80
9.3.21	@20 > C10 Locking.....	80
9.3.22	@20 > C30 Symbol construction.....	80
9.3.23	@20 > C40 Maintenance.....	81
9.3.24	@20 > C50 General objects.....	81
9.3.25	@20 > C70 Automation objects.....	82
9.3.26	@20 > C80 SAP objects.....	82
9.3.27	@20 > C90 Objects for PipeSpec Designer.....	82
9.3.28	@20 > D10 Axis definition.....	82
9.3.29	@20 > D20 Base objects from external CAE data imports.....	83
9.3.30	@20 > D30 PDMS objects.....	83
9.3.31	@20 > D40 Reserved object names.....	83
9.3.32	@20 > Z10 Customer.....	85
10	"@30 Structures": Editing module-specific structures.....	87
10.1	Purpose of the "@30" node.....	87
10.2	Basic rules in the "@30" node.....	88
10.3	Specifying project structures as project setting.....	91
10.4	Overview of module-oriented first level.....	94
10.5	Overview of the structure of @30 below the modules.....	94
10.6	Designing project structures.....	96
10.7	Example for unit structure A30.....	99
10.8	Categories A35 below the modules.....	100
10.9	Device structure A50 below the modules.....	102
10.10	Document structure A90 below the modules.....	103
10.11	Information about labeling systems in the M00 branch.....	105
11	"@40 @Y Attribute catalog": Editing attributes and tabs.....	107
11.1	Overview of the administration of attributes and tabs.....	107
11.2	Note on attribute import.....	108
11.3	Uniqueness principle of attributes.....	108
11.4	Editing "@40 > A10 Attribute catalog".....	110
11.4.1	Overview of the nodes in "@40 > A10".....	110
11.4.2	Overview of the properties of catalog attributes.....	111
11.4.3	"Attribute mapping table" tab.....	112
11.4.4	Structuring variant: A05 Semantic structuring.....	113
11.4.5	Structuring variant: A10 Packaged structuring.....	114
11.4.6	Structuring variant: Y30 Special attributes.....	115
11.4.7	Structuring variant: Z10 Folder for customer sorting.....	116

11.4.8	Basic rules for the "Description" field for catalog attributes.....	116
11.4.9	Supplementary rules for the "Description" field for catalog attributes.....	117
11.4.10	Special attributes: Automation Interface (PCS 7).....	119
11.5	Editing "@40 > A20 Tabs".....	119
11.5.1	Overview of the nodes in "@40 > A20".....	119
11.5.2	Storage structure and names of the tabs	120
11.5.3	Change the sort sequence for the tabs.....	123
11.6	Placing catalog attributes on tabs (derived attributes).....	124
11.6.1	Placing a catalog attribute on a tab.....	124
11.6.2	Overview of the properties of the derived attribute.....	124
11.6.3	General rules for the layout of tabs.....	126
11.6.4	Attributes with the "List" display type (list specs).....	128
11.6.5	Attributes with the "Object query" display type (SUI query).....	129
11.6.6	Attributes with the "Button" display type.....	129
11.6.7	Attributes with the "Frame" or "Description" display type.....	130
11.6.8	Placing attributes invisibly.....	131
11.6.9	Placing a catalog attribute on a tab multiple times.....	131
11.7	Placing a tab on an object multiple times.....	132
11.8	Dynamically created attributes.....	133
11.9	Dynamically created tabs.....	134
12	"@50 Manufacturer devices": Editing the manufacturer catalogs.....	137
12.1	Purpose of @50.....	137
12.2	Classification in the iDB system.....	138
13	"@99 System settings": Editing default lists and settings.....	141
13.1	Reference: "@99 System".....	141
13.2	Overview of status management A40.....	143
13.3	"A40 Status": Object-based status management.....	144
13.3.1	Creating status base objects.....	144
13.3.2	Status value defaults in base objects.....	145
13.3.3	Controlling the status through attributes.....	145
13.3.4	Controlling the status using scripts.....	146
13.3.5	Controlling Navigator commands via script.....	147
13.4	"A40 Status": Query-based status management.....	148
13.4.1	Purpose of query-based status management.....	148
13.4.2	Creating and administering a "Status" query.....	148
13.4.3	Hierarchical query-based status management.....	149
13.4.4	Background knowledge of XML storage of the status.....	150
13.4.5	Query-based status management: Script examples.....	150
13.4.6	Attributes for query-based status management.....	152
13.5	"A50 Profiles/user settings": Administration of personal settings.....	153
13.6	"A90 Hardcoding": Using base data in script and code.....	155
13.6.1	Objective of the hardcoding object.....	155
13.6.2	Overview of the "A90 Hardcoding" object.....	156
13.6.3	Creating a hardcoding link.....	156
13.6.4	Accessing the hardcoding object in script and source code.....	157

13.6.5	Alternative operation modes for hardcoding.....	158
13.7	"B10 Tooling > @30 Tool": Administering cross-inheritances.....	159
13.7.1	Objective of "@30 Tool".....	159
13.7.2	User interface reference of "@30 Tool".....	159
13.7.3	Using "@30 Tool".....	162
14	Administration of standard tables.....	165
14.1	Overview of standard tables in the iDB.....	165
14.2	"@40 Attribute catalog table".....	165
14.3	Y10 System tables.....	167
14.3.1	Y10 Standard tables for line types.....	168
14.3.2	Y10 Standard tables for connector subtypes.....	168
14.4	Y30 Special standard tables.....	168
14.5	Y40 Mapping tables.....	169
14.6	Y60 NLS tables.....	169
15	Administering units.....	171
15.1	Administering unit groups for the iDB.....	171
15.2	Using units in iDB attributes.....	171
16	Administration of the labeling system and ALIAS labeling system.....	173
16.1	Constructing the labeling system with base object elements.....	173
16.2	Labeling systems in the M00 branch.....	175
16.3	Purpose of ALIAS labeling systems.....	178
16.4	Administration of the ALIAS labeling system.....	178
16.5	Using scripts to adjust ALIAS labeling.....	180
17	Classifying objects.....	183
17.1	Purpose of classification.....	183
17.2	Structure of classifications.....	183
17.3	Performing the classification.....	186
17.4	Classification of documents.....	187
18	Administration of documents in the iDB.....	191
18.1	Root nodes of the documents.....	191
18.2	Naming convention for report templates and document groups.....	193
18.3	Using or editing subreports.....	194
18.4	Object queries for reports.....	195
18.5	Base objects for documents and document groups.....	197
18.5.1	@20 > A60 Documents.....	197
18.5.2	Basic rules for documents and document groups in @30 > M00.....	198
18.5.3	@30 > M00 > A80 > A10 Document base objects acc. IEC 61355.....	199
18.5.4	@30 > M00 > A90 Document groups.....	200

18.5.5	Managing documents without IEC 61355 labels (@30 > xxx > A80).....	201
18.6	Classification of documents.....	201
18.7	Layout recommendations for report templates.....	202
18.8	Using external document files.....	203
18.9	Company logos.....	203
18.10	Displaying AutoCAD drawing in the report.....	204
18.11	Creating headers and footers.....	205
18.12	Interaction of tabs and report templates.....	206
18.13	Drawing type-specific symbol display.....	206
18.14	Important information on reports.....	206
19	Developing scripts for the iDB.....	207
19.1	Basic principles of scripts.....	207
19.2	Script library.....	208
19.2.1	Structure of the script library.....	208
19.2.2	Determining use of a script block.....	209
19.2.3	Methods in the script library.....	210
19.2.4	Naming of script objects.....	210
19.3	Structures/ conventions.....	211
19.3.1	Overview of the structures and conventions.....	211
19.3.2	Header block.....	211
19.3.3	Variable declaration.....	213
19.3.4	Variable initialization.....	214
19.3.5	Assigning transfer parameters.....	214
19.3.6	Return values of functions.....	214
19.3.7	Calling central scripts.....	215
19.3.8	Comments in the script.....	218
19.3.9	Formatting the script.....	219
19.3.10	Use of parentheses for If...Then...Else instructions.....	221
19.3.11	Use of variants for If...Then...Else instructions.....	221
19.3.12	Merging of character strings.....	222
19.4	Naming.....	222
19.4.1	Naming for methods.....	222
19.4.2	Naming for variable names.....	222
19.4.3	Variable types.....	223
19.4.4	Other properties.....	224
19.4.5	COMOS object types.....	224
19.5	Incorporating COM components.....	225
19.5.1	Overview of the technology in the COM components.....	225
19.5.2	Declaration components of the method "CreateObject".....	225
19.5.3	Memory storage management or cleaning with incorporated COM components.....	226
19.6	Avoiding runtime errors.....	226
19.7	Improve script performance.....	227
19.8	Localizing texts in scripts with NLS.....	228

19.8.1	Overview of the use of NLS texts.....	228
19.8.2	Administering NLS texts in standard tables.....	229
19.8.3	Functions for localizing texts.....	232
19.8.4	Call example of an NLS text.....	234
19.9	Prevent pop-up window.....	234
19.10	Logging script actions.....	235
19.11	Information about the Object Debugger.....	236
19.12	Using hard-coded base objects in scripts.....	236
19.13	iDB script editing.....	237
19.13.1	Opening script editing.....	237
19.13.2	Reference of the traversal search settings.....	237
19.13.3	Using the "Search and export of scripts" tab.....	239
19.13.4	Using the "Import scripts" tab.....	241
19.13.5	Using the "Search for script usages" tab.....	242
20	Editing project properties.....	245
20.1	Global project options.....	245
20.2	Base references: References to database objects.....	245
20.3	Project properties reference.....	246
21	Customer-specific additions.....	247
21.1	General rules for customer-specific supplements.....	247
21.2	Inserting customer-specific base objects in the iDB.....	248
21.3	Creating customer-specific attributes and tabs.....	251
21.4	Creating customer-specific documents.....	253
21.5	Customer-specific project properties.....	256
21.6	Importing attributes and tabs in bulk.....	257
21.6.1	Aim of the iDB import.....	257
21.6.2	Overview of the import of attributes and tabs.....	258
21.6.3	Overview of the Excel workbook.....	259
21.6.4	Excel workbook: "Attributes" Excel sheet.....	260
21.6.5	Excel workbook: "Tabs" Excel sheet.....	263
21.6.6	Structuring variants of the iDB attribute import.....	266
21.6.7	Preparing the "Attributes" Excel table for attribute import.....	269
21.6.8	Importing attributes by means of "(@40) @Y Attributes and tabs".....	273
21.6.9	Controlling repeated synchronization.....	275
21.6.10	Structuring variants and types of the iDB tab import.....	277
21.6.11	Preparing the "Tabs" Excel sheet for the tab import.....	279
21.6.12	Quick import of tabs.....	282
21.6.13	Importing tabs by means of "(@40) @Y Attributes and tabs".....	284
21.7	Attributes and tabs are generating individually automatically.....	285
21.7.1	Aim of individual editing of the @40 catalog.....	285
21.7.2	Configure "(@40) @Y Attributes and tabs" for individual creation.....	286
21.7.3	Create individual attribute with "(@40) @Y Attributes and tabs".....	288
21.7.4	Create individual tab with "(@40) @Y Attributes and tabs".....	289
21.7.5	Refreshing the naming area.....	291

21.8	Obsolete base data.....	291
22	iDB-specific rules for using external files.....	293
22.1	Using or editing external files.....	293
22.2	Physical storage of the report templates.....	294
23	Using case variants as design cases or editing them.....	297
24	Administration of templates.....	299
25	Database update.....	301

Objective and requirements

Objective

This document contains guidelines for editing the base data and templates in "COMOS iDB (industrial data base)" COMOS databases. In this documentation, the abbreviation "iDB" is used for "COMOS iDB (industrial data base)".

Required expertise

Knowledge of the following manuals is required:

- COMOS Installation
- COMOS Operation
- COMOS Administration

COMOS version

The following COMOS version is required:

- COMOS 10.2 or higher

Older COMOS versions cannot open the iDB since the current classification key and other new iDB features are only supported by the current COMOS version.


Using working layers

Do not work in the released area.

Use working layers when you adapt the iDB. In a working layer, you can test the changes without obstructing COMOS users.

This also applies to correction of translation errors.

Consequences of deviations

 CAUTION
Data loss and loss of performance possible
Any deviation from the guidelines described in this documentation can have the following consequences:
<ul style="list-style-type: none">• Loss of data due to inconsistencies• Increased expenditure for administering the database• Increased expenditure for using the database• Loss of update capability of the database through database update

If you are not sure which customizations you should make to the database, speak to COMOS support.

Searching for objects and data in the iDB

2.1 Overview: Search techniques for the iDB

Overview: Identifying objects

The following options are available for identifying objects in COMOS:

- Name
See chapter Standardized designations for folder objects (Page 21).
See chapter Standardized designations for base objects (Page 27).
- Label
- Description
See chapter Generally-applicable writing rules in standardized descriptions (Page 33).
See chapter Localization rules for iDB texts (Page 35).
- Classification
 - For engineering objects
 - For documentsSee chapter Classifying objects (Page 183).

Alternative 1: "Attribute search and creation" tab

You can enter one or multiple strings in the "Attribute search and creation" tab and the "Synonyms" tab to search for the description of attributes. The results list is optionally displayed in the tab or exported to an Excel list.

See chapter Finding attributes with "(@40) @Y Attributes and tabs" (Page 18).

Alternative 2: Query "A10 Attribute descriptions DE/EN"

The "Attribute descriptions DE/EN" query is the fastest way to find an attribute based on description in the attribute catalog. The description of the attribute contains a text for the attribute purpose.

See chapter Using "Attribute descriptions DE/EN" query (Page 17).

Alternative 3: Query "A10 Tab descriptions DE/EN"

The "Tab descriptions DE/EN" query is the fastest way to find a tab based on description in the tab catalog. The description of the tab contains a text for the tab purpose.

See chapter Using the "Tab descriptions DE/EN" query (Page 19).

Alternative 4: Creating your own query

You can find objects in the iDB using queries. Queries provide various search techniques:

- **Recursive search**
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Queries for engineering objects".
The recursive search occurs starting from a start object via the collections of this object. This search procedure is suitable for searching in deeper structure levels with a manageable number of objects. Example: Search for equipment of a pump station.
- **Back pointer search**
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Query for engineering objects".
The back pointer search starts searching from the base object and finds all instances of this base object. The search via base object pointer in the properties of the base object is also supported.
Example: Search for all occurrences of an object in a project.
- **Reference attributes**
The search using reference attributes (also called link specifications) is a search in which a reference to a "common feature" (or a common object) is set in the project. This link must be provided in the data model. This search is fast and targeted. However, it requires that the project and engineering objects be created accordingly.
Example: Process streams which are transferred from the PFD to one or more pipes or pipe segments.
- **Database search (Search Manager)**
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Database search".
The Search Manager is used for searching on the database layer directly. A query is converted directly into an SQL query, which is able to search even larger object amounts very quickly. Note: For an object to be found, the information for which the search is being performed must be checked in at the object.
There are two variants:
 - **Direct database search**
A search for any properties in the complete database. Examples:
Properties which are always checked in: name and label.
Properties which can only be searched for if they have been checked in at the object: description and value.
 - **Specialized search for classification keys.**
Classification keys are always checked in at the object. This means that it is possible to search for objects across engineering projects.
- **Scan Manager**
The Scan Manager is outdated and should no longer be used.

Alternative 5: Full-text search for documents

The full-text search allows you to search within all documents on the file level. Both internal COMOS documents such as reports and revisions and external COMOS documents such as Word, Excel, or pdf files can be searched. In addition to this, metadata of objects can be searched.

A requirement for full-text search is the indexing of the project. The search is very fast and can search 50,000 documents, for instance, in an extremely short amount time. A disadvantage is the large amount of data which arises through indexing.

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Overview of the full text search".

2.2 Using "Attribute descriptions DE/EN" query

Requirement

- You are familiar with the overview of the search for objects and data. See chapter Overview: Search techniques for the iDB (Page 15).

Procedure

The "Attribute descriptions DE/EN" query is the fastest way to find an attribute based on description in the attribute catalog. The description of the attribute contains a text for the attribute purpose.

1. Open the "Base data" tab in the Navigator.
2. Find the following object:
"@40 > A10 @Y Attribute catalog"
3. Open the tree structure below the object.
Query "A10 Attribute descriptions DE/EN"

Note

Visibility when using the detail area

The query "A10 Attribute descriptions DE/EN" is created as element. When the detail area is open, the query is only visible there. To do this, the owner object of the query must be selected in the main navigator.

4. Select the "Open (classic)" command in the context menu of the "A10 Attribute descriptions DE/EN" query.
5. For example, set the following start object:
"@40 > A10 > Y00 > A05 @Y Semantic attribute collection"
6. Click the "Search" button in the command bar of the query.
Wait for the result.
7. Select the "AutoSort" command in the column header of the query.
Sort either the "Description DE" or "Description EN" column.
8. Scroll in the list to the desired description.
9. Alternatively: Select the "Autofilter" command in the column header of the query.
Filter by a description.
10. Select the "Navigate > Attribute" command in the context menu of the "Description" column.

2.3 Finding attributes with "@40 @Y Attributes and tabs"

Requirement

- You are familiar with the overview of the search for objects and data. See chapter Overview: Search techniques for the iDB (Page 15).
- You are familiar with the attribute catalog "@40". See chapter Overview of the administration of attributes and tabs (Page 107).

Searching for objects using the "Attribute search and creation" tab

1. Open the base project "@40 @Y Attributes and tabs".
2. Open the "Attributes > Attribute search and creation" tab. There you will find the "Search string" area:

The screenshot shows the 'Attribute search and creation' tab in a software interface. The 'Search string' section is active and contains the following elements: a 'Start object' input field with the text '@40|A10|Y00', a 'String' input field, an 'Or' operator button, an 'Info' button, a 'Language' dropdown menu currently set to 'ALL', and two checkboxes for 'Options': 'With usage' and 'Case-sensitive'. A 'Search string' button is positioned at the bottom right of this section.

3. Check the "Start object" field. If necessary, set a start object from the base data.
4. Select the "String" field. Enter one or two strings. Click "Info" for information on the use of placeholders. If a search is made for two strings, the strings are linked with "OR". It is not possible to select another operator.
5. In the "Language" field select the language in which the search for the string is to be performed.
6. "With usage" option. This option takes account of the use of attributes in the node "@40 @Y > A20 Tabs catalog".
7. "Case-sensitive" option. Activates case sensitivity.
8. Click the "Search string" button.

"Create attributes" group

See chapter Aim of individual editing of the @40 catalog (Page 285).

2.4 Using the "Tab descriptions DE/EN" query

Requirement

- You are familiar with the overview of the search for objects and data. See chapter Overview: Search techniques for the iDB (Page 15).

Procedure

The "Tab descriptions DE/EN" query is the fastest way to find a tab based on description in the tab catalog. The description of the tab contains a text for the purpose or module of the tab.

1. Open the "Base data" tab in the Navigator.
2. Find the following object:
"@40 > A20 @Y Tab catalog"
3. Open the tree structure below the object.
Query "A10 Tab descriptions DE/EN"

Note

Visibility when using the detail area

The query "A10 Tab descriptions DE/EN" is created as element. When the detail area is open, the query is only visible there. To do this, the owner object of the query must be selected in the main navigator.

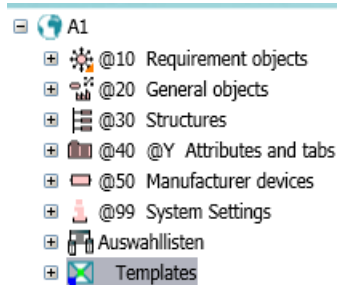
4. Select the "Open (classic)" command in the context menu of the "A10 Tab descriptions DE/EN" query.
5. For example, set the following start object:
"@40 > A20 > Y00 > A10 @Y Tabs 00001 - 00500"
6. Click the "Search" button in the command bar of the query.
Wait for the result.
7. Select the "Sorting / Filter" command in the column header of the query.
Sort and group the "Description DE" column or the "Description EN" column.
8. Scroll in the list to the desired description.
9. Alternatively: Select the "Autofilter" command in the column header of the query.
Filter by a description.
10. Select the "Navigate > Attribute" command in the context menu of the "Description" column.
Note on the entry in the "Navigate" menu: A tab is technically created as an attribute.

Standardized designations for folder objects

3.1 Names and meaning of the main levels in the iDB

Relationship between the system types and the main nodes

The COMOS iDB is subdivided into main nodes:



The main nodes start with an @.

- @10 COMOS base objects
 - Base objects
 - Connectors

General base objects which are not dependent on particular standards. This is where you can edit settings of the base objects required for the COMOS basic functionalities. @10 is used in @30.

On the lower levels, the basic objects are differentiated according to physical requirements and logical requirements.

See chapter Structure of the "@10" node (Page 61).

See chapter Use/expansion of the "@10" node (Page 63).
- @20 General objects, not dependent on engineering
 - Base objects
 - Connectors

General base objects which are not dependent on particular standards. That also includes base objects which are not part of a unit but which generally support the work in COMOS. Example: Decision tables, engineering tasks, workflow objects.

See chapter "@20 General base objects": Editing basic base objects (Page 67).

3.1 Names and meaning of the main levels in the iDB

- @30 Structures
 - Base objects
 - Connectors
 - Base objects for documents
 - Categories
 - Labeling systems
 - General base objects for structure objects (units, locations, folders)

The standard-specific or industry-specific base objects are not new creations, but are an expansion of @10.

See chapter "@30 Structures": Editing module-specific structures (Page 87).

- @40 @Y Attributes and tabs
 - Attributes
 - Tabs

Root nodes for tabs and attributes

The name affix "@Y" must be retained for compatibility reasons.

See chapter "@40 @Y Attribute catalog": Editing attributes and tabs (Page 107).

- @50 Manufacturer devices
 - Base objects
 - Connectors

Manufacturer device catalog that is an expansion of the objects in @30.

See chapter "@50 Manufacturer devices": Editing the manufacturer catalogs (Page 137).

- @99 System settings
 - Classifications
 - Database-dependent user interfaces
 - Document type mapping
 - Revision base objects
 - Status of objects
 - Profile objects
 - Working layers
 - Layers of documents

System-related objects

See chapter Reference: "@99 System" (Page 141).

- Standard tables
 - See chapter Administration of standard tables (Page 165).

- Templates
 - Templates are not base data, but instead are engineering objects which are prepared so that they are ready for use.
 - You can find additional information on this topic in the "COMOS Administration" manual, keyword "Administering templates".

Connectors do not have their own main node. Instead, the connectors are prepared in nodes @10, @20, and @30. See chapter Connector types in the iDB (Page 53).

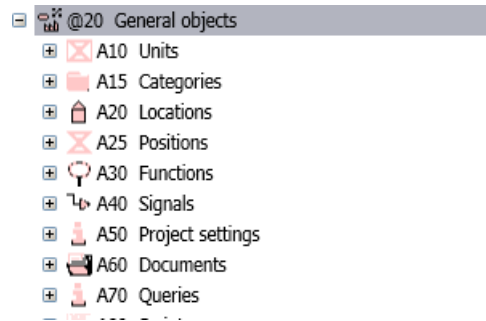
Documents do not have their own main node. Instead, base objects of documents are sorted in node @30. The template objects of the documents are on the "Documents" tab in the base project. See chapter Administration of documents in the iDB (Page 191).

3.2 Names of additional levels

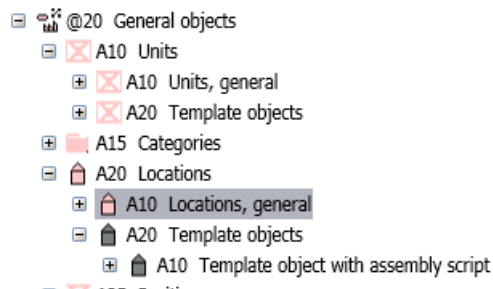
Counting convention for names

The following structure exists below the main nodes:

- Standardized counting with a capital letter and a two-digit number
Example: A10
- Counting in increments of five or ten (numbers in between as reserve)



- Lower levels, alternative 1: The first letter is incremented. Example: "A10 > B10".
- Lower levels, Alternative 2: The first letter is repeated:



- The identifier remains empty in principle.
If additional identification is required for a base object due to software reasons, you can use the identifier in exceptional cases:
- Customer objects
The prefix "Z" designates customer-specific nodes and objects.
The suffix "_Z" designates customer-specific nodes and objects.
See chapter Inserting customer-specific base objects in the iDB (Page 46).

Additional information

See also chapter System characters and special characters (Page 28).

See also chapter Generally-applicable writing rules in standardized descriptions (Page 33).

3.3 Special sub-branches with Y prefix

Some areas below the main nodes are labeled with a Y prefix:

- Y00
The general objects (catalog objects) are located in this area.
- Y10
Objects with system access are located in this area.
- Y20
The core objects are located in this area.
- Y30
This area includes objects which require special attributes and tabs to support an interface and usually have a deviation from the standardized naming convention. The attributes and tabs are taken from the following nodes:
 - "@40 > A10 > Y30 @Y Attribute catalog > Y30 @Y Special attributes collection"
and
 - "@40 > A20 @Y Tab catalog > Y30 @Y Special tab collection"
- Y40
Objects for mapping objects to one another are located in this area.

You will find an example for the names based on the above list below "@20 > A70 Queries".

3.4 The @Local branch for local base objects

Global base objects versus local base objects

- Global base objects
base objects created in the base project, which can only be modified there.
You can find additional information on this topic in the "Operation" manual, keyword "Fundamental knowledge about objects".
- Local base objects
Base objects that are created in the engineering project on the "Base objects" tab. The local base object is distinguished from the base objects of the base project by a blue globe.

Avoiding mixed structures

1. A local base object should not be created below a global base object.
Failure to heed this recommendation causes the situation in which global base objects are regarded as owners of local base objects. In this case, the following applies:
 - The "Copy structure" function generates new local base objects from the global base objects within the owner structure.
 - The "Project: Export" function followed by the "Project: Import" function creates new local base objects from the global base objects within the owner structure.

In both cases, the database is expanded and contains many of the same base objects multiple times.

2. Mixed structure are technically prevented for documents.
You can only create documents under a base object if the base object is located within the project that was just opened.

Recommendation:

- Create one or more branches for local base objects below the project root in the engineering project on the "Base data" tab. Create local base objects only in these branches.
- If you need parts of the global base objects in the local base objects, set a reference to a global base object in the "Reference" field in the properties of the local base object.
- Use @Local if possible.

@Local branch for preferential use of local base objects

To automatically search for local base objects and use them instead of the base objects from the base project, proceed as follows:

1. Create a "@Local" node below the local base objects in the engineering project.
2. Create a new base object below "@Local".
3. In this new base object, set a base object reference to the original from the base project.

If you create an engineering object on the basis of an "original" from the base project, COMOS checks the following: Is there a "BackPointerCDevice" from the current engineering project for this "original" located under "@Local"? If yes, this local base object is used in the engineering object as the base object pointer.

This mechanism also works for elements. So you can use the above method to create a project-specific context menu.

Similar functions

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Objective of instantiation".

Standardized designations for base objects

4.1 Overview of the naming convention in the iDB

The following provides an overview of how the names of the objects in the iDB are formed. Using these masks you can, for example, search for items specifically by name.

- Attribute:
Y00A12345
See chapter Structuring variant: A05 Semantic structuring (Page 113).
- Tab:
Y00T12345
See chapter Storage structure and names of the tabs (Page 120).
- Standard table:
Y00MxxN12345
See chapter "@40 Attribute catalog table" (Page 165).
- Reserved name:
Y00R12345
See chapter Base objects with fixed names (Page 31).
- Border on tab:
FR001
See chapters Design guidelines for tabs (Page 40) and Overview of the properties of catalog attributes (Page 111).
- Label on tab:
LA001
See chapters Design guidelines for tabs (Page 40) and Overview of the properties of catalog attributes (Page 111).
- Script in the script library:
MxxS12345
See chapter Naming of script objects (Page 210).
- Customer objects
The prefix "Z" designates customer-specific nodes and objects.
The suffix "_Z" designates customer-specific nodes and objects.
Example: Z00A12345

Explanation of the wildcards used in the name masks:

- Mxx corresponds to the module key with a two-digit count index (00-99)
See chapter Special cases when using letters (Page 30).
- 12345 corresponds to a five-digit count index (00001-99999)
- 001 corresponds to a triple-digit count index (001-999)

4.2 Uniqueness of names

"Unique name across folders" option

Objects in COMOS are technically managed using an invisible "UID" (unique identifier). In the user interface, the name is the identification feature used first.

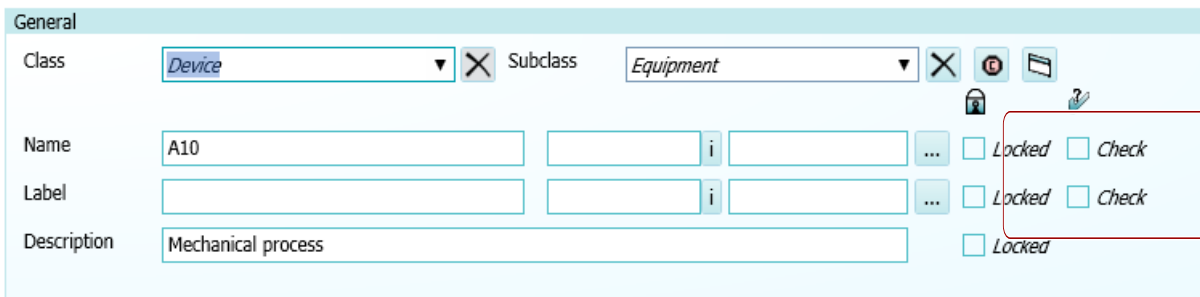
A name can be unique in two contexts:

- Unique in the entire project
If a name is not unique across the project, the object is identified using the name and position. The name is then referred to as "SystemFullName".
- Unique within a hierarchical level.

You can specify this context in the properties of the project, "Options" tab, "Unique name across folders" option.

Checking of name masks

In the "@10" and "@20" branches, the checking of naming conventions is disabled for all objects in the COMOS iDB in the "System" tab.



4.3 System characters and special characters

The following characters are used internally in COMOS:

- Pipe sign "|"
Interpreted in the name as part of the Systemfullname.
Interpreted in the description and other translatable fields as a delimiter for the translations of the entry.
- Period "."
Interpreted in the name as structure information of potentials.
- @
 - cDB: The name is only visible to administrators.
 - iDB: No technical significance. Only used for main nodes in the delivery state.

Other conventions for entering strings also exist:










- Names of attributes
 - cDB: May not start with a special character.
 - iDB: For name template Y00A12345, see section Overview of the naming convention in the iDB (Page 27).
- The relevant manual sections contain descriptions of other conventions which are only valid in specific interfaces.

The system characters and special characters must be considered in scripts and during imports. Errors may occur if an invalid system character is entered in a name or a description by a script or import.

4.4 Names of base object elements

The elements of a base object are named according to the "Counting convention for names" principle for base objects. See section Names of additional levels (Page 23).

Example:

- [-]  A20 Pipe section -> @10|A20|A10|A10|A40|A10|A20 Pipe section
 - [+]  A20 Pipe valves (Fittings) -> @30|M22|A50|A10|A40|A20 Pipe valves (Fittings)
 - [+]  A30 Pipe parts (Fittings) -> @30|M22|A50|A10|A40|A30 Pipe parts (Fittings)
 - [-]  A40 Nozzle for pipe -> @30|M22|A50|A10|A70|A20|A10|A20 Nozzle for pipe
 - [+]  A10 Nozzle, straight -> @10|A20|A10|A10|A70|A20|A10|A20|A10 Nozzle, straight
 - [+]  A20 Nozzle, screwed -> @10|A20|A10|A10|A70|A20|A10|A20|A20 Nozzle, screwed
 - [+]  A30 Nozzle, welded -> @10|A20|A10|A10|A70|A20|A10|A20|A30 Nozzle, welded
 - [+]  A40 Nozzle, -> @10|A20|A10|A10|A70|A20|A10|A20|A40 Nozzle,
 - [+]  A50 Heatinn. coolinn devices and Insulation for nine -> @30|M22|A50|A10|A70|A60 He

Special features of text masks for elements: See section Administering properties of elements in the base object (Page 40).

4.5 Names in the labeling system

Constructing labeling systems with base objects

If a different type of designation is specified by labeling systems or standards in catalogs, etc., then deviations from the standard designation, starting from the node for the corresponding substructure, are permitted. See chapter Constructing the labeling system with base object elements (Page 173).

Labeling systems can also be constructed using text masks for the names. See chapter Labeling systems in the M00 branch (Page 175).

4.6 Special cases when using letters

There are special letters in the COMOS iDB that are not used for counting and are skipped for this reason:

- "M"

The letter M is reserved for COMOS modules.

The list of current modules can be found in the iDB below "@30 Structures" The following list shows a snapshot as an example:

Area	Number range	Module	Module number
Platform	M0x	COMOS general	M00
		Basic	M01
		Enterprise Server	M02
		SAP	M03
		View/Web view	M04
		Interfaces	M05
		Teamcenter Interface	M06
Process	M2x	Process general	M20
		FEED	M21
		P&ID	M22
		PipeSpec Designer	M23
		PipeSpec Manager	M24
		Isometrics	M25
		PlantModeler	M26
		PDMS Integration	M27
		MTO (Material management)	M28
		Automation	M4x
EI&C	M41		
Logical	M42		
Fluidics	M43		
PCS7 AdvES – omitted since V10.1.3.0.0	M44		
Automation Designer – omitted since V10.1.3.0.0	M45		
Operations	M6x	Operations general	M60
		MRO	M61
		Shutdown	M62
		Portable & Direct	M63
		Inspection	M64
		PQM	M65
		PQM4RI	M66

Customer	Zxx	Customer	Z00
		etc.	Z..

- "W"
The letter "W" identifies nodes and objects belonging to Siemens in-house topics.
- "X" in the "Documents" topic area
The letter "X" designates object queries on report templates.
Example: AB_A10_X10
- "X" in the "Database update / Database extension" topic area
The letter "X" indicates nodes and objects that belong to the content layer.
- "Y"
The letter "Y" is reserved for designating attributes, tabs, standard tables, and scripts.
- "C"
The letter "Z" designates customer-specific nodes and objects.

4.7 Base objects with fixed names

There are base objects in the database that must have a predetermined name and must be located at a predetermined location within the base object structure in order to be found and used. So these base objects have to be available at a particular location, under a particular name.

Names are reserved in the iDB for this reason and stored in the following node:

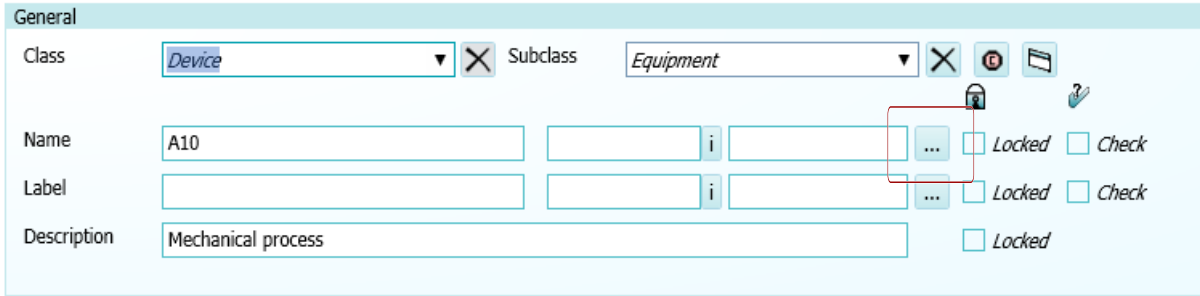
- @20 > D40
See chapter @20 > D40 Reserved object names (Page 83)
 - The "Y00" node stores objects with reserved names that belong to the delivery state of the iDB.
 - The "Z10" node stores customer-specific objects with reserved names.
- In the other main nodes, base objects are created which have a base object pointer set to the base object with a reserved name created in the "@20 > D40". The reserved names are transferred through the base object pointer.
- The name template for reserved names is Y00R12345.

Connectors sometimes have fixed names too. See chapter Reference: Names of the connector types (Page 54).

4.8 Entering norm-specific text masks

Procedure

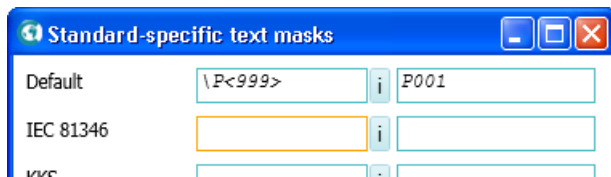
After the text fields for text masks and standard texts, there is a button labeled "...":



In the next dialog, you can enter the following text masks and standard values (or a @MASK entry):

- Default value
This value is used if there is no entry available for the standard. The entry should be as neutral as possible so that it can be used universally.
- Standard-specific entries (list of standards is predefined)
- Company-specific entry
- User-specific entries

The usual rules for entering text masks apply. You can find more information on these by clicking on the "i" button next to the text fields:



Use of standard-specific text masks

See chapter Basic rules in the "@30" node (Page 88).

Generally-applicable writing rules in standardized descriptions

5

Capitalization

For all texts, use capitalization in accordance with the spelling rules for the relevant language. This also applies to the "Description" field for objects and attributes.

In English, this means that apart from recognized exceptions, only the first word in a sentence has a capital letter, e.g. "Inner diameter".

Use of special characters

- Use of hyphens
 - Terminology (fixed terms, e.g. *EN 1092-1*): Hyphen without blank spaces
 - Compounds (e.g. 3-pole): Hyphen without blank spaces
For compounds, check whether a spelling without a hyphen is considered correct.
 - Separating nouns and attributes (e.g. Ladder – neutral): A blank space before and after the hyphen

Label

Objects have no entry in the "Label" field in factory state. The only exception is when a label is required for technical reasons.

See also

Localization rules for iDB texts (Page 35)

Localization rules for iDB texts

Requirement

- You are familiar with the notation for iDB texts.
See chapter Generally-applicable writing rules in standardized descriptions (Page 33).

Including texts which can be localized

Always translate all texts which can be localized in order to avoid inconsistencies.

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Application area of language management".

Default: German and English

Primary language is English. If no entry is found in a language, the English text is displayed.

Always translate into the following languages:

- German
- English
 - Preferred variant: US English

Do not use any foreign-language entries in the current input language. Example: Do not enter German text in an English field or vice versa.

Neutral names

The names of engineering objects, attributes, and documents are untranslatable. Therefore:

- Never use language-dependent information in object names.

Neutral names usually consist of sequential alphanumeric characters. Translatable name elements must be entered in the description or tooltips.

Do not use country codes

Do not use the country code for labeling translations. The country code is not a unique label for a language variant.

Supported localization tools

- Alternative 1: Translate new
"Extra > Translation > Single translation", "Extra > Translation > Bulk translation"
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Using single translation: Translation of the description".
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Manual translation with bulk translation".
- Alternative 2: Check existing translations:
"Extra > Translation > Translation search"
- Alternative 3: Automatically apply existing translations:
"Extra > Translation > Bulk translation: "Database translation > Automatic translation" option"

Note

Important:

Use the translation tools even when you are only editing one language. If you enter translations directly into text fields, COMOS inserts the prefix "?0?" in front of texts in other languages as an inconsistency marker.

Error in delivery state

Do not correct the delivered database of the iDB. Local corrections can result in inconsistent translations.

Proceed as follows instead:

- Create a working layer in the base project.
- Correct the data in the working layer.
- Export the working layer.
- Send the exported working layer to the following address:
SolidBaseRequest.i-ia@siemens.com
- Use the next version of the iDB.

See also

Localizing texts in scripts with NLS (Page 228)

COMOS fundamentals of base objects

7.1 "Object locking" and "System object" state

Overview of the functions for protecting system-relevant objects

The iDB contains system-related objects which must not be changed. The following functions are available to protect these objects:

- "Lock object"
This function right can be set and also unlocked again by administrators. The "Lock object" function right has the states "Unlock" and "Lock".
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Function right "Lock object"".
- "Object locking"
The "Object locking" function is available in two versions:
 - Object locking in the delivery state (locking on the system side; SysLock)
This function is only available for Siemens employees.
 - Object locking after delivery of the iDB (locking by customer; AdminLock)
This function can be used by administrators.
- "System object"
For reasons of clarity, the "system object" status is also available. This status shows that the object is a system-related object. The status has no other implications. In practice, the "System object" status is often combined with the "Object locking" function.

Effect of the "Object locking" function

Objects which have been locked on the system side have the following properties:

- Locked against being changed, deleted, moved
- The following can be changed:
 - Properties for user inputs ("Value", "XValue", "Unit", "Link objects")
 - Description
- The creation of elements can be independently permitted or prohibited
- The creation of objects can be independently permitted or prohibited

The following object types can be locked on the system side:

- Base objects
- Attributes
- Attributes tab
- Connectors
- Standard table

7.1 "Object locking" and "System object" state

- Standard table value
- Unit group
- Unit
- Document
- Engineering object

Inheriting of the "Object locking" function

The "Object locking" property is inherited in the base data. In contrast to other properties it is not possible to see if the setting is inherited in the case of the "Object locking" property.






If the "lock-lock" property has been edited on a base object and thus the inheritance is interrupted, it is not possible to return to the inheritance.

Checking the state of the "Object locking" function (locking on system side)

You can check the status of an object as follows:

- Properties of the object, menu bar of the tab
 - An icon for the status regarding the object locking
 - An icon for the status regarding the "system object" property.
- Window for the status of the object rights (Ctrl+A)
 - "Status of the object locking" field

The object locking status can be as follows:

	Unlocked
	AdminLock (without creation of substructures)
	AdminLock (with creation of substructures)
	SysLock (without creation of substructures)
	SysLock (with creation of substructures)

Objects with object locking are displayed in gray in the navigator.



Editing the "Object locking" state

When the object is unlocked, the object is switched over in the state "AdminLock (without creation of substructures)" or "SysLock (without creation of substructures)" with a click on the button.

When the object is in the state "AdminLock (without creation of substructures)" or "SysLock (without creation of substructures)", the following options are available:

- Clicking the icon again
The object is switched back to the "Unlocked" state.
- Selection via the menu
An arrow is displayed at the icon. It is then possible to switch between the states "without creation of substructures" and "with creation of substructures".

"System object" status

	Object is not a system object.
	Object is a system object.

7.2 Creation option for base objects

Properties of a base object, "System" tab, "Creation option":

- Base objects for structuring the base data (folders)
Creation option: "Structure"
- Base objects that form the basis for engineering objects
Creation option: "Normal"

7.3 Adding tabs to base objects

Procedure

1. Optional: Edit the "@40" catalog
 - Add missing tabs in the "@40" tab catalog.
 - Edit tabs in the "@40" tab catalog.
2. Open the properties of a base object.
3. Select the "Attributes" tab.
4. Switch to design mode.
5. Move a tab from the "@40" tab catalog to the design area using drag&drop.
No changes to the tab are allowed at the base object itself.

See also

Design guidelines for tabs (Page 40)

7.4 Design guidelines for tabs

Requirement

- You know how tabs are added.
See chapter Adding tabs to base objects (Page 39).

Specifications for frames on tabs

Use the following default settings:

- Title visible: Switched on
- Frame type: Normal
- Font
 - MS Sans Serif
 - Normal
 - 8
 - Black

7.5 Creating elements at base objects

7.5.1 Administering properties of elements in the base object

Requirement

- An element has been created.
See chapter Constructing the labeling system with base object elements (Page 173).
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Creating an element for the base object".

Creation mode

You can specify the creation mode at each base object.

The creation mode controls how the base objects and elements are created below the engineering object in the engineering view:

- | | |
|----------|---|
| Free: | All prepared objects are created in the engineering view. |
| Element: | Only the objects which are defined as elements for this object in the "Elements" tab on the base data side are created in the engineering view. |

"Inheritance mode" setting

Value	Description
"Active"	The element is inherited by the derived objects and by the base objects under the element owner.
"Inactive"	Inheritance is switched off completely. You can use this inheritance mode to prepare multiple elements at one level of the structure tree and deactivate the elements which are no longer needed in the lower levels.
"Inactive for base objects"	Inheritance within the base data is deactivated.

Inheritance mode is also available for the following system types:

- "Specification"
- "Connector"

"Virtual" setting (for elements only)

Only elements have this control group. It governs how often and when the element is created in the engineering project. Elements which have been prepared in the base data but have not yet been created in the engineering data are designated as "virtual".

The "Virtual" setting has the following options:

- "[] times" The element can be inserted into the engineering project as often as specified and can be deleted. Once the maximum number is reached, the object is no longer offered in the context menu. The name template is taken into consideration. The inheritance mode is not evaluated.
- "N times" The element can be inserted into the engineering project any number of times and can be deleted. The name template is taken into consideration. The inheritance mode is not evaluated.
- "Off" The element is created automatically when the engineering object is created and cannot be deleted. The name template is not taken into consideration. In addition to this option, the inheritance mode is evaluated.
- "Preselection" The element is created automatically when the engineering object is created but can be subsequently deleted. After deletion it can be created again manually. The name template is taken into consideration.
 - Database of the type iDB: The name template is taken into consideration.
 - Database of the type cDB: The name template is not taken into consideration.
 In addition to this option, the inheritance mode is evaluated.

Note

For virtual "Off":

"Dereference" is automatically set to "No" internally (regardless of the setting in the "Reference" group)

Additional information

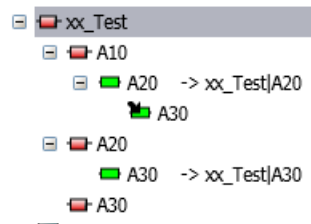
You can find additional information on this topic in the "COMOS Administration" manual, keyword ""Elements" tab for base objects".

7.5.2 Constructing context menus via elements

Procedure

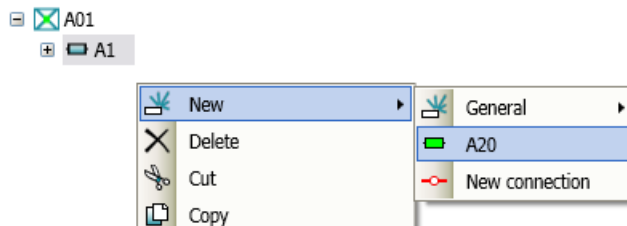
A hierarchical structure is created from elements. Example:

- Base object A30 is inserted on the "Elements" tab of base object A20
- Base object A20 is inserted on the "Elements" tab of base object A10



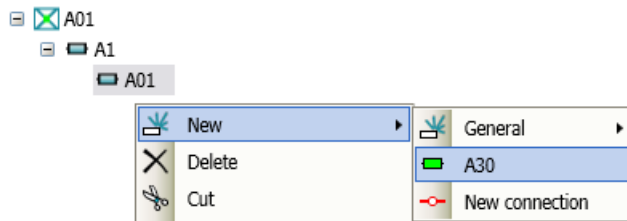
Result in the context menu

If base object A10 is created as an engineering object, then (in this example) engineering object A1 is produced. Engineering object A1 has the following context menu:



If the element A20 is selected in the context menu of object A1, then engineering object "A01" is produced. The name change from "A20" to "A01" is compliant with the COMOS naming rules.

Object A01 has the following context menu:



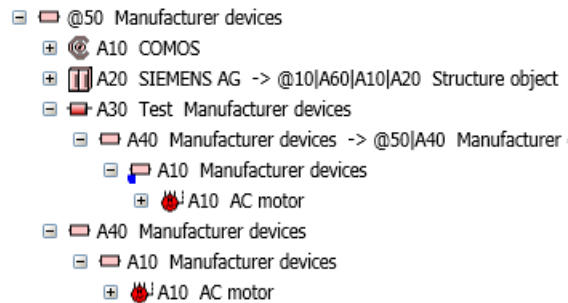
7.5.3 Constructing context menus via subelements

Procedure

A hierarchical structure is created from base objects. The complete structure is used as an element via a root object.

Example:

- Base object A40 is created with the "Structure" creation option.
- Base object A10 is created below this, with the "Structure" creation option.
- Base object A10 is created below this, with the "Normal" creation option.
- Base object A40 is inserted on the "Elements" tab of base object A30.

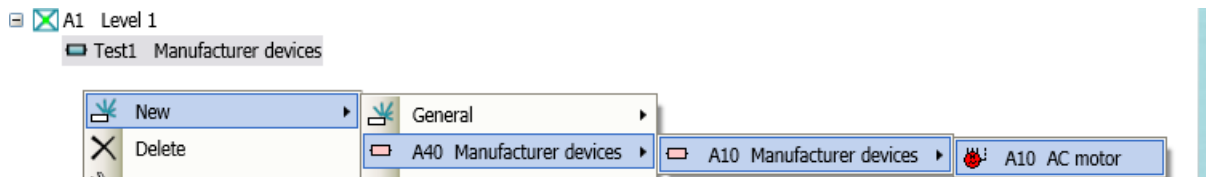


Note

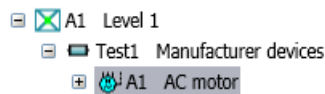
The creation options for base objects A40 and A10 must be set as a "Structure" in order that they are displayed via the context menu.

Result in the context menu

If base object A30 is created as an engineering object (with the name "Test1" in the following example), then this engineering object has the following context menu:



This results in the following structure in the engineering view:



Note that the intermediate layers have not been created as engineering objects.

7.6 Forming a labeling system with elements

Aim of a labeling system

Labeling systems are a standard method for labeling all objects in a unit. In COMOS, labeling systems are constructed via the automatic assignment of names or labels for newly created engineering objects. In this case, the base data is prepared so that, when the engineering objects are created, the names are assigned according to the labeling system.

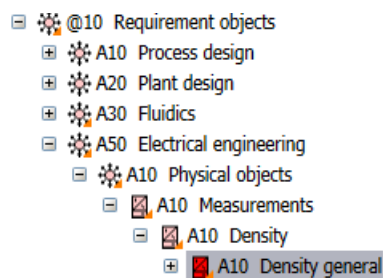
Standard-specific labeling systems are constructed with the help of base object structures.

See Chapter Administration of the labeling system and ALIAS labeling system (Page 173).

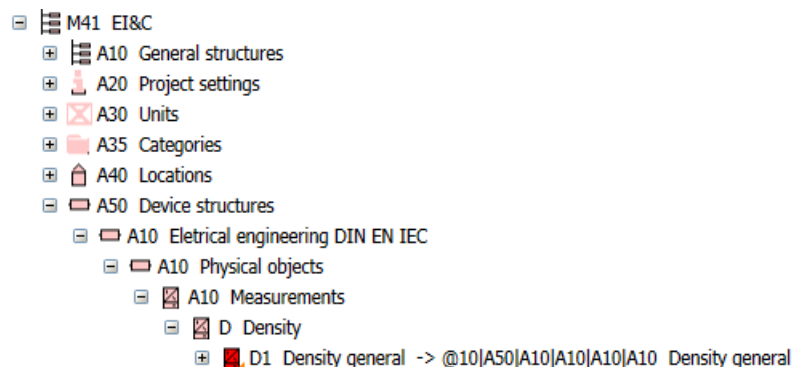
7.7 Prohibited change for the target of cross-inheritances

In the following example, base object (target) has been produced from base object (source) by means of a cross-inheritance (reference in the "Base object" field).

Source:



Target:



The following rules apply:

- Do not change the base object (target).
- Create a child object below base object (target).
- Edit the base object (child object).

7.8 Inserting customer-specific base objects in the iDB

Independent structures in customer area Z

You can create and configure your own base objects below the customer nodes in accordance with the rules in this document. You can create more customer nodes within the existing COMOS iDB structures using the letter Z and COMOS iDB counting. See Chapter Customer-specific additions (Page 247).

7.9 Controlling base object links using dereferencing

Aim

The "Dereference" option is displayed if the "Base object" field has been set. See chapter Administering properties of elements in the base object (Page 40).

- The fields "Name", "Label", and "Description" are applied to the engineering object from the base object from which it is derived. All other properties are applied from the base object entered in the "Base object" field as the base object.
- The information about which base object was used to derive the engineering object originally is lost. Instead, the connection to the base object referenced in the "Base object" field is saved at the engineering object.
Any future changes to the base object from which an engineering object was originally derived have no effect on the engineering object. Changes to the base object referenced in the "Base object" field are applied.

The "Dereference" option is mainly used in the case of labeling systems. Labeling systems are constructed using base object links from other branches.

Using "dereferencing" in labeling systems

The following rule applies when constructing labeling systems:

- If the base object in the "Reference > Base object" field is applied without changes, then:
The "Dereference" option is activated.
Effect:
 - In the right mouse button context menu of the engineering object, the structure from the @30 node is provided first.
 - However, the objects are created from the @10 node when they are created.
- If standard-specific customizations are made in the labeling system, then:
The "Dereference" option is deactivated.

Examples

- Official labeling systems (KKS, RDS-PP, etc.)
For general objects, the "Dereference" option is set to "No"
For device objects, the "Dereference" option is set to "Yes"
- General ET labeling
"Dereference" option set to "No"

See also

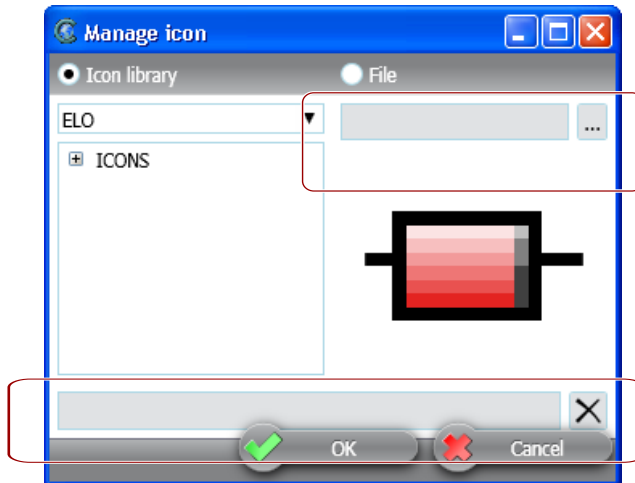
Labeling systems in the M00 branch (Page 175)

7.10 Standard icon of an object class

Assigning an icon to a COMOS base object by means of an object class

Default: The icon is read out from Picture.dll based on the object class.

No path is entered in the "Manage icon" dialog:



See also

Setting the object icon manually (Page 47)

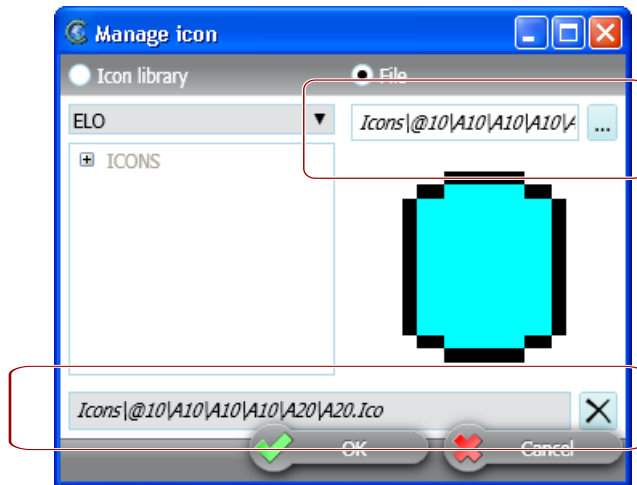
7.11 Setting the object icon manually

Requirement

- You are familiar with the standard icons of an object class.
See chapter Standard icon of an object class (Page 47).

Assigning an ico file from the file system to a COMOS base object

An ico file is selected in the "Manage icon" dialog:



General graphical properties of self-created icons:

- 16x16 pixels
Only icons with the size 16x16 are officially supported. Other formats are not displayed in all parts of COMOS.
- 32-bit color depth
Reason: Support for alternative displays (working layer display mode, status displays, base data display)
- 72 dpi
- Alpha
- Scaling
The location being used in COMOS determines how the image is displayed
- Background: Transparent
In some cases, COMOS uses the color of the pixel in the bottom left corner as the transparency color. This also applies when the "Transparent" property is set in the icon.
Exception: the color black

Note: Limit the number of different icons because the icons have an effect on storage management.

You can find additional information on the use of the "Manage icon" COMOS user interface in the "Administration" manual, keyword "Assign object icon".

Default storage in the iDB

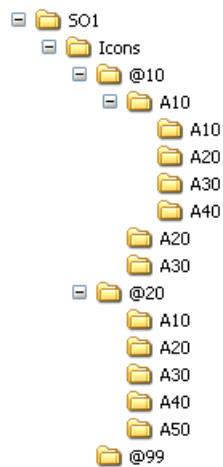
As of COMOS 10.2.1 storage of the icons has been standardized in the iDB. Icons are no longer stored multiple times.

The following conditions apply to the standardized icons:

- Storage in the file system: "<database> SO1 > ICONS > IconMasters"
- Name template: "Y00innnnn"

- Storage in the base data: "@20 > D50 Icon library"
- Change management: Excel document "@20 > D50 Icon library > map_icons"
 - Column "A"
Path and name of the replaced icon.
 - Column "B"
Checksum of the standardized icon
 - Column "C"
Path and name of the standardized icon. Multiple icon entries in column "A" can be replaced with the same standardized icon.
- You determine the relative path of the icon file with the script commands "IconFileName" and "OwnIconFileName".
Example of a return value: "Icons\@00\@40.Ico"
- The script command "Project.GetDocumentDirectory" returns the path to the external storage of the documents and other files.

Archiving of the ico files prior to COMOS 10.2.1.



Outdated: Assignment using the "icon library"

This variant only continues to be supported for compatibility reasons and is not permitted in the COMOS iDB.

Additional information on external files

See chapter Using or editing external files (Page 293).

7.12 Overview for administration of symbols

- Interaction of symbols and drawing types
See chapter Drawing type-specific symbol display (Page 50).
- Using generic symbol
See chapter Using generic symbol with variable number of connectors (Page 51).
- Information on export of symbols to AutoCad
See chapter Preparing symbols for AutoCAD export (Page 53).
- Storage of symbols in branch @20
See chapter @20 > C30 Symbol construction (Page 80).

7.13 Drawing type-specific symbol display

Aim

The aim in this case is to display objects differently for different drawing types. Module-specific display characteristics are created at the object for this purpose. Standard-specific displays are also possible.

Procedure

The naming of the individual drawing types is divided into three blocks:

SAMPLE: M20_P1

1. Three-digit module label
The module letters are used here (e.g. M20). This name is also used here if you are dealing with a customer-owned module starting with a "Z". (e.g. Z00) If you are dealing with a drawing type that is used by several modules, it is assigned to the Basic module.
2. Separators
Blocks 1 and 2 are separated with the "_" separator.
3. One-digit sequence number with leading "P"
This block uses a single-digit sequence number. Customer-specific symbols can be added by using the suffix "_Z" (e.g., P1_Z).

This means the following three cases apply:

M20_P1	Standard drawing type
Z00_P2	Customer drawing type
M20_P1_Z	Customer-specific extension of a standard drawing type

Exception for the 3D area:

M20_P1_-XY

M20_P1_-XZ

M20_P1_-YZ

The respective module affiliation is described in the drawing type column:

The screenshot shows a tree view on the left with the following structure:

- recovery
 - Y10 System tables
 - M00 COMOS general system tables
 - A10 System tables 00001 - 00025
 - Y10M00N00001 Symbol type
 - Y10M00N00002 Line type
 - Y10M00N00003 Line width
 - Y10M00N00004 RGB colors
 - Y10M00N00005 Yes/No
 - Y10M00N00006 Contact point types
 - Y10M00N00007 Contact point types opti
 - Y10M00N00008 User-defined line types
 - Y10M00N00009 Drawing type specific lin
 - Y10M00N00010 Line types for action line
 - Y10M00N00011 Nozzle relevant activities
 - Y10M00N00012 Activity relevant specific

The detailed view on the right shows the following fields:

Name: Y10M00N00001
 Description: Symbol type
 Default value: Check symbols for revision

Below this is a table with the following data:

Name	Beschreibung	Planart M00_P1
M00_P1	Check symbols for revision	
M21_P1	PFD	
M22_P1	P&ID (ISO 10628)	
M22_P2	P&ID (DIN 2481) KKS Stand:	
M22_P3	P&ID (DIN 2481) KKS Stand:	

This description is to be done in several languages.

Relationship with the drawing types

If a module or an operation mode is to have its own display, a drawing type must be available for it.

Suppress placement of symbols on datasheets

If you want to display notes on a datasheet, you need to create the document as an interactive report. If no further measures are taken, symbols are also be placed on datasheets.

To suppress the placement of symbols on datasheets, the undefined drawing type "DATASHEET" is entered on the datasheets.

7.14 Using generic symbol with variable number of connectors

Aim

If a COMOS engineering object has more prepared connectors on the "Connectors" tab than in the symbol of the base object, you can use the generic symbol. The following procedure describes how you replace the symbol with the generic symbol for a drawing type in an existing base object.

Operation mode of the generic symbol

In a generic symbol, the number of connectors is not defined by the base object. Instead, the connectors are identified for each engineering object and the symbol on the report is changed automatically.

The generic symbol consists of the following components:

- Appearance of the symbol on the report
 - Rectangle
 - Label left
 - Inputs on top
 - Outputs on bottom
- Connectors of the engineering object taken into consideration
 - Properties of the connector: I/O field is "Input" or "Output"
 - Properties of the connector: "Type" field corresponds to the drawing type with the generic symbol

Transferring generic symbol to existing base object

Symbols are scripted in COMOS. This means that transferring a symbol is the same as transferring the symbol script.

Note

Existing symbol script is overwritten

The symbol script for the respective drawing type that is being edited is overwritten in the target base object. Based on the inheritance in COMOS, the following locations in the engineering projects change:

- All engineering objects that are derived from this base object.
 - All reports on which a corresponding engineering object is placed.
-

You transfer the symbol script to the required base object with the "A10 Generic symbol" object.

1. Navigate to the object: "@20 > C30 > M00 > A10 > Generic symbol".
2. Open the "Attributes > Symbol configuration" tab in the properties of the object.
3. In the "Link to source base object" field, select the object "@20 > C30 > M00 > A10 > Generic symbol".
4. In the "Link to target base object" field, select the base object to which you want to transfer the generic symbol.
Example: The base object of a contactor.
5. In the "Drawing type" list, select the drawing type for which the generic symbol is to be transferred.
6. Optional: Activate the "Text script" option.
With this option, you copy the text script to the target base object instead of the symbol script.
7. Click the "Copy symbol script" button.

Result

The script for the generic symbol is transferred to the target base object. The derived engineering objects get a new symbol. The new symbol is displayed on the reports.

7.15 Preparing symbols for AutoCAD export

If a symbol contains white areas (filled hatching), no pure white may be used (255,255,255) as AutoCAD interprets this as transparent. A "semi-white" such as 250,250,250 must be used instead of this.

7.16 Connector types in the iDB

7.16.1 Purpose of the connector types

Definition

Connectors are special objects in COMOS below an object. They represent the connection points of an object. Two objects can be connected with connectors this way. Examples include process engineering connectors for pumps and electrical connectors for motors.

Important properties of connectors

- Name
See chapter Reference: Names of the connector types (Page 54).
- Label
A connector either has its own label or it takes on the label of the owner.
- Description
Allocating a description to a connector is not prescribed as mandatory but can be stated where required.
Example:
Module EI&C, Logical:
 - Symbol graphics
 - Connection list
 - Signal list
- Input/Output
One property of a connector is "I/O". This property identifies the connector as input, output or neutral (and also each version in "intrinsically safe"). You can also enter a type here. In this way, a connector of the "Single line" type cannot be connected with a connector of a different type. Furthermore, there are regulations defined for the permissible connections between connectors.
- Type/Subtype

Links

You can find additional information on this topic in the "Administration" manual, keyword "Configuring connectors".

7.16.2 Reference: Names of the connector types

Naming system for connectors

The name given is neutral and is assigned for each module.

Attention must be paid to the length of the name as the name will be displayed on different plans.

Connectors are generally independent of the drawing type, i.e. a connector may be displayed on different drawing types. In some cases connectors are only displayed on one drawing type or on no drawing type at all.

Structure of the name: 3 or 4 characters (the first two characters are letters; the following characters are figures with consecutive numbering):

Note

The consecutive numbering may have one or two digits.

1. Position	2. Position	3. Position	4. Position	Note
Letter	Letter	Number	Number	
A				Basic
B				Basic
C				Process
D				Process
E				Automation
F				Automation
G				Operations
H				Operations
Z				Often used for customer-specific connectors
	A			Free naming within each module
	...			Free naming within each module
	Z			Free naming within each module
		0	1	Consecutive numbering (one or two digits)
		0	2	Consecutive numbering (one or two digits)
			...	Consecutive numbering (one or two digits)
		0	9	Consecutive numbering (one or two digits)

1. Position	2. Position	3. Position	4. Position	Note
Letter	Letter	Number	Number	
		1	0	Consecutive numbering (one or two digits)
			...	Consecutive numbering (one or two digits)

The first digit is for the purposes of assignment to the 4 higher-level areas in COMOS. The letter "Z" is reserved for customer-specific connectors. The second digit is assigned within an area. After this the number is consecutive (with one or two digits).

Fixed names in the Automation area

ET Multiline	EAXx
ET Singleline	EBxx
ET Bridge internal	ECxx
ET Bridge external	EDxx
ET Reserve	EExx – EHxx (4 Reserve)
IC Multiline	EIxx
IC Singleline	EKxx
IC Reserve	ELxx – EOxx (4 Reserve, when J is not used)

Fluid Hydraulic	FAXx
Fluid Pneumatic	FBxx
Fluid Lubrication	FCxx
Fluid Reserve	FDxx-FHxx (5 Reserve)
Logical FUP input	FIxx
Logical FUP output	FKxx
Logical Reserve	FLxx

There are more exceptions in the area of automation for connector arrays and PE connectors.

Fixed names in the M20 Process area

Old name	New name	Note	Module
CX0	CX0	for origin connectors (never used in BaseProject)	VIPER
CX1	CX1	Viper ComosConnectors	VIPER
CX2	CX2	Viper ComosConnectors	VIPER
CX3	CX3	Viper ComosConnectors	VIPER
CX4	CX4	Viper ComosConnectors	VIPER
CA1	CA1	for actuator connectors	VIPER
CA2	CA2	for actuator connectors	VIPER
CA3	CA3	for actuator connectors	VIPER
CA4	CA4	for actuator connectors	VIPER
CAn	CAn	for actuator connectors	VIPER
CY1	CY1	for dynamic connectors (never used in BaseProject)	VIPER

7.16 Connector types in the iDB

Old name	New name	Note	Module
CY2	CY2	for dynamic connectors (never used in BaseProject)	VIPER
CY3	CY3	for dynamic connectors (never used in BaseProject)	VIPER
CY4	CY4	for dynamic connectors (never used in BaseProject)	VIPER
CYn	CYn	for dynamic connectors (never used in BaseProject)	VIPER
CP1	CP1	for physical connectors (never used in BaseProject)	VIPER
CP2	CP2	for physical connectors (never used in BaseProject)	VIPER
CP3	CP3	for physical connectors (never used in BaseProject)	VIPER
CP4	CP4	for physical connectors (never used in BaseProject)	VIPER
CPn	CPn	for physical connectors (never used in BaseProject)	VIPER
CM1	CM1	for attachment connectors (for mounting parts like nuts or slip on flanges)	VIPER
CM2	CM2	for attachment connectors (for mounting parts like nuts or slip on flanges)	VIPER
CM3	CM3	for attachment connectors (for mounting parts like nuts or slip on flanges)	VIPER
CM4	CM4	for attachment connectors (for mounting parts like nuts or slip on flanges)	VIPER
CMn	CMn	for attachment connectors (for mounting parts like nuts or slip on flanges)	VIPER
C#1	C#1	for CP/CX in IsoSymbols.	VIPER
C#2	C#2	for CP/CX in IsoSymbols.	VIPER
C#3	C#3	for CP/CX in IsoSymbols.	VIPER
C#4	C#4	for CP/CX in IsoSymbols.	VIPER
C#n	C#n	for CP/CX in IsoSymbols.	VIPER
CX#	CX#	for connectionTypes in IsoSymbols	VIPER
CP#	CP#	for connectionTypes in IsoSymbols	VIPER
I1	CI1	input connector	FEED
O1	CO1	output connector	FEED
O2	CO2	output connector	FEED
O3	CO3	output connector	FEED
O4	CO4	output connector	FEED
001	CN1	dynamic and/or neutral connector	FEED
002	CN2	dynamic and/or neutral connector	FEED
003	CN3	dynamic and/or neutral connector	FEED
004	CN4	dynamic and/or neutral connector	FEED
W	CS1	signal line connector	FEED
W1	CS2	signal line connector	FEED
W2	CS3	signal line connector	FEED
W3	CS4	signal line connector	FEED
W4	CS5	signal line connector	FEED
Wn	CSn	signal line connector	FEED
I1	DI1	input connector	P&ID
I2	DI2	input connector	P&ID
O1	DO1	output connector	P&ID

Old name	New name	Note	Module
O2	DO2	output connector	P&ID
001	DN1	dynamic and/or neutral connector	P&ID
002	DN2	dynamic and/or neutral connector	P&ID
W	DS1	signal line connector	P&ID
W1	DS2	signal line connector	P&ID
W2	DS3	signal line connector	P&ID
W3	DS4	signal line connector	P&ID
W4	DS5	signal line connector	P&ID
Wn	DSn	signal line connector	P&ID

Comparison of names in the old COMOS database and COMOS iDB

Old connector name in DB V10.0 and below	New connector name in the COMOS iDB
SLI	EB01
SLO	EB02
SL1	EB01
SL	EB01
SL1	EB01
SL11	EB02
CPxx	EAx
BI[1]	EC[1]
BI[2]	EC[2]
BE[1]	ED[1]
BE[2]	ED[2]
I	I
I[3]	I[3]
I1	I1
I2	I2
IN1	EA01
IN2	EA02
O	O
O[1]	O[1]
O1	O1
O2	O2
O3	O3
OUT1	EA02 or EA03
PE	EE01

See also

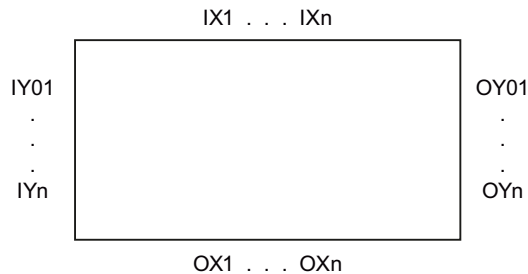
Purpose of the connector types (Page 53)

7.16.3 Connectors on EI&C reports

Dynamic connectors on EI&C reports

If you move an object which does not have a symbol onto an EI&C report via drag&drop, a "black box" is created as the symbol for this object.

The following picture shows a black box:



A black box has dynamic connectors. You can use the dynamic connectors from all sides at the same time.

If you wire the black box, the connectors are created during runtime. To enable COMOS to assign the location, a distinction is made between X/Y and I/O. Because wiring of this type does not take place until the project planning stage, this naming does not exist in the base project.

7.17 General object class

The General object class (TopLevel)

Background: When objects are displayed in the Navigator, these objects will be presented only on one tab in most cases. The tab "owns" the object. You can set a "reference" on the other tabs, but these references are not the original object.

In contrast to this, an object with the "General" class is always displayed on all tabs. The corresponding icon adjusts to the tabs in question. Technically, however, in spite of the different icons, this is always the same object.

An object of the "General" class must be located below the project or below another object of the "General" class.

The objects are displayed in the Navigator depending on the following attributes of the "Check box" type:

- "SYS.InvisibleInUnitTree"
- "SYS.InvisibleInLocationTree"
- "SYS.InvisibleInDocumentTree"

The attribute has to be set at the base object, the setting at the engineering object is not evaluated.

7.18 Deleting objects

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Overview of deletion of objects".

7.19 Note on obsolete base data

See chapter Obsolete base data (Page 291).

Administering "@10 Requirement objects"

8.1 Purpose of the "@10" node

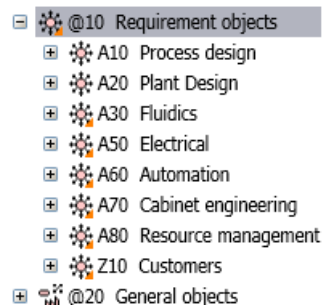
The "@10" node includes the standard-neutral COMOS base objects and requirement objects. These objects are reduced to the permitted minimum.

For the general method of requirements refer to the "EI&C Administration" manual, keyword "Preparing requirements".

8.2 Structure of the "@10" node

Objects are sorted by COMOS topic, COMOS module and object type in the "@10" node.

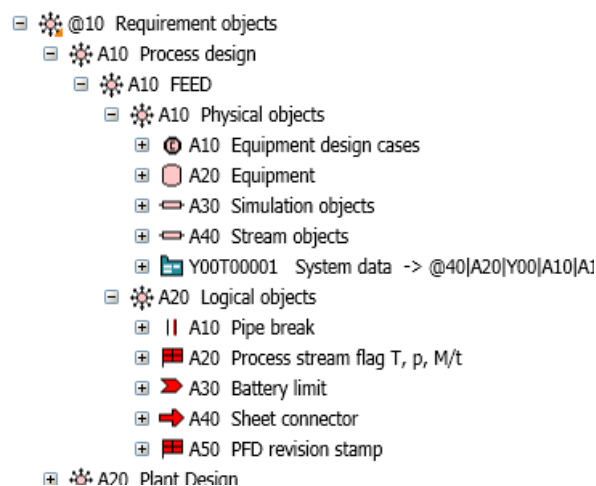
1. First level and possibly second level: Sorting by COMOS topic, then by COMOS module:



2. Second or third level: a distinction is made between the "Physical objects" and "Logical objects" nodes for all branches.

This is where the requirement objects are sorted:

- "A10 Physical objects"
- "A20 Logical objects"



NOTICE

Rules for cross-inheritance

There may only be a cross-inheritance from the "@10" node to the "@20" or "@30" node. Cross-inheritance within the "@10" node is only permitted as an exception.

8.3 Editing options of the "@10" node

All settings of the base objects that are required for the basic functionalities are edited in the "@10" node:

- General information on the base object:
 - Description
 - Class / subclass
- Classification
- Tabs with system access (hardcoding)
- System-related tabs
- Graphic-related tabs
- Symbols of different drawing types (if available)
- Connectors (as required for basic functionalities)
- Elements (as required for basic functionalities)

8.4 Use/expansion of the "@10" node

Node @10	Settings of base objects required for basic functionalities are edited here. See chapter Editing options of the "@10" node (Page 62).	
	@10 is used in:	
	"@30" node	Expansions for specific standards and topics: <ul style="list-style-type: none"> • General information on the base object: <ul style="list-style-type: none"> – Description – Name template • Expansions of the tabs: <ul style="list-style-type: none"> – Standard-specific forms – Example attribute (Technical data) – Device-specific forms • Connectors • Elements / element structures • Symbols
	@10 is used in:	
	"@50" node	Device-specific expansions: <ul style="list-style-type: none"> • General information on the base object: <ul style="list-style-type: none"> – Name – Name template – Description • Input of data in tabs • Connectors • Elements / element structures • Symbols

See also

Names and meaning of the main levels in the iDB (Page 21)

8.5 Notes on expansion of tabs

Requirement

- You are familiar with the use/expansion of the @10 node. See chapter Use/expansion of the "@10" node (Page 63).

8.6 Example for use of the "@10" node

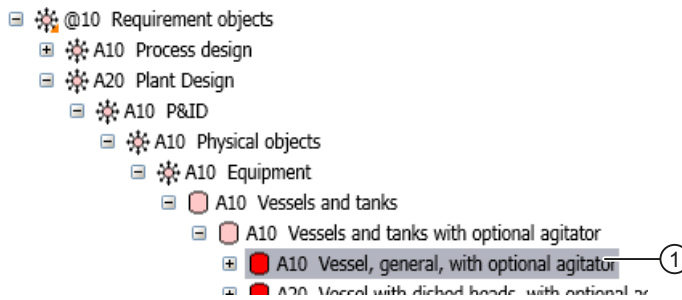
Criteria for the tabs

Node @10	Criteria for tabs and attributes: <ul style="list-style-type: none"> • With system access (hardcoding) • System-related • Graphic-related 	
	@10 is used in:	
	Node @30	Criteria for tabs and attributes: <ul style="list-style-type: none"> • Standard-specific forms • Example attribute (Technical data)

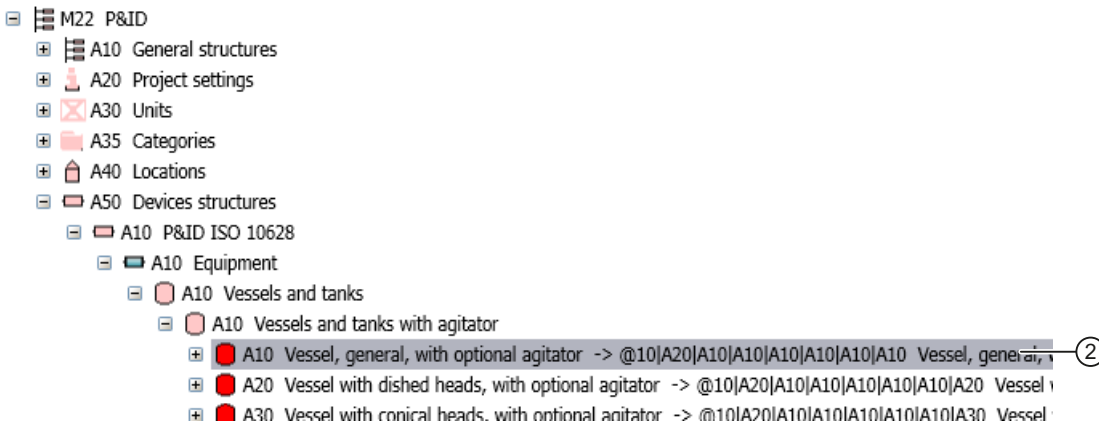
Objects are only created with system-related information in the "@10" node. These objects are then used in standard-specific structures in the "@30" node and additional information is added.

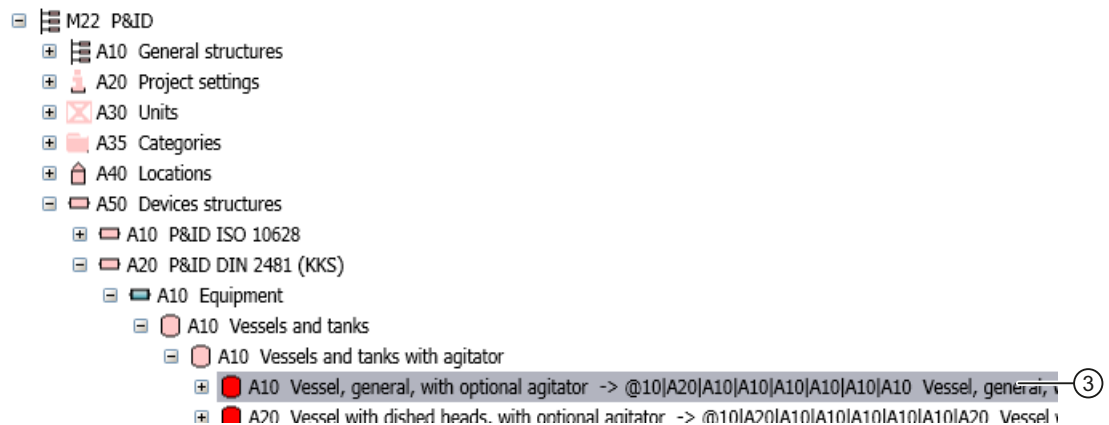
8.6 Example for use of the "@10" node

The base object of a vessel is prepared in the "@10" node:

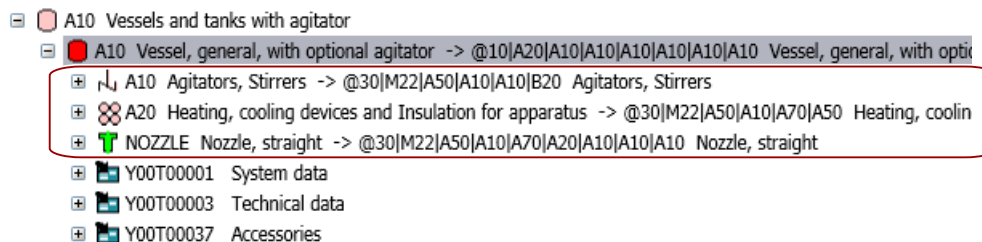


Base object 1 only has symbols, settings, attributes, and connectors that are independent of standards. Base object 1 is used in different device structures in the "@30" node (2 and 3):





Base objects (2) and (3) are developed on a standard-specific basis. This involves, for example, inserting standard-specific elements which were not present for base object (1) (using base object 2 as example):



Standard-specific information is constructed in node @30, where (with reference to this example) it is used for base object (2) and base object (3). Because the objects in the standard-specific node of @30 have a reference to objects in the node in @10, the inheritance is interrupted at this point. This means it is not possible to prepare the manufacturer devices in node @30 at an upper node. Instead, the following applies to each base object with a reference to an @10 base object: The standard-specific elements and/or attributes must always be built into the @30 structure using a reference. See section Notes on multiple use of "@10" objects (Page 65).

See also

Device structure A50 below the modules (Page 102)

8.7 Notes on multiple use of "@10" objects

The hierarchical inheritance is interrupted due to the cross-inheritance of the objects in the "@30" node. This means standard-specific elements and standard-specific attributes must be created new for each object in the "@30" node.

Example "@30 > M21 > A50 > A10 > A10 > A30 > A30":

8.7 Notes on multiple use of "@10" objects



The different elements and tabs were created in object A10 (item 1).

These standard-specific elements and tabs must be created once again for objects 2, 3 and 4. You cannot create the elements and tabs on a higher level and inherit them to objects 1 to 4.

This procedure creates several copies of the elements and attributes. However, all elements and attributes of objects 1 to 4 are based on the same source objects from @30 and @40. When these source objects in @30 and @40 are updated, all uses in the objects 1 to 4 mentioned above are updated as well.

"@20 General base objects": Editing basic base objects

9.1 Purpose of @20

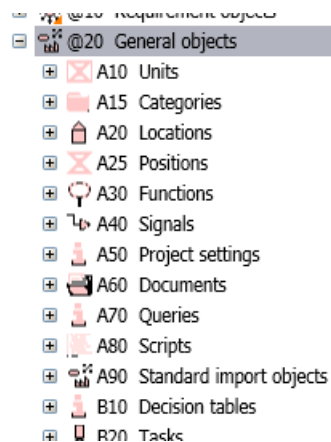
Purpose of the "@20 General objects" node

This node contains base objects for logical objects such as units, functions, queries, etc. As with the @10 node there are no standard-specific structures here either.

Structure and order of the "@20 General objects" node

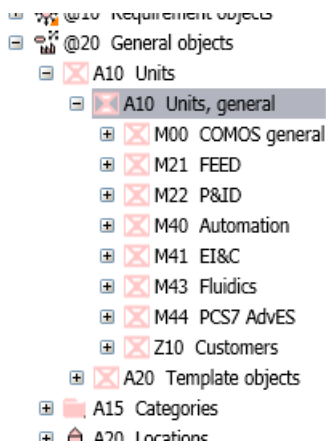
Different sorting criteria are permitted for nodes below @20:

- Sorting according to COMOS object types with specified class/subclass
Examples: Units, categories
- Sorting according to base objects related by topic with different object configuration
Examples: SAP objects, PDMS objects



Additional subnodes for specific objects of the relevant COMOS modules can exist below the subnode for the "@20 General objects" node. Only the structures required for the COMOS modules are created at this stage in order to increase clarity. Missing nodes can be added, if necessary.

Example of a COMOS-module specific subnode structure:



Basic rules for all objects in the "@20 General objects" node

The sub nodes of the @20 "General objects" node may only be created with the "Structure" option. Class/subclass and classification are set at these structure objects. These structure objects may not have elements. General base objects are created below. The order and structure of the "General objects" are explained in the sections for the individual object groups. ##??

9.2 Interface objects for base data import (dynamic objects)

Aim

Base objects which change again and again when importing takes place are called dynamic base objects. Interface objects that create dynamic base objects (i.e. import base data) can be prepared in the database.

The interface objects are stored below the following nodes:

- @20 > A90 Standard import objects
- @20 > D20 Base objects from external CAE files

The nodes are explained in detail below.

@20 > A90 Standard import objects

General import interfaces and module-specific import interfaces are stored in the "@20 > A90 Standard import objects" node. Only the interface objects are prepared in this node, base data are not imported into it.

- A90 > M00 > A10
Excel import for manufacturer catalogs
- A90 > M40 > A10
Import for Automation Hardware

@20 > D20 Base objects from external CAE files

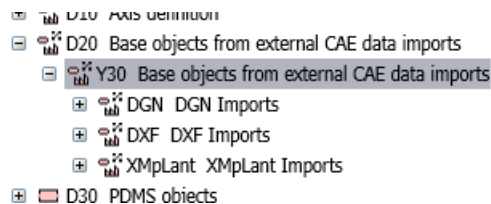
The dynamic objects from CAE data are stored in the "@20 > D20" node.

You can create a node for each import interface below the "@20 > D20 > Y30 Base objects from external CAE files" node.

The names of the nodes below the "@20 > D20 > Y30 Base objects from external CAE files" node should be based on the imported file extension:

- PDMS
- DXF
- DWG
- IDF
- XM plant

Example:



The name can also include specific extensions.

Note

No maintenance of the Y30 node

Note that the contents of the "@20 > D20 > Y30 Base objects from external CAE files" node are not maintained by Siemens.

Some import interfaces are prepared in the COMOS iDB.

Tabs and attributes of the imported base objects

The tabs and attributes of the imported base objects are created and adapted during the import. They are not known beforehand.

See section Dynamically created attributes (Page 133) and section Dynamically created tabs (Page 134).

9.3 Reference: "@20 General base objects"

9.3.1 @20 > A10 Units

Overview

Objects below this node should have the class "Unit" and the subclass "(None)". This means the parameter settings for this node are as follows:

Class: Unit
Subclass: (None)

The following three nodes exist below this node:

- A10 Units, general
- A20 Template objects
- A30 Categories

@20 > A10 Units > A10 Units, general:

The following settings apply here:

Class: Unit
Subclass: (None)

Base objects for units/subunits are created here which are then used in the specific tagging systems.

These base objects are stored specific to the COMOS module.

@20 > A10 Units > A20 Template objects:

The following settings are made as of the Templates node:

Class: Unit
Subclass: (None)
Name template: @"\T\elm\p\la\tle"
Name template specified value: "@Template"

General

Class: Unit Subclass: (None)

Name: A20 \@\T\e\m\p\1\ i @Template ...

Label: i ...

Description: Template objects

The name template with the specified value "@Template" must be set so that the name @Template is guaranteed at the engineering end for certain software functions to work.

@20 > A10 Units > A30 Categories:

The following settings apply here:

Class: Unit
Subclass: Category

Here you create the category folder required for the "Units" tab.

The individual COMOS modules are located below the categories.

9.3.2 @20 > A15 Categories

Overview

The following settings apply here:

Class: Unit
Subclass: Category

The base objects for category folders are stored here. The base objects are sorted by COMOS modules.

9.3.3 @20 > A20 Locations

Objects below this node should have the "Location" class. The subclass depends on the use of the base object.

Class: Location
Subclass: (variable)

Both the following nodes exist below this node:

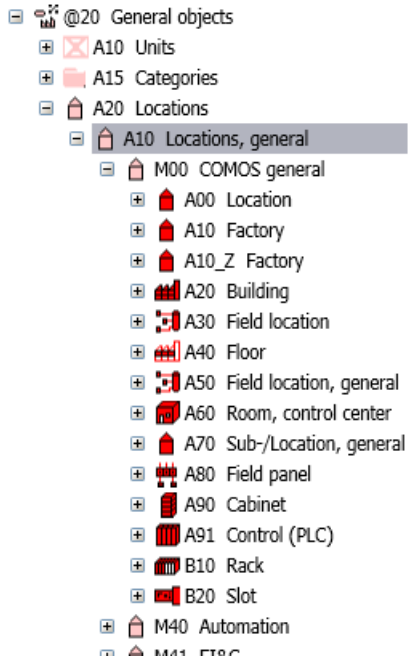
- A10 Locations, general
- A20 Template objects

@20 > A20 Locations >A10 Locations, general

The following settings apply here:

Class: Location
Subclass: (variable)

Underlying base objects are stored here specific to the COMOS module for engineering in the "Locations" tab. The base objects are sorted by COMOS modules:



@20 > A20 Locations >A20 Template objects

The following settings apply here:

Class: Location
Subclass: Category
Name template: "@\T\l\m\p\l\alt\l"

The name template with the specified value "@Template" must be set so that the name @Template is guaranteed at the engineering end for certain software functions to work.

9.3.4 @20 > A25 Positions

The base objects for the positions are stored here specific to the COMOS module.

The following settings apply here:

Class: Position
Subclass: (None)

An ALIAS node is prepared for positions. For more on the topic of ALIAS, see Chapter Administration of the ALIAS labeling system (Page 178).

9.3.5 @20 > A30 Functions

The base objects for the functions and function elements are stored here.

The following settings apply here:

Class: Function
Subclass: (None)

The following nodes exist below this node:

- A10 Predefined functions
- A20 Structure elements for functions

9.3.6 @20 > A40 Signals

The base objects for the signals are stored here.

The following settings apply here:

Class: Signal
Subclass: (None)

The base object for the general signal is located below this node:

- A10 General signal

9.3.7 @20 > A50 Project settings

The following settings apply here:

Class: Dataset
Subclass: (None)

The following sub nodes exist below this node:

- A10 Project settings, general

@20 > A50 Project settings > A10 Project settings, general

The project settings are stored here.

Note

It only includes the descriptive attributes.

9.3.8 @20 > A60 Documents

The base objects for documents are stored here. You can find more detailed information on the base objects for the documents at "Base objects for documents and document groups (Page 197)".

The following settings apply here:

Class: Dataset
Subclass: (None)

9.3.9 @20 > A70 Queries

Overview

The following settings apply here:

Class: Dataset
Subclass: (None)

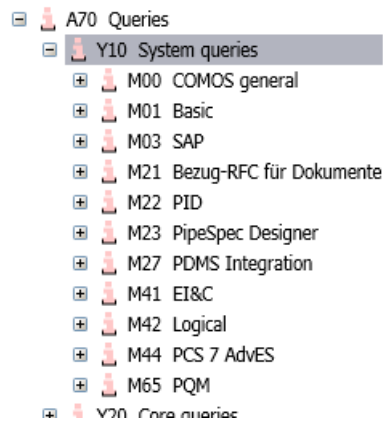
Queries are stored here which are type-specific and specific to the COMOS module:

- A70 > Y10 System queries
- A70 > Y20 Core queries
- A70 > Y30 Customer queries for interfaces
- A70 > Z10 Customer queries

Substructures

A distinction is made between the following types:

- Y10 System queries
System queries are stored here specific to the COMOS module.
(Queries used by the software).



- Y20 Core queries
The queries supplied by COMOS specific to the COMOS module are stored here:
 - A70 > Y20 > M00
Queries which are available in the COMOS menu
 - A70 > Y20 > Mxx
Queries which are used by documents or objects. The meanings of these queries are described in the specialist COMOS manuals.
- Y30 Customer queries for interfaces
This is the storage location of queries that have fixed names due to an interface or use tabs and attributes, the names of which are not structured according to the existing guidelines.
 - A70 > Y30 > M03 SAP
Queries for SAP
- Z10 Customer queries
Customer-specific queries are stored here.
 - Z99 Menu
This is where queries for customer-specific expansions to the COMOS menu should be stored.
The COMOS menu displays the content from both the "@99 > A20 > M00 > A40" node as well as the Z99 node. If the nodes do not exist or are empty, the COMOS "Extra > Query" menu at the corresponding locations is empty.

9.3.10 @20 > A80 Scripts

The following settings apply here:

Class: Action
Subclass: Script

Scripts are stored here specific to the COMOS module.

You can find more information on scripts under "Developing scripts for the iDB (Page 207)".

9.3.11 **@20 > A90 Standard import objects**

The following settings apply here:

Class: Dataset

Subclass: (None)

This is where the import interfaces are stored specific to the module.

See also

Interface objects for base data import (dynamic objects) (Page 68)

9.3.12 **@20 > B10 Decision tables**

The following settings apply here:

Class: Dataset

Subclass: (None)

Decision tables are stored here specific to the COMOS module.

9.3.13 **@20 > B20 Tasks**

The following settings apply here:

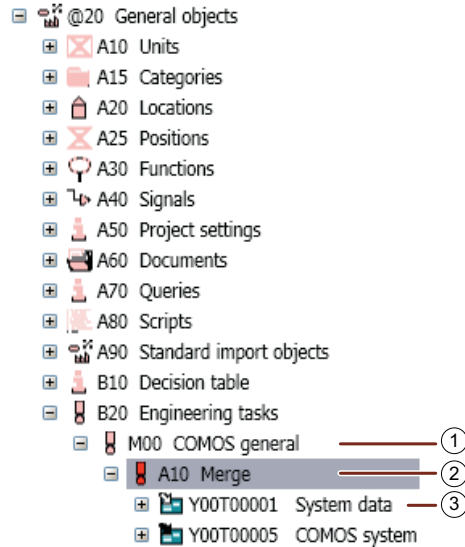
Class: Dataset

Subclass: (None)

The engineering tasks (eBlocks) are sorted into module-specific folders:

- "B20 > Y10 > M00"

The basic types of engineering tasks are sorted in the "M00" node.



①	Task class
②	Engineering task
③	Tab for controlling the engineering task

For administration of engineering tasks, see "Administration" manual, keyword "Preparing engineering tasks".

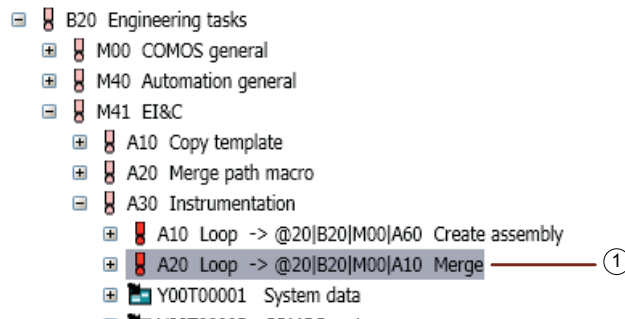
For an overview of the basic tasks, see "Administration" manual, keyword "Reference of predefined task functions (basic types)".

- "B20 > Y10 > M40", "B20 > Y10 > M41" and more

By means of cross-inheritance, a basic type is inherited in the module-specific folders and is adapted there (renamed, for example).

Example:

The engineering task "A10 Merge" from node M00 is inherited in node M41 at position (2) and is renamed there to "A20 Control loop":



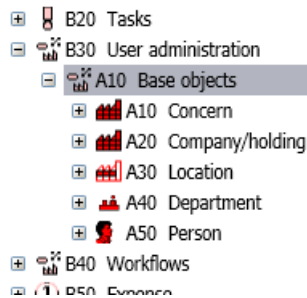
Engineering tasks that do not rely on an object from M00 may also be created in the M40, M41, etc. nodes.

9.3.14 @20 > B30 User administration

The following settings apply here:

Class: Dataset

Subclass: (None)



The base objects for user administration are stored here.

9.3.15 @20 > B40 Workflows

The following settings apply here:

Class: Dataset

Subclass: (None)

The base objects for workflows are stored here.

The following sub nodes exist underneath the "M01 Basic" node:

- A10 Workflow Tasks
Master objects for the individual tasks in a workflow.
- A20 Workflows
Base objects for a workflow. The tasks in a workflow are inserted as elements.
- A30 @ProjectManagement Workflows in Project
The base object "A30 @ProjectManagement Workflows in Project" with the name "@ProjectManagement" must be inserted on the "Units" tab under the project in order to use workflows.

9.3.16 @20 > B50 Expenses

The following settings apply here:

Class: Costs

Subclass: (None)

Cost objects are stored here specific to the COMOS module. Example:

- B50 > M41 > A10 Expense example

Note

The maintenance costs are determined with a DLL.

9.3.17 @20 > B60 Configuration and mapping objects

The following settings apply here:

Class: Dataset
Subclass: (None)

Special import interfaces are constructed in this node. In contrast to A90, in node B60 several base objects are required which together form an import interface.

- B60 > M06
The base objects for the Teamcenter interface are stored underneath this node.
The following subnodes exist here:
 - B60 > M06 > A10 Interface objects
Here you store the definitions of the configuration objects of the Teamcenter interface.
 - B60 > M06 > A20 Object assignment
The resources, attributes and units used in Teamcenter are imported here and mapped on base objects, attributes and units in COMOS.

9.3.18 @20 > B70 Installation

The following settings apply here:

Class: Mounting
Subclass: (None)

The base objects for installation objects are stored here specific to the COMOS module.
Example:

- B70 > M41
Objects for hook-up reports

9.3.19 @20 > B80 Routing

The following settings apply here:

Class: Line layout
Subclass: (None)

This is where the base objects are stored, specific to the module, for routing.

9.3 Reference: "@20 General base objects"

Routing is currently only available for EI&C:

- B80 > M41 > A10
Point to point connection

9.3.20 @20 > B90 Risk-, safety evaluation

The following settings apply here:

Class: Dataset
Subclass: (None)

The base objects for "Risk-, safety evaluation" are stored here specific to the COMOS module.

9.3.21 @20 > C10 Locking

The following settings apply here:

Class: Modify
Subclass: (None)

This is where the base objects are stored, specific to the module, for the interlock line.

The interlock list is currently only available for EI&C:

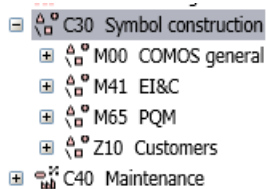
- C10 > M41 > A10 Locking

9.3.22 @20 > C30 Symbol construction

The following settings apply here:

Class: Element
Subclass: Graphic

This is where the base objects are stored, specific to the module, for special symbols.



- C30 > M00
 - Generic symbol
See chapter Using generic symbol with variable number of connectors (Page 51).
- C30 > M41
 - Symbols in the style of the IEC 60617 standard
 - Symbols which are used on control cabinet plans
- C30 > M65
 - Symbols which are used for redlining revision files in PDF format

9.3.23 @20 > C40 Maintenance

Comprehensive catalog with base objects for the "Maintenance" area.

The following settings apply here:

Class: Dataset
Subclass: (None)

9.3.24 @20 > C50 General objects

The following base object will be available here for future versions of the database:

- General object class (TopLevel)

The following settings apply here:

Class: General
Subclass: (None)

When objects are displayed in the Navigator, these objects will be presented only in one tab in most cases. The tab "owns" the object. You can set a "reference" on the other tabs, but these references are not the original object. In contrast to this, an object with the "General" class is always displayed on all tabs. The corresponding icon adjusts to the tabs in question. Technically, however, in spite of the different icons, this is always the same object.

An object of the "General" class must be located below the project or below another object of the "General" class. The objects are displayed in the Navigator depending on the following attributes:

- "InvisibleInUnitTree"
- "InvisibleInLocationTree"
- "InvisibleInDocumentTree"

The attributes have to be edited at the base object, because the settings are not evaluated at the engineering object.

9.3.25 @20 > C70 Automation objects

Comprehensive catalog with base objects for PCS 7/Automation objects.

The following settings apply here:

Class: Dataset

Subclass: (None)

- C70 > A10
Only available in COMOS for compatibility reasons.
- C70 > A20
The objects that appear in the Automation Tree are created and stored here.
- C70 > A30
The automation objects are created and stored here. These are used by the PCS7 interface and the FUP objects and are required for a data comparison between PCS7 and Comos.

9.3.26 @20 > C80 SAP objects

Comprehensive catalog with base objects for SAP objects.

The following settings apply here:

Class: Dataset

Subclass: (None)

The SAP objects are used by the SAP import interface.

9.3.27 @20 > C90 Objects for PipeSpec Designer

Comprehensive catalog with base objects for FEED, Viper, and pipe specs.

The following settings apply here:

Class: Dataset

Subclass: (None)

9.3.28 @20 > D10 Axis definition

Definition of the axes and the north arrow in isometrics.

The following settings apply here:

Class: Dataset

Subclass: (None)

9.3.29 @20 > D20 Base objects from external CAE data imports

This is where base objects originating from imports can be stored.

The following settings apply here:

Class: Dataset

Subclass: (None)

- D20 > Y30 > DGN
- D20 > Y30 > DXF
- D20 > Y30 > XMpLant

See also

Interface objects for base data import (dynamic objects) (Page 68)

9.3.30 @20 > D30 PDMS objects

This is where base objects which are required in PDMS can be stored.

The following settings apply here:

Class: Dataset

Subclass: (None)

9.3.31 @20 > D40 Reserved object names

Base objects

Objects with fixed names are created below the "@20 > D40 Reserved object names" node.

The "D40 Reserved object names" node is divided into the following areas:

- Y00 Reserved object names from COMOS
Objects in the COMOS iDB which require a fixed name
- Z10 Reserved object names of the customer
Customer-specific objects which require a fixed name

Below these, the objects are created in groups of 500; and below these, in groups of 25 for greater clarity.

Assigning names

The name of an object with a reserved name is composed as follows:

- Name of the area: "Y00" or "Z10"
- Letter "R"
- Five-digit consecutive number

Example: Y00R00001

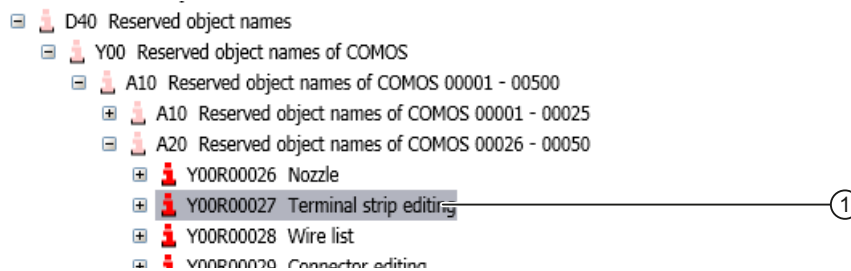
The name is assigned as part of the process for requesting reserved objects.

Name template

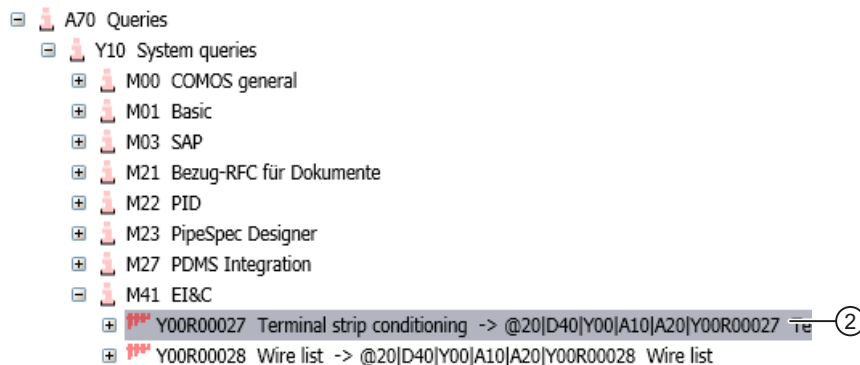
Base objects with a fixed name have a name template:

Using the objects

Using the "Terminal strip editing" object as an example:



Object (1) is used in module M41 as a query:



Tab with description

Each object with a reserved name has a tab on which there is a description for the application purpose.

The inheritance mode of this tab is set to "inactive" so that the description is only visible on the main object.

Buttons with reserved names

If you have to access the name of a button, create a button with a reserved name.

You can create attributes below a button with a reserved name.

See also section Attributes with the "Button" display type (Page 129).

9.3.32 @20 > Z10 Customer

All base objects for customer-specific objects are stored here.

"@30 Structures": Editing module-specific structures

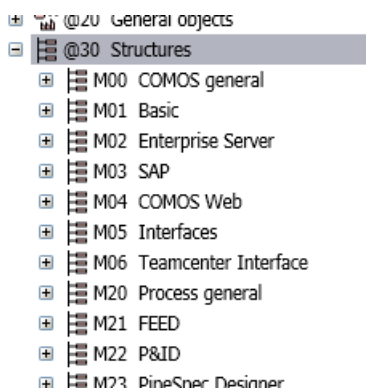
10.1 Purpose of the "@30" node

This node includes the standard-specific structures as they are used in the engineering objects. Whereas the "@10" and "@20" nodes contain the definitions of the base objects themselves, the "@30" node defines the relations between these objects as specified in the various standards. This means this node is responsible for the structures of the right mouse button, project presettings, labeling systems, position catalogs according to EN or ANSI, etc. for example.

There are no direct base object pointers to the objects in the "@10" and "@20" nodes from the engineering data. The objects are used as base objects in the "@30" node instead and the objects in the "@30" node have base object references to objects in the "@10" and "@20" nodes.

To create these structures, the required "subobjects" are defined as elements of the "main object" to enable the "subobjects" to be created below the "main object" using the right mouse button. Note that these relations must be defined in the "@30" node. This means it is necessary to use objects from the "@10" and "@20" nodes in the "@30" node and to create the structures with the objects in "@30".

The individual nodes for the base objects specific to the COMOS module are located below the "@30" node:



The structures that span all COMOS modules are saved in subnode M00. Nodes that start with a Z should be used for creating customer-specific structures. Existing structures can be expanded specific to a customer by supplementing the suffix "_Z".

See also

Basic rules in the "@30" node (Page 88)

10.2 Basic rules in the "@30" node

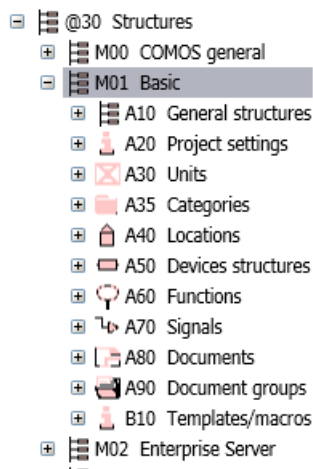
Requirement

- You are familiar with the purpose of the "@30" node.
See chapter Purpose of the "@30" node (Page 87).

Basic structure rules of the "@30" node

The "@30" node is structured according to the following rules:

- First level: Module-specific sorting
See chapter Overview of module-oriented first level (Page 94).
- Second level: There is always the same, topic-oriented basic structure in Mxx branches:
 - A10 General structures
 - A20 Project settings
 - A30 Units
 - A40 Locations
 - A50 Device structures
 - A60 Functions
 - A70 Signals
 - A80 Documents
 - A90 Document groups
 - B10 Templates/macros



See also chapter Overview of the structure of @30 below the modules (Page 94).

- The @30 structure is constructed below the topic-oriented basic structure using cross-inheritance.

- These structures are stored in corresponding nodes for the respective standard or topic structure. See the following block "Cross-inheritance" for more.
- Structure objects (that were not created for engineering but are only used for structuring) may be changed in the "@30" node. See the following block "Permitted customized settings in the "@30" node".

Cross-inheritance interrupts the hierarchical inheritance in the "@30" node.

The topic-oriented basis structure in the "@30" node is constructed of objects of the "@10", "@20" and "@50" nodes. All objects (except for structure objects) must have a base object pointer to an object in the "@10", "@20" or "@50" nodes or must be derived from them using hierarchical inheritance. See chapter Device structure A50 below the modules (Page 102).

Because the objects in the "@30" node have a reference to objects in the "@10" node, the hierarchical inheritance is interrupted at this point. This means it is not possible to prepare the manufacturer devices in the "@30" node at an upper node.

Instead, the following applies to each base object with a reference to an @10 base object: The standard-specific elements and/or attributes must always be built into the @30 structure using a reference. See also chapter Notes on multiple use of "@10" objects (Page 65).

To administer the cross-inheritance in the "@30" node, you can use the "@30 Tool". See also chapter "B10 Tooling > @30 Tool": Administering cross-inheritances (Page 159).

Permitted customized settings in the "@30" node

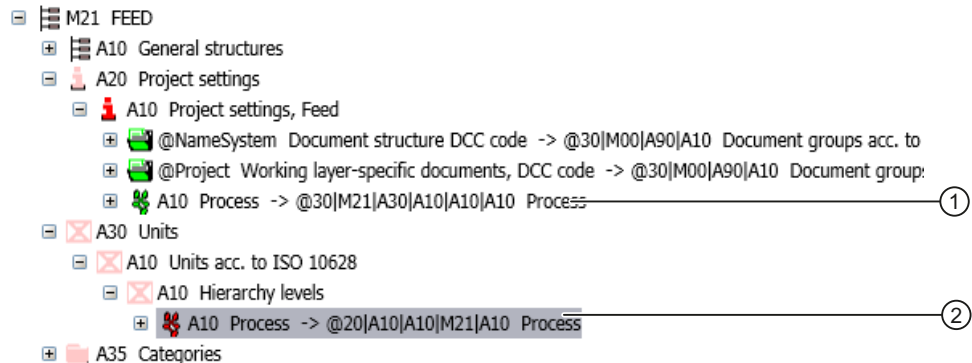
- The following customized settings can be made at objects in the "@30" node:
 - Name
See chapter Entering norm-specific text masks (Page 32).
 - Description
 - Label
 - Name template
 - Adding tabs and overbooking tabs
 - "Configuration" tab (e.g. engineering task)
 - Checking in icons
 - Creation mode
 - Object behavior
 - CDevice scripts
 - Customer-specific tabs can also be added in subsequent customizing
 - User classification (hierarchical & functional)
- The following properties cannot be customized:
 - Class / subclass
 - "Classification" subject card (hierarchical & functional)
 - Adding tabs and overbooking tabs at objects in the "@20" node
- Icons may be customized at structure objects

10.3 Specifying project structures as project setting

Purpose of the "Project settings" node

The construction of structures that are required within projects takes place below the following node in the individual modules:

- Project settings
All structures required for the COMOS module are assembled and thus documented by means of referenced objects below this node. The sources for these structures can be from the own module or from external modules. In the following example, you can recognize that object 2 is used in the project setting for FEED:



Differentiation of the different "Project settings" nodes

Module structures are stored in two locations:

- In the Mxx node module as module documentation
- In the "@30 > M00 > A20 Project settings" node as a presetting for projects

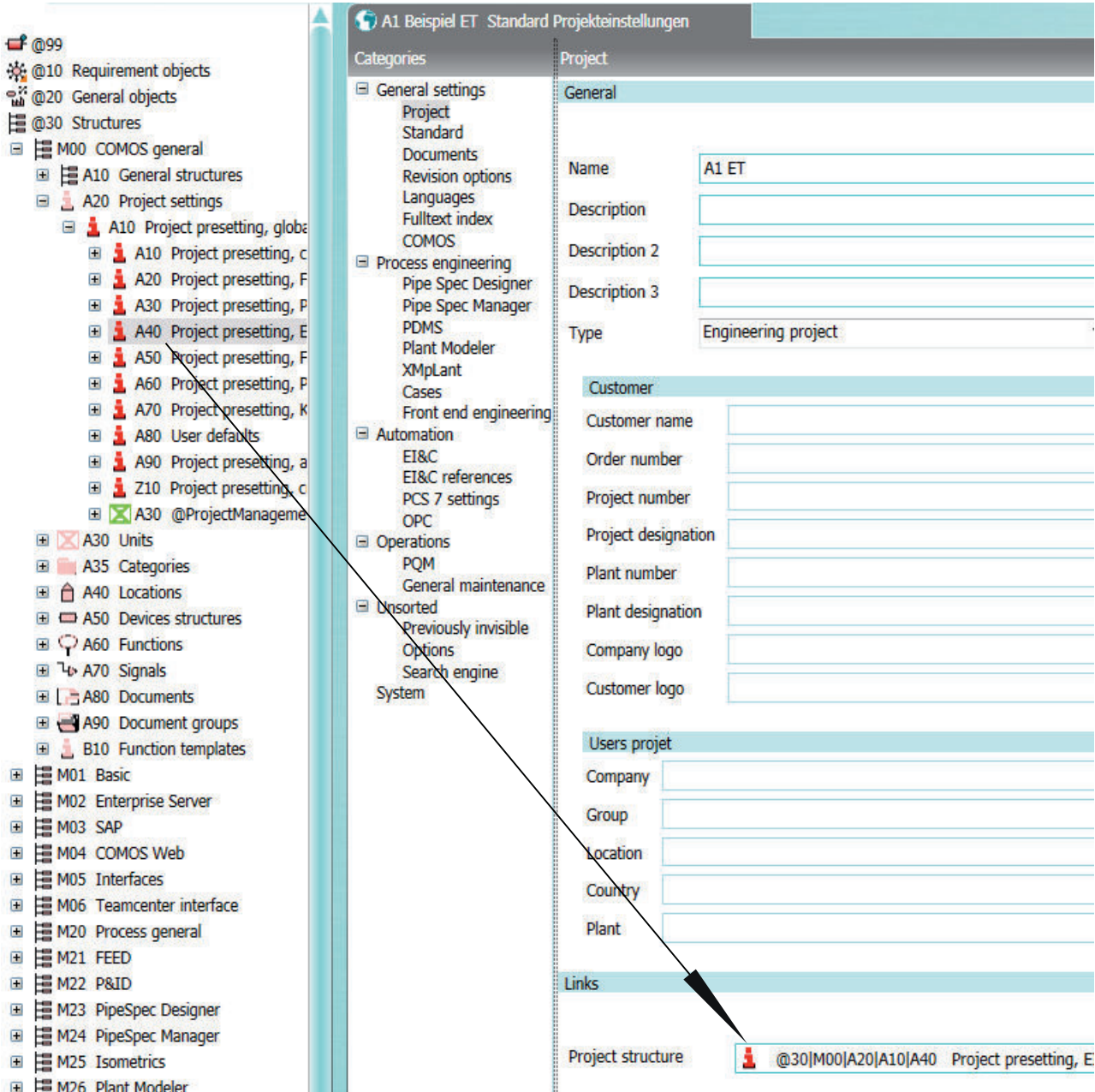
The following figure shows the project structure for FEED, once below the "@30 > M00 > A20 Project settings" node and once in the "@30 > M21 > A20 Project settings" node:

- [-] @30 Structures
 - [-] M00 COMOS general
 - [+] A10 General structures
 - [-] A20 Project settings
 - [-] A10 Project presetting, global
 - [+] A10 Project presetting, common example
 - [+] A20 Project presetting, FEED example
 - [+] A30 Project presetting, P&ID example
 - [+] A40 Project presetting, EI&C example
 - [+] A50 Project presetting, Fluidics example
 - [+] A60 Project presetting, PCS7 AdvES example
 - [+] A70 Project presetting, KKS example
 - [+] A80 User defaults
 - [+] A90 Project presetting, automation example
 - [+] Z10 Project presetting, customer specific
 - [+] A30 @ProjectManagement Workflows in project
 - [+] A30 Units
 - [+] A35 Categories
 - [+] A40 Locations
 - [+] A50 Devices structures
 - [+] A60 Functions
 - [+] A70 Signals
 - [+] A80 Documents
 - [+] A90 Document groups
 - [+] B10 Function templates
 - [+] M01 Basic
 - [+] M02 Enterprise Server
 - [+] M03 SAP
 - [+] M04 COMOS Web
 - [+] M05 Interfaces
 - [+] M06 Teamcenter interface
 - [+] M20 Process general
 - [-] M21 FEED
 - [+] A10 General structures
 - [-] A20 Project settings
 - [+] A10 Project settings, Feed
 - [+] A30 Units
 - [+] A35 Categories

Cross-module structures are developed in the module group or within M00. If a module does not need its own structure, the higher-level structure is used.

Purpose of the "@30 > M00 > A20 Project settings" node

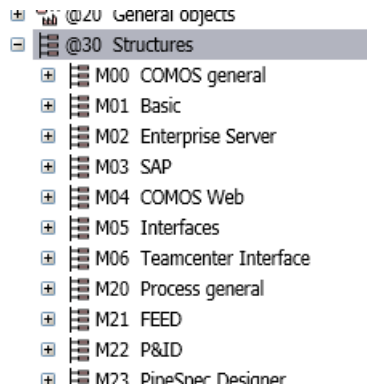
The "M00 > A20 Project settings" node has a special meaning. The structures are not only stored as documentation at this location, this node is also evaluated by COMOS in the "Project settings" field. Only the structures stored here are evaluated as project settings. The following figure shows that an object from "@30 > M00 > A20 Project settings" node is set as project setting for the "A1 ET" project:



10.4 Overview of module-oriented first level

Overview

The "@30" node is sorted module-specifically:



The structures are then constructed in the respective category below the corresponding module within a standard node or topic node. See section Overview of the structure of @30 below the modules (Page 94) for more on this. The standard nodes or topic nodes can then be added as elements below "@30 > M00 > A20 Project settings". This means separate structures can be made available for different modules.

The structures that span all COMOS modules are saved in subnode M00. These are structures with properties from different modules. For structures that consist solely of components from one module or module group, these structures are stored at the corresponding module location. If these structures are required in other modules, they are linked to the respective module.

The storage of the structures in the respective module or module group depends on the origin of the used structure components. If a structure includes components that span modules, it is saved in M00.

If a module does not need separate structures, they remain empty. If a module does not need its own structure, the used structure is linked from another module to the project setting.

Nodes that start with a Z are used for creating customer-specific structures. Existing structures can be expanded specific to a customer by supplementing the suffix "_Z".

Below the COMOS module-specific nodes, the structures are sorted by topic.

The base objects used are cross-inherited from the @10, @20 and @50 nodes.

10.5 Overview of the structure of @30 below the modules

The structures are divided into their respective topic below the module nodes.

A10 General structures

Here, you store general structures.

A20 Project settings

The structures for projects of the relevant COMOS module are stored here.

See section Specifying project structures as project setting (Page 91).

A30 Units

The structures for setting up the engineering in the "Units" tab are stored here. Unit structures are an important part of the project structure. You can find an example on the structure of a unit structure in section Designing project structures (Page 96).

A35 Categories

The category technique collects engineering objects in folders. For this purpose, the engineering objects are automatically moved to the category folders after they are created.

See section Categories A35 below the modules (Page 100).

A40 Locations

The structures for the unit world are constructed at this location. This takes place analogous to the unit structure.

A50 Device structures

The structures for devices and accessories are stored here.

See section Device structure A50 below the modules (Page 102).

A60 Functions

The structures for the functions are stored here.

A70 Signals

The structures for signals are stored here.

A80 Documents

The structures for documents are stored here (see "@30 > M00 > A80 > A10 Document base objects acc. IEC 61355 (Page 199)" and "Managing documents without IEC 61355 labels (@30 > xxx > A80) (Page 201)").

A90 Document groups

The structures for document groups are stored here (see "@30 > M00 > A90 Document groups (Page 200)").

B10 Templates/macros

The structures for templates or macros are stored here.

10.6 Designing project structures

Overview of the project structures

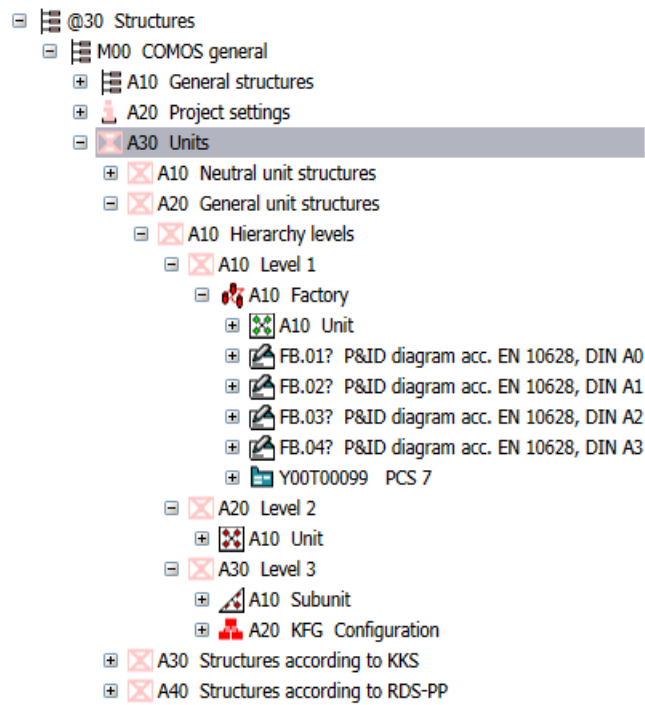
The project structures are designed in the module nodes (M00 and the Mxx nodes). The structures are divided and defined in hierarchy levels on a lower level below the module nodes:

- A10 Hierarchy levels
 - A10 Level 1
 - A20 Level 2
 - A30 Level 3
 - ...

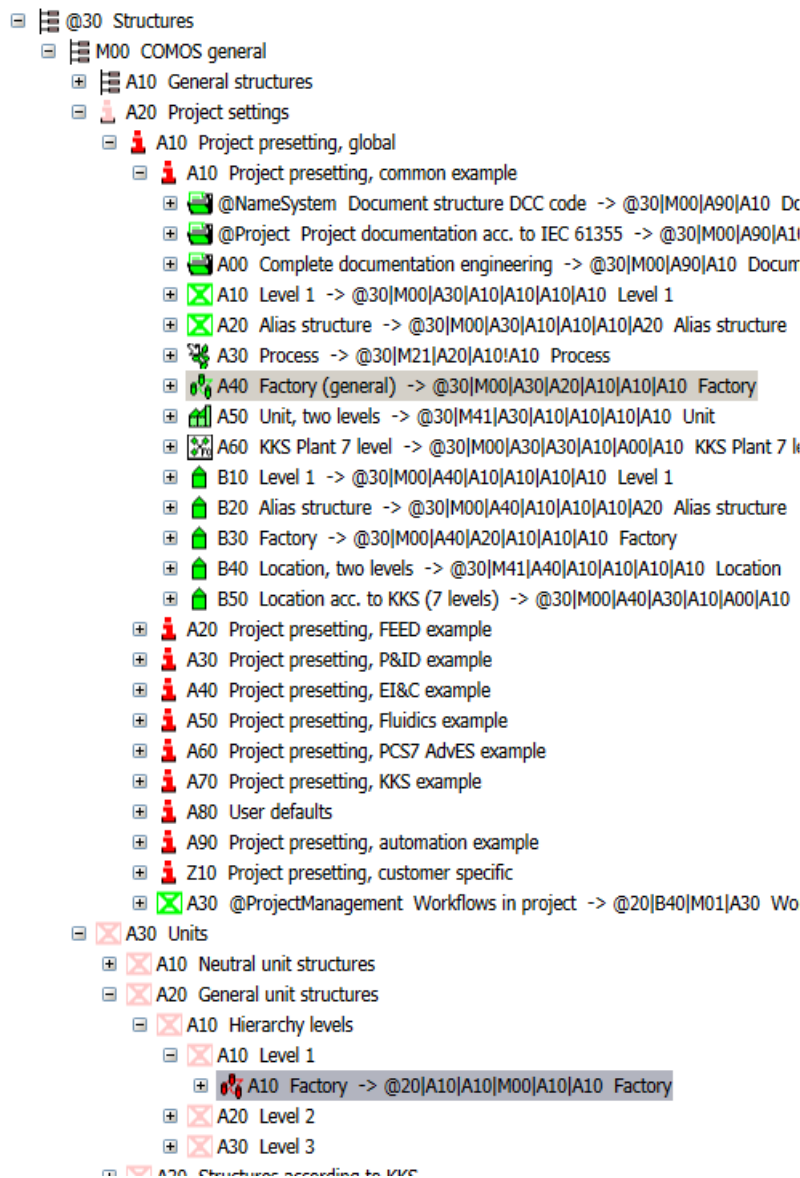
If there is only one level, the "A10 Level 1" node can be missing. The corresponding level is constructed from objects of the "@10", "@20", "@30" or "@50" nodes below each hierarchy level. This also includes that the respective subordinate hierarchy level is added as element. This means the different objects are combined into a structure and the complete structure is located on the first level.

The procedure is presented below based on a structure below "A30 Units". There are the following nodes below A30:

1. A node for the standard system
2. A collective node for the hierarchy levels below this node
3. Optional: One node each for the level below this collective node
4. A node for the merged structures below this node. The base objects in the "Elements" tab are used to merge the structures.



These structures can be added under "@30 > M00 > A20 Project settings":



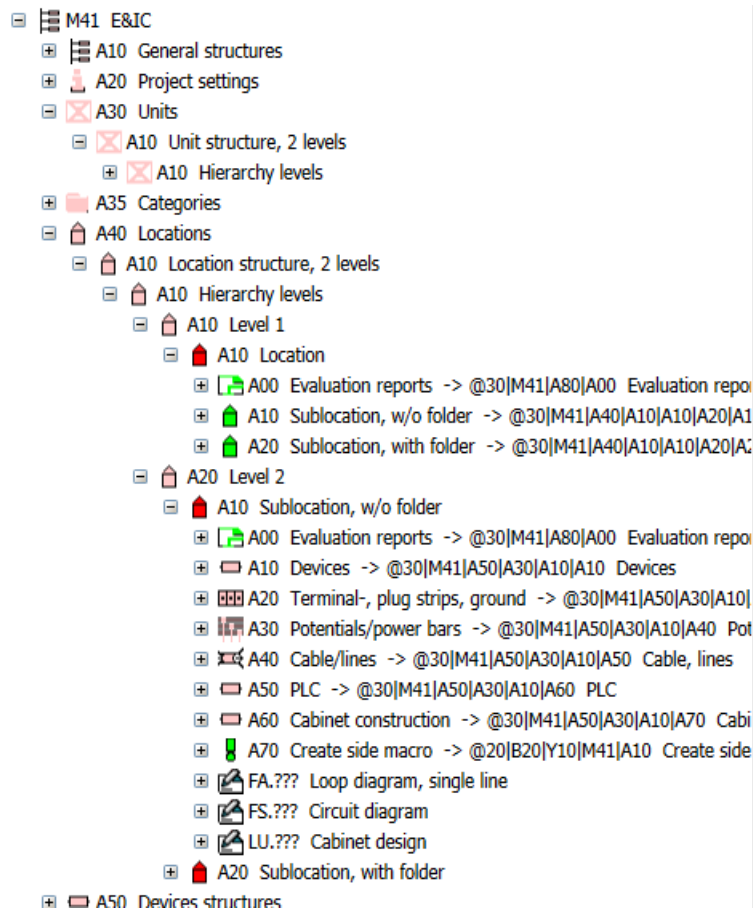
Ban on dereferencing

The "Dereference" object must be deactivated for all objects in the @30 branch, because the objects from the @10 branch would otherwise be used.

See chapter Controlling base object links using dereferencing (Page 46).

Example of a project structure below the "A40 Locations" node

Depending on the module, there can be project structures below the node "A30 Units" as an independent project structure deviating from the structure in A30 below the "A40 Locations" node:



See also

Example for unit structure A30 (Page 99)

10.7 Example for unit structure A30

Requirement

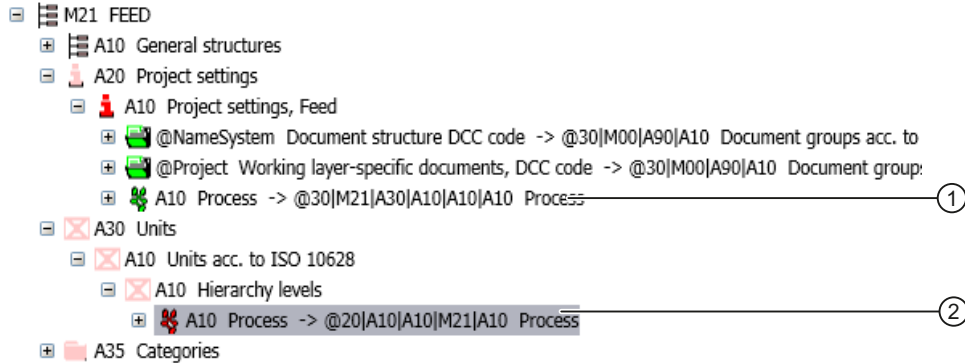
- You are familiar with the design of the project structure. See section Designing project structures (Page 96).

Example: Module node M21

In the following figure, the base object (2) is created according to the rules for designing a project structure.

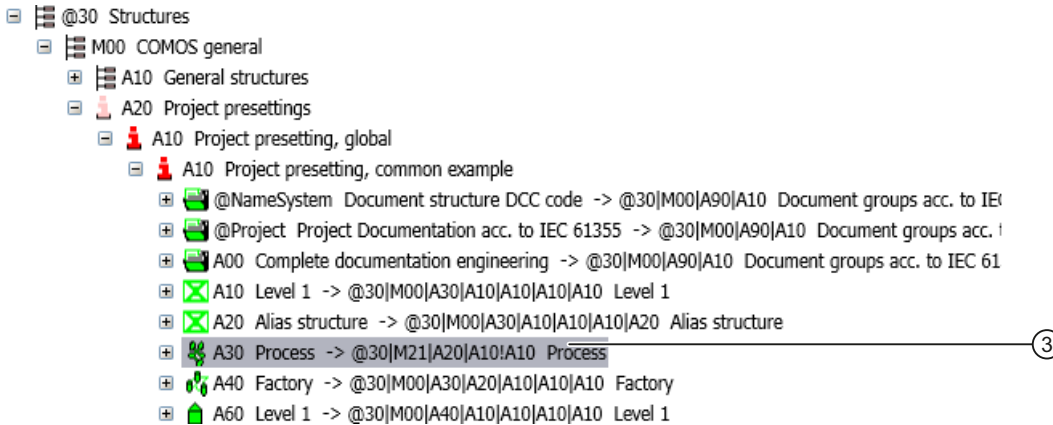
The standard system in this example is in the style of "ISO 10628".

The optional level is missing in this example.



The base object (2) is cross-inherited to base object (1) using a base object link.

The base object (1) is cross-inherited to base object (3) under "@30 > M00 > A20" using a base object link:



10.8 Categories A35 below the modules

Aim

The category technique collects engineering objects in folders. For this purpose, the engineering objects are automatically moved to the category folders after they are created.

Basic sequence of category technique

When a new engineering object is created, the following applies:

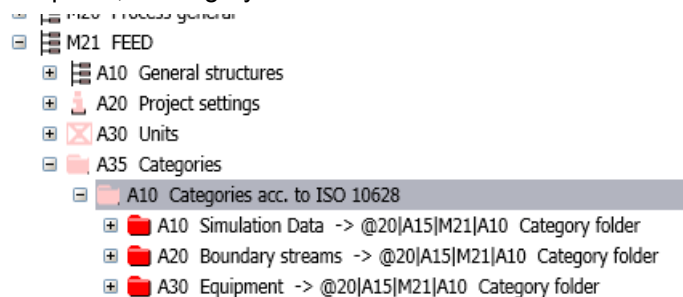
- The category folders below the point at which the new engineering object is created are searched one after the other in the Navigator.
- When a suitable category folder is found (in which the engineering object is designated as an element), the engineering object is moved into this folder. Engineering objects which already exist are not moved.
- Objects which are parallel to the document can also be sorted into categories by using the script option in the "SortNewObjectsInCategories" report template. If the document itself is below a category, it is sorted even without this option.
- If no suitable category folder is found, then the newly created engineering object remains in the position at which it was created.

Basic technical structure of the "Categories" function

1. Base objects of the "Unit" or "Location" class
 - "Subclass": "Category"
 - "Creation mode": Extend subelements.
2. The base object from list point (1) has other base objects on the "Elements" tab. These base object elements are the objects which are collected in the category in the engineering view.
3. On the engineering side, the categories are created as engineering objects below a unit or a location.

Base data structure of the categories in the iDB

- Generally applicable categories are constructed in the M00 branch.
 - The category folders are derived from "@10 > A20".
 - The structure objects inserted into the categories are derived from the module-specific branches in "@30 > Mxx".
 - The base objects inserted into the structure objects are derived from "@10".
- If required, a category structure can also be constructed in an Mxx branch. Example:



- The categories can be inserted for the "A10 Hierarchy levels" objects. In this way, the categories are automatically created when the units or locations are constructed. Example: "@30 > M21 > A30 > A10 Hierarchy levels"

10.9 Device structure A50 below the modules

Overview

The structures for devices and accessories are stored standard-specific or devices-specific in the "A50" node.

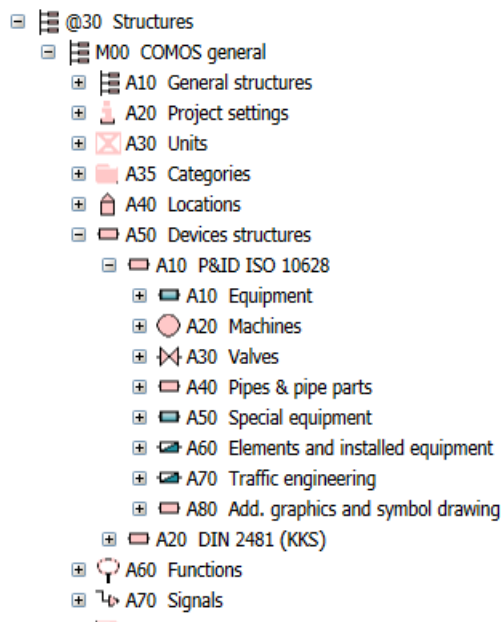
The nodes for a corresponding standard system or for a corresponding topic (devices-specific structures) are located below the "A50 Device structures" node:

- Standard-specific structure
- Devices-specific structure

The base objects originate from the "@10" node and are used standard-specific or topic-specific here.

Standard-specific structure

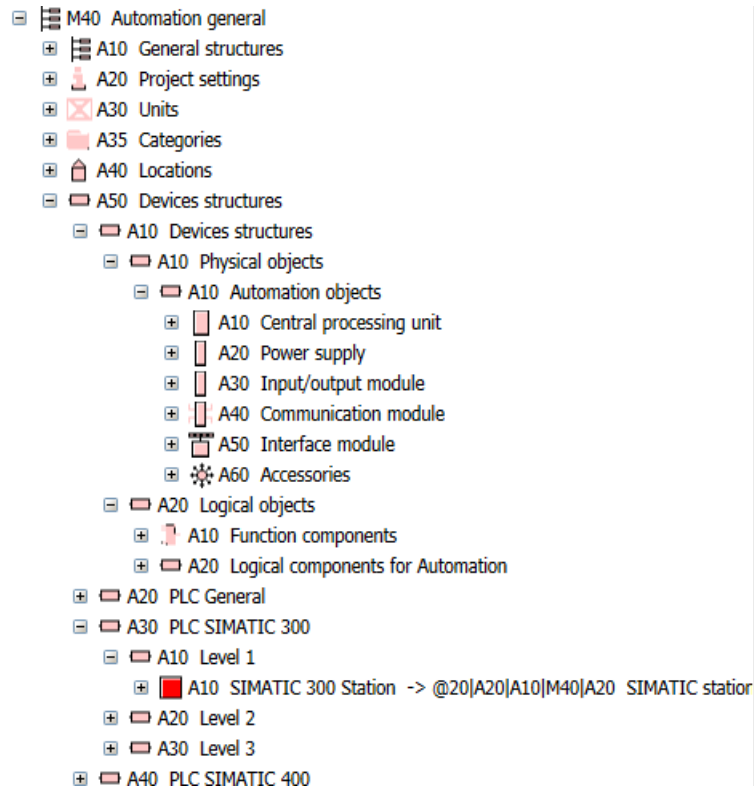
The following is a devices-specific structure using the example "@30 > M00 > A50":



See section Example for use of the "@10" node (Page 64).

Devices-specific structure

The following is a devices-specific structure using the example "@30 > M40 > A50":



For a distinction between cross-module devices and module-specific devices, see chapter Overview of module-oriented first level (Page 94).

Interruption of hierarchical inheritance in the "@30" node

Because the objects in the "@30" node have a reference to objects in the "@10" node, the hierarchical inheritance is interrupted at this point. This means it is not possible to prepare the manufacturer devices in the "@30" node at an upper node. See chapter Basic rules in the "@30" node (Page 88), keyword "Cross-inheritance".

10.10 Document structure A90 below the modules

Overview

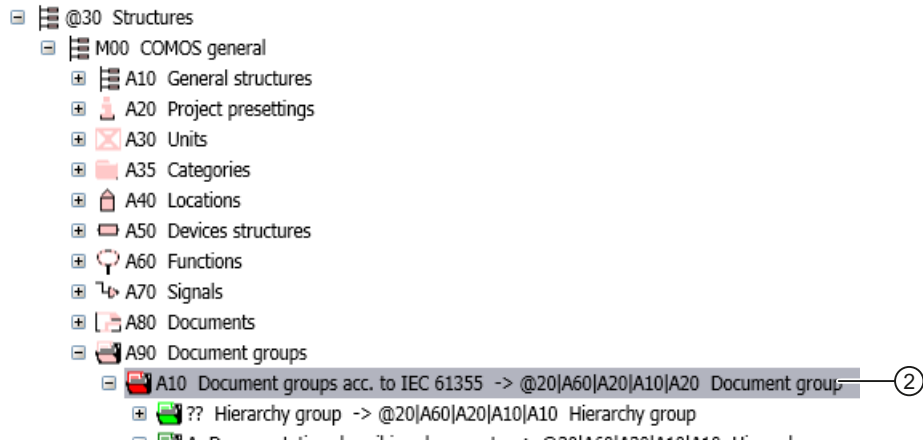
The structures for the document groups are stored here. The document groups are located in the M00 node as well as in the Mxx nodes. See section Specifying project structures as project setting (Page 91) for more on this.

Using document groups in the M00 node and the Mxx nodes

In the following example, the base object (1) is inherited from the M00 branch to the Mxx branch and is entered there as a base object reference for base object (2):



- Document structure according to IEC 61355
The document structure generally used according to IEC 61355 is stored in M00:



- Context menu for the project object

Using project structures as project setting

The distinction between documentary structures and structures that serve as project setting is explained in section Specifying project structures as project setting (Page 91).

See also

@30 > M00 > A90 Document groups (Page 200)

10.11 Information about labeling systems in the M00 branch

Labeling systems in @30 > M00

Because tagging systems are structural information, they are stored in node @30 Structures. The labeling systems are integrated into the normal subnodes in node "M00". There are no central root nodes below which all components of a labeling system would be available.

See Chapter Administration of the labeling system and ALIAS labeling system (Page 173).

"@40 @Y Attribute catalog": Editing attributes and tabs

11.1 Overview of the administration of attributes and tabs

Requirement

- You are familiar with the overview of the hierarchy of base data.
See chapter Standardized designations for folder objects (Page 21).

Step-wise definition of the attribute properties

An attribute is only created once in the attribute catalog. If you require attributes with the same content but with different display types, these attributes are derived from the same catalog attribute.

Basic properties such as "Name" and "Description" are entered in the catalog attribute. You set other properties such as the "Display type" for the derived attribute.

See chapter Uniqueness principle of attributes (Page 108).

See chapter Overview of the properties of catalog attributes (Page 111).

Storage location and name of catalog attributes and catalog tabs

The catalog attributes and catalog tabs are stored as follows:

- "@40 @Y Attributes and tabs"

See chapters Editing "@40 > A10 Attribute catalog" (Page 110) and Editing "@40 > A20 Tabs" (Page 119).

Placement of catalog attributes on the catalog tabs (derived attributes)

The catalog attributes are placed on the catalog tabs. The attributes are now called derived attributes.

- See chapter Placing catalog attributes on tabs (derived attributes) (Page 124).

Catalog attributes and tabs can be placed multiple times, provided the name specifications are adhered to.

Use of the catalog tabs with base objects

The base objects use the catalog tabs. There are various special features:

- Sorting tabs for the base object
See chapter Change the sort sequence for the tabs (Page 123).
- Multiple placement of a tab
See chapter Placing a tab on an object multiple times (Page 132).

11.3 Uniqueness principle of attributes

Dynamically created attributes and tabs

See chapter Dynamically created attributes (Page 133) and chapter Dynamically created tabs (Page 134).

Attributes for third-party software

If third-party software (such as SAP or Teamcenter) places special requirements on the names of attributes, dedicated nodes for this are available in the attribute catalog and tab catalog. See chapter Overview of the nodes in "@40 > A10" (Page 110) and chapter Overview of the nodes in "@40 > A20" (Page 119).

Finding attributes

See chapter Finding attributes with "@40 @Y Attributes and tabs" (Page 18).

See chapter Using "Attribute descriptions DE/EN" query (Page 17).

11.2 Note on attribute import

A workflow is available in the iDB that can be used to generate attributes in bulk based on external specifications. The core element of the workflow is that the "Description" field of the newly created attributes is set and that the purpose of the attributes is defined with the description.

See chapter Importing attributes and tabs in bulk (Page 257).

11.3 Uniqueness principle of attributes

Requirement

- You are familiar with the overview of the administration of attributes and tabs. See chapter Overview of the administration of attributes and tabs (Page 107).

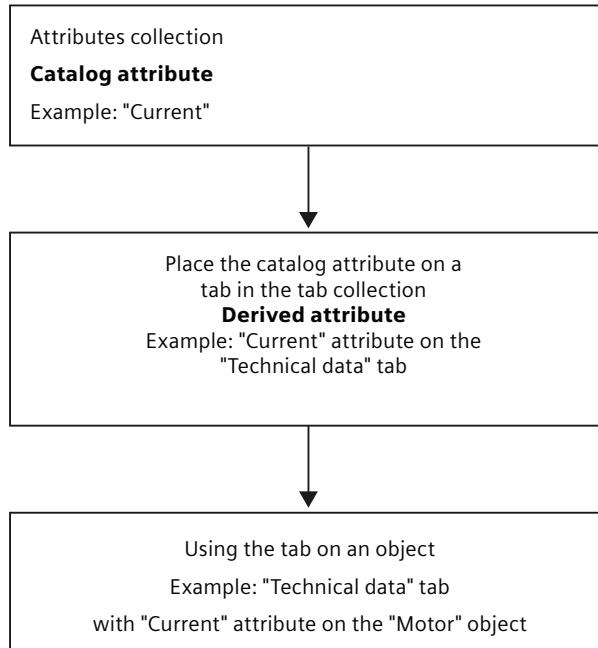
Definition of an iDB attribute

Attributes in the COMOS iDB sense are value-saving objects. Attributes are managed in the attribute catalog.

Buttons, frames, and descriptions save no values and are therefore handled separately. See chapter Attributes with the "Button" display type (Page 129) and chapter Attributes with the "Frame" or "Description" display type (Page 130).

Principle of uniqueness

An attribute is only created once in the attribute catalog. If you require attributes with the same content but with different display, these attributes are derived from the same catalog attribute.



The derived attributes can also be placed multiple times on a tab.

Changes to catalog attributes prohibited

Note

No changes to the catalog attributes in the attribute catalog

You may not change the catalog attributes in the attribute catalog.

See also

Overview of the properties of catalog attributes (Page 111)

11.4 Editing "@40 > A10 Attribute catalog"

11.4.1 Overview of the nodes in "@40 > A10"

Requirement

- You are familiar with the overview of the administration of attributes and tabs. See chapter Overview of the administration of attributes and tabs (Page 107).

Storage of attributes directly below the CDevice

Two methods are available in COMOS to prepare attributes in the base data:

- Standard case: CDevice > Tab > Attribute
- Special case: CDevice > Attribute
It is possible to create an attribute as child object of a CDevice in the COMOS object model. The attribute catalog uses this special case:
 - CDevice
 - In the Navigator, the attribute is located below the CDevice.
 - The attribute is not placed on any tab. You will not see the attribute when you open the properties of the CDevice. When customer-specific attributes are imported, the CDevice will not even have a tab for attributes.

Three main nodes in the attribute catalog

Root nodes for all catalog attributes:

- "@40 > A10 @Y Attribute catalog"
Generally applicable procedure for catalog attributes:
See chapter Overview of the properties of catalog attributes (Page 111).
See chapter "Attribute mapping table" tab (Page 112).

Below the root node, three different nodes are prepared:

- "Y00 @Y Attribute collection" (General catalog attributes)
 This area is used for managing the general catalog attributes for the delivered COMOS objects. This area is maintained by Siemens.
 See chapter Structuring variant: A05 Semantic structuring (Page 113).
 See chapter Basic rules for the "Description" field for catalog attributes (Page 116).
 The general catalog attributes can be locked to prevent changes by a "lock-lock" mechanism. This lock cannot be unlocked.
 See chapter "Object locking" and "System object" state (Page 37).
- "Y30 @Y Special attributes collection" (Special catalog attributes)
 This area contains attributes that deviate from the naming rules of the general catalog attributes. For example, some COMOS interfaces require attributes whose name is specified by the third-party software (e.g. SAP or Teamcenter). This area is maintained as an example by Siemens and can be supplemented by the customer.
 See chapter Structuring variant: A05 Semantic structuring (Page 113).
 See chapter Special attributes: Automation Interface (PCS 7) (Page 119).
- "Z10 @Y Customer attributes collection"
 This area is used for managing the customer catalog attributes and is maintained by the customer.
 See chapter Structuring variant: A05 Semantic structuring (Page 113).
 See chapter Customer-specific additions (Page 247).

11.4.2 Overview of the properties of catalog attributes

Requirement

- You are familiar with the overview of the nodes in "@40 > A10".
 See chapter Overview of the nodes in "@40 > A10" (Page 110).

Properties of the catalog attribute

The following table shows the properties that are provided when the catalog attribute is created:

Property	Description
"Name"	Unique character string See chapter Structuring variant: A05 Semantic structuring (Page 113).
"Description"	As precise as possible description of the contents. See chapter Basic rules for the "Description" field for catalog attributes (Page 116). Description may be adapted to the derived attribute.
"Display type"	Always: "Edit field".
"Type"	Always: "Alphanumeric".
"Inheritance mode"	Always: "Inactive".

You may make changes to all other properties only for the derived attribute on the tab.

Additional properties of the derived attribute

See chapter Overview of the properties of the derived attribute (Page 124).

11.4.3 "Attribute mapping table" tab

Requirement

- You are familiar with the overview of the nodes in "@40 > A10". See chapter Overview of the nodes in "@40 > A10" (Page 110).

Objective of the mapping table

- "Calculation"
 - Application 1: Link to an attribute that is also used for calculation of a value
 - Application 2: Link to an attribute to which a conversion is made
Example: Mapping of rated voltage onto nominal voltage. In this case, the rated voltage can be used for calculation in a script instead of the nominal voltage.
- "Customer mapping"
For free use.
- Mapping of a standard
On the Y00A10 tab, entry fields are prepared for mapping to important standards.
 - "eCl@ss"
 - "ETIM"
 - "ISO 15926"

Create further entry fields to support additional standards. You can store the "Identifiers" of the individual standard attributes that are to be mapped in the XML attributes. If a standard has several mapping options, enter the standard attribute.
- SIEMENS
Mapping to other Siemens attribute catalogs
- "COMOS classic"
Mapping to the attribute with identical contents in the cDB.

Calling the "Attribute mapping table" tab

"@40 > A10 > Y00A10 @Y Attribute mapping table"

This mapping table has an effect on the link to a product request. The attribute for a product request appears as follows in the iDB:

Attributes of the base object, "System data" tab: Attribute "Y00A04982 Link to product request"

This attribute for the product request is evaluated in the following cases:

- In the case of a change to the base object
The new base object can have another device request.
- For the "Return to request" function
In the case of an engineering object, this function sets the product request as the base object.

11.4.4 Structuring variant: A05 Semantic structuring

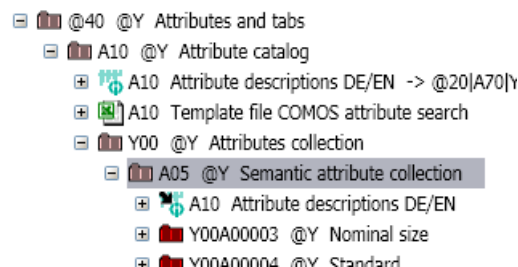
Requirement

- You are familiar with the overview of the nodes in "@40 > A10".
See chapter Overview of the nodes in "@40 > A10" (Page 110).

"Y00 @Y Attribute collection > A05"

Object structure:

- Collector node "A05 @Y Semantic attribute collection"
 - Below are folders for grouping according to German description (semantic folders)
Below that is a variable number of carrier objects. One carrier object each per catalog attribute.
Example:



"Semantic folder" means that attributes having an identical substring in the German description are grouped. Example:

"Performance level" (German: "Leistungsgrad") and "Efficiency level" (German: "Wirkungsgrad") are sorted into the shared "Y00A05246 Level" folder (German: "Grad"). Attributes without shared semantic folders are grouped in the "Y00A04928 Miscellaneous" folder.

- Structure of the carrier objects
 - The carrier object has the same name as the catalog attribute
 - Catalog attribute with checked-in name
You can place the catalog attributes on a tab. See chapter Placing catalog attributes on tabs (derived attributes) (Page 124).
 - "Y00A10 Attribute mapping table" tab for mapping of the attribute to standard systems
See chapter "Attribute mapping table" tab (Page 112).

Assigning names:

- Name of the area: "Y00" + letter ("A", etc.) + 5-digit consecutive number + Variant label ("AB", etc.) (if required)
- Maximum number of catalog attributes: 99999

Example: "Y00A00003 Nominal size", "Y00A00003AB Nominal size for bolt point item"

Outdated: Variant of a catalog attribute

Previous iDB versions had variant attributes that were identified using a variant suffix "AB", "AC", etc. in the name. Some attributes still have this variant suffix in their names. Example: "Y00A00417 Shaft power". This variant suffix is no longer relevant.

There continues to be variants for tabs.

11.4.5 Structuring variant: A10 Packaged structuring

Requirement

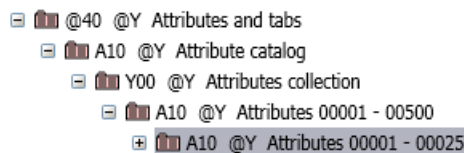
- You are familiar with the overview of the nodes in "@40 > A10". See chapter Overview of the nodes in "@40 > A10" (Page 110).

"Y00 @Y Attribute collection > A10"

Storage structure:

- Folder structure:
 - Folders with 500 entries
 - Below these, folders with 25 entries

Example:



- Structure below the last folder layer
 - A base object as carrier object for the catalog attribute
 - The carrier object has the same name as the catalog attribute
- Structure below the carrier object
 - Catalog attribute
 - Name of the catalog attribute
 - "Y00A10" tab, on which mapping information can be saved for the catalog attribute See chapter "Attribute mapping table" tab (Page 112). You can place the catalog attributes on a tab. See chapter Placing a catalog attribute on a tab (Page 124).

Assigning names:

- Name of the area: "Y00"
- Letter
Example: "A"
- Five-digit consecutive number
- Maximum number of catalog attributes: 99999

Example: Y00A00001

11.4.6 Structuring variant: Y30 Special attributes

Requirement

- You are familiar with the overview of the nodes in "@40 > A10".
See chapter Overview of the nodes in "@40 > A10" (Page 110).

"Y30 @Y Special attributes collection" (Special catalog attributes)

Object structure:

- Collector node based on module ("M03", etc.)
 - Below are folders for 25 carrier objects each
One carrier object each per special attribute
- Structure of the carrier objects
 - The carrier object has the same name as the catalog attribute
 - Catalog attribute with checked-in name
You can place the special catalog attributes on a tab. See chapter Placing catalog attributes on tabs (derived attributes) (Page 124).
 - "Y00A10 Attribute mapping table" tab for mapping of the attribute to standard systems
See chapter "Attribute mapping table" tab (Page 112).

Assigning names:

- Names are assigned by third-party software

Example: APPLICATION_ID

11.4.7 Structuring variant: Z10 Folder for customer sorting

Requirement

- You are familiar with the overview of the nodes in "@40 > A10".
See chapter Overview of the nodes in "@40 > A10" (Page 110).

"Z10 @Y Customer attributes collection"

Recommendation: Use the same rules as in the "Y00 @Y Attributes collection" area

Assigning names:

- Name of the area: "Z10" + letter ("A", etc.) + 5-digit consecutive number

Example: Z10A00001

11.4.8 Basic rules for the "Description" field for catalog attributes

Requirement

- You are familiar with the overview of the nodes in "@40 > A10".
See chapter Overview of the nodes in "@40 > A10" (Page 110).

General rules

You must observe three basic rules for describing a general catalog attribute:

- No object reference in the description
Examples of prohibited object descriptions: Pipe inside diameter, motor power, pump operating pressure, etc.
If you still need an object reference in the description, adapt the description of the derived attribute on the tab.
- Uniform writing
Consistent writing in order to avoid duplicates
- Geometry mapping is permitted
Examples: inside diameter, external diameter

See also

Supplementary rules for the "Description" field for catalog attributes (Page 117)

11.4.9 Supplementary rules for the "Description" field for catalog attributes

Requirement

- You are familiar with the general rules
See chapter Basic rules for the "Description" field for catalog attributes (Page 116).

General linguistic rules

- Abbreviations
 - Permitted: fixed specialist abbreviations
Abbreviations without a period that are often written entirely in upper case
Examples: CPU, XML, TÜV, GmbH
 - Allowed in exceptional cases
Part descriptions in brackets
 - Not allowed: Colloquial abbreviations
Examples: etc., e.g., max., min.

Incorrect	Correct
max. current	maximum current
e. g.	for example

- Abbreviations
 - Permitted: For a derived attribute on a tab
- Compounds
 - Allowed in exceptional cases
Suffixes
Prefixes

Writing of technical specification

- Electrotechnical properties
 - Permitted: Compounds
Descriptions of location, time and state are at the start of an attribute description
Example: inside, outside, input, output, start-up, shut-down, etc.
 - Example of permitted writing for electrical current
Operating current, nominal current, rated current, switching current, control current, input current, output current, breaking current, leakage current, maximum current
 - Example of permitted writing for electrical voltage
Operating voltage, nominal voltage, rated voltage, switching voltage, control voltage, auxiliary voltage, input voltage, output voltage, maximum voltage
 - Example of permitted writing for electrical power
Nominal power, rated power, switching power, input power, output power, reactive power, apparent power, runtime power, runtime reactive power, nominal apparent power, power consumption, power factor
 - Not allowed: "Voltage, input"
- Process-related properties
 - Permitted
Aggregate states
Medium (solid, liquid, light, fluid, powder, gas, etc.)
Media groups (oil, acid, air, water, etc.)
"Material"
Supplement/refinement/revaluation
 - Example of permitted writing for temperature
Operating temperature, design temperature, ambient temperature
 - Example of permitted writing for pressure
Operating pressure, design pressure, air pressure
 - Other permitted examples:
Molar enthalpy of heavy liquid
Liquid, cover side
Mass flow of the water upon discharge
Water hazard class at the outlet

Non-permitted combinations with process media

Example:

- Methane gas
Combination with:
- Temperature

In these cases you adapt the description of the derived attribute in the tab:

- "Temperature methane gas"

11.4.10 Special attributes: Automation Interface (PCS 7)

Requirement

- You are familiar with the overview of the nodes in "@40 > A10".
See chapter Overview of the nodes in "@40 > A10" (Page 110).

Overview

Special attributes with the CAX name from the transmission model are created for transmission of values using the Automation Interface.

These attributes are also made available for normal COMOS objects. This ensures that the PCS 7 data can be saved and transmitted in a way that is traceable within COMOS.

The PCS 7 attributes are prepared as follows:

- "@40 > A10 > Y30 > M40 > A10"

The attributes are collected in the "Y30T00001 Automation Standard" tab and made available under the general objects. Example:

- "@20 > A10 > A10 > M00 > A10 General unit"

11.5 Editing "@40 > A20 Tabs"

11.5.1 Overview of the nodes in "@40 > A20"

Requirement

- You are familiar with the overview of the administration of attributes and tabs.
See chapter Overview of the administration of attributes and tabs (Page 107).

Core statement

Root nodes for all tabs:

- "@40 > A20 @Y Tab catalog"
Generally applicable procedure for tabs:
See chapter Change the sort sequence for the tabs (Page 123).

Below the root node, three different nodes are prepared:

- "Y00 @Y Tab collection" (General tabs)
This area is used for managing the general catalog attributes for the delivered COMOS objects. The catalog attributes from the attribute catalog are placed on these tabs. This area is maintained by Siemens.
See chapter Storage structure and names of the tabs (Page 120).
- "Y30 @Y Special tab collection" (Special tabs)
This area contains tabs that deviate from the naming rules of the general tab catalog. For example, some COMOS interfaces require tabs whose name is specified by the third-party software (e.g. SAP or Teamcenter). This area is maintained as an example by Siemens and can be supplemented by the customer.
See chapter Storage structure and names of the tabs (Page 120).
- Customer tabs
"Z10 @Y Customer tab collection"
This area is used for managing the customer tabs and is maintained by the customer.
See chapter Storage structure and names of the tabs (Page 120).

Note

Cross-inheritance in the tab catalog

Not permitted: Cross-inheritance within the tab catalog.

Permitted: Inheritance from the "Y00 @Y Tab catalog" area in the "Z10 @Y Customer tab collection" customer area.

11.5.2 Storage structure and names of the tabs

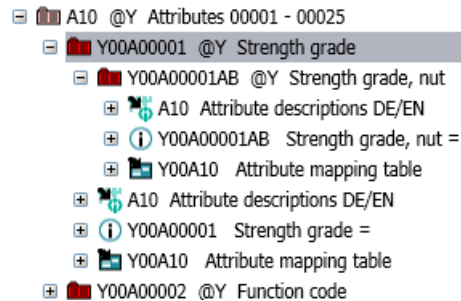
Requirement

- You are familiar with the overview of the nodes in "@40 > A20".
See chapter Overview of the nodes in "@40 > A20" (Page 119).

"Y00 @Y Tab catalog" (General catalog tabs)

Object structure:

- Collecting folders for 500 tabs each
 - Subfolders for 25 tabs each
 - Below that are one or more levels with folders for grouping according to German description.
 - Below that is a variable number of carrier objects.



- Structure of the carrier objects
 - A base object as carrier object of the tab.
 - The carrier object has the same name and the same description as the tab.
 - The tab must be empty at this position.
- Structure of the carrier object variants
 - The variants of the tab
 - A tab variant carrier object is created underneath the tab carrier object for every tab variant.
 - In turn the tab variant is located under this tab variant carrier object.
 - Tab Y00A20 for mapping of the tab to the COMOS modules.

Assigning names:

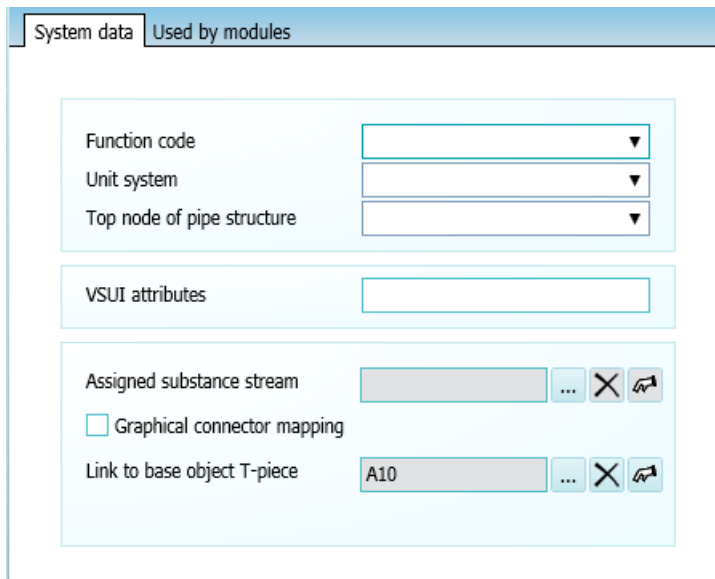
- Carrier object: Name of the area: "Y00" + letter "T" + 5-digit consecutive number
Example: Y00T00001
- Carrier object variants: Letter of the hierarchy level ("A", "B", etc.) + 2-digit consecutive number
Example: A01

Inheritance:

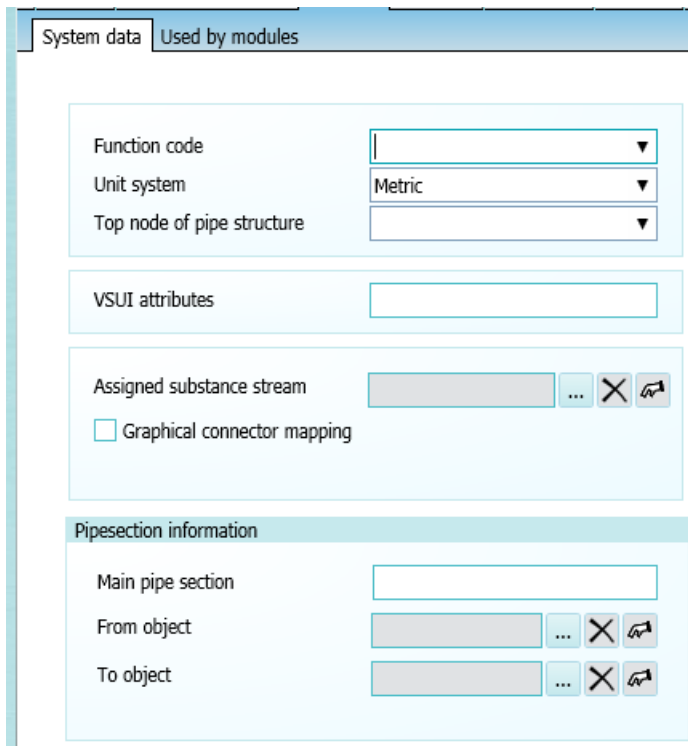
The content of the tab variants is inherited. You create generally used content at the top level
You add specific content at the next level.

Example:

"@40 > A20 > Y00 > A10 > A10 > Y00T00001 > A06 > B01 > C01":



"@40 > A20 > Y00 > A10 > A10 > Y00T00001 > A06 > B01 > C02":



The C02 object has the additional "Pipe section" option group on the "System data" tab.

Note that any changes to a tab variant affect all inheritance points.

"Y30 @Y Special tab collection" (Special tabs)

Object structure:

- "@40 > A20 > Y30 @Y Special tab collection":
 - Folders with 25 entries
- Structure below the last folder layer
 - A base object as carrier object for the tab.
 - The carrier object is always given the same name and same description as the tab itself.
- Structure of the carrier objects
 - Tab
 - Tab Y00A20 for mapping of the tab to the COMOS modules.

Assigning names:

- Names are assigned by third-party software

Example: CL_CRUSHING_MACHINE

"Z10 @Y Customer tab collection"

Recommendation: Use the same rules as in the "Y00 @Y Tab collection" area.

Assigning names:

- Name of the area: "Z10" + letter "T" + 5-digit consecutive number

Example: Z10T00001

11.5.3 Change the sort sequence for the tabs

Requirement

- You are familiar with the overview of the nodes in "@40 > A20".
See chapter Overview of the nodes in "@40 > A20" (Page 119).

Procedure

The tabs below the "Attributes" tab are sorted alphabetically by name from left to right.

You change the order of the tabs at the base object that uses the tab. Changing the order of the tabs does not impair check-in or inheritance.

The only exception is the "Y00T00001 System data" tab, which is always to be sorted in the far left-hand column. The sort text "Y00" is therefore preset for this tab in the tab catalog.

1. In the Navigator open the path to the object whose tab you wish to resort.
2. Open the properties of the tab which you wish to resort.

11.6 Placing catalog attributes on tabs (derived attributes)

3. In the "Sort text" input box enter a text which should be used for sorting the tab instead of its name.
Use the typical counting convention for the COMOS iDB as sort text: A10, A20, etc.
4. Confirm your entries.

11.6 Placing catalog attributes on tabs (derived attributes)

11.6.1 Placing a catalog attribute on a tab

Requirement

- You are familiar with the overview of the administration of attributes and tabs.
See chapter Overview of the administration of attributes and tabs (Page 107).

Procedure

1. Open the tab to which you wish to add a catalog attribute.
2. Select the "Design mode" command in the working area in the context menu of the tab.
COMOS switches the tab to Design mode.
3. Select the catalog attribute to be placed below the "@40 > A10 @Y Attribute catalog" node.
4. Use drag&drop to move the required catalog attribute from the Navigator to the tab.
5. Where necessary change the properties and the display for the catalog attribute.
More detailed information on the properties for an attribute can be found at Overview of the properties of the derived attribute (Page 124).

11.6.2 Overview of the properties of the derived attribute

Overview

For the properties that are specified for the catalog attribute, see section Overview of the properties of catalog attributes (Page 111).

The following table shows the properties which you stipulate at the derived attribute on the tab:

Property	Special feature
"Name"	Change is not permitted Exception: Counter suffix when used multiple times on a tab. See chapter Placing a catalog attribute on a tab multiple times (Page 131).
"Description"	Recommendation: Include additional information. Example: Inside diameter becomes pipe inside diameter Abbreviations are also permitted for the derived attribute. See chapter Basic rules for the "Description" field for catalog attributes (Page 116).
"Unit"	Currently, the reference unit is still used as the preferred unit, e.g. meter. In the future you will also be able to specify the preferred unit from the unit group here, e.g. millimeter.
"Display type"	The following display types are created directly on the tab and are not derived from catalog attributes from the attribute catalog: <ul style="list-style-type: none"> • Description (LA001...) • Frame (FR001...) • Button (BU001...)
"Type"	Permissible types: <ul style="list-style-type: none"> • "Text" Values that are translatable in engineering projects and that also have to be translated in these projects. • "Number" (numerical) Text box for numbers • "Alphanumeric" Input box for texts that are not being translated. Examples: standard tables, checkboxes • "Date" Date information • "Signature" Digital signature (eSign)
"Standard table"	Freely available
"Case variants"	Freely available
"Value"	You can provide a setpoint value for an attribute at the derived attribute on the tab or at the base object.
"Decimal positions"	Freely available
"Formatting digits"	Freely available
"Format"	Freely available
"Length"	Freely available
"Working area"	Freely available
"Edit mode"	Freely available
"Catalog attribute"	Here, at the derived attribute on the tab you set the associated catalog attribute from which the attribute is derived from the attribute catalog. This entry is set automatically depending on the procedure when placing the catalog attribute on the tab.
"Engineering object status"	Freely available

Property	Special feature
"Engineering object status value"	Freely available
"Inheritance mode"	Freely available
Scripts	Freely available
Info text	Freely available <ul style="list-style-type: none"> • The help text should take the context into account (tab and CDevice). Recommended help text for system attributes: Describe the type of software usage in multiple languages. • "Tooltip" field Recommended tooltip for system attributes: Enter a corresponding label when an attribute is being used on the software end.
Tooltip entry with system attributes	If an attribute is being used on the software end, identify this attribute as a system attribute on the "Help" tab. Furthermore you can describe the software usage at this point in multiple languages.

Editing derived attributes of base object (CDevice)

The properties of the derived attributes are defined in the tab catalog.

If the tab of a base object (CDevice) is used, the attributes may at most still be edited in the base object as follows:

1. Value
2. Preferred unit (°C or °F; m or km)
3. Mapping to other standards (e.g. eCl@ss)

11.6.3 General rules for the layout of tabs

Overview

Observe the following rules when creating tabs:

- Uniform configuration
- Group attributes that belong together and separate the groups by frames.
- The attributes in one column must have the same width as the description.
The description must be wide enough to be able to display all the languages used, and at least English and German.
- All units on a tab should be of the same width and all the units that can be used must be capable of being displayed in full.
- Plan for sufficient space for attribute values.
- The sizes and the horizontal positions of the attributes in a column should be identical.
- The distances between the attributes of a summary block should be identical.
- The distance between attributes and frame should always be the same.

- Tabs should be capable of being displayed in full on the screen.
Note: The resolution of COMOS in full screen mode is 1280x1024. The space for the Navigator still has to be subtracted from this maximum size.
- All attributes, including control elements such as check boxes and buttons, must have a description.
Permitted: Description of multiple grouped attributes using a common attribute of "Description" display type.
Example of a common description "Diameter inside/outside" for the "Inside diameter" and "Outside diameter" attributes.

Configuration examples

- Example 1:

The screenshot shows a 'Delivery data' form with the following fields and controls:

- Supplier: text input field
- Order number: text input field
- List price: text input field with a currency dropdown set to 'EUR'
- As of: text input field with a date value '00.10.0000'
- Packaging quantity: text input field with a unit dropdown set to 'pcs'
- Supplied length: text input field with a unit dropdown set to 'mm'
- Delivery time: text input field
- Delivery date: text input field
- Order relevant: checked checkbox
- Already ordered: unchecked checkbox

Four red boxes with circled numbers highlight specific UI elements:

- (1) Uniform alignment of texts (Supplier, Order number, List price, As of, Packaging quantity, Supplied length)
- (2) Uniform length of the text placeholder (Supplier, Order number, List price, As of, Packaging quantity, Supplied length)
- (3) Uniform length of the unit field (EUR, pcs, mm)
- (4) Uniform length of edit fields (Supplier, Order number, List price, As of, Packaging quantity, Supplied length)

- (1) Uniform alignment of texts
- (2) Uniform length of the text placeholder
- (3) Uniform length of the unit field
- (4) Uniform length of edit fields

- Example 2:

The screenshot shows an 'Available catalogs' table with the following structure:

Available catalogs			Activate
[Placeholder]	[...]	[X] [↩]	<input type="checkbox"/>
[Placeholder]	[...]	[X] [↩]	<input type="checkbox"/>
[Placeholder]	[...]	[X] [↩]	<input type="checkbox"/>
[Placeholder]	[...]	[X] [↩]	<input type="checkbox"/>
[Placeholder]	[...]	[X] [↩]	<input type="checkbox"/>

Two red boxes with circled numbers highlight specific UI elements:

- (5) Group titles for options of the same type (Available catalogs)
- (6) Individual labeling for identical checkboxes (Activate)

- (5) Group titles for options of the same type
- (6) Individual labeling for identical checkboxes

11.6.4 Attributes with the "List" display type (list specs)

Requirement

- You are familiar with the overview of properties of derived attributes. See chapter Overview of the properties of the derived attribute (Page 124).

Catalog attributes for list attributes

The following also applies to attributes with the "List" display type:

- One catalog attribute for the list attribute. Display type: "Edit field".
- One catalog attribute each for every row attribute or in the mirrored display for every column attribute. Display type: "Edit field".

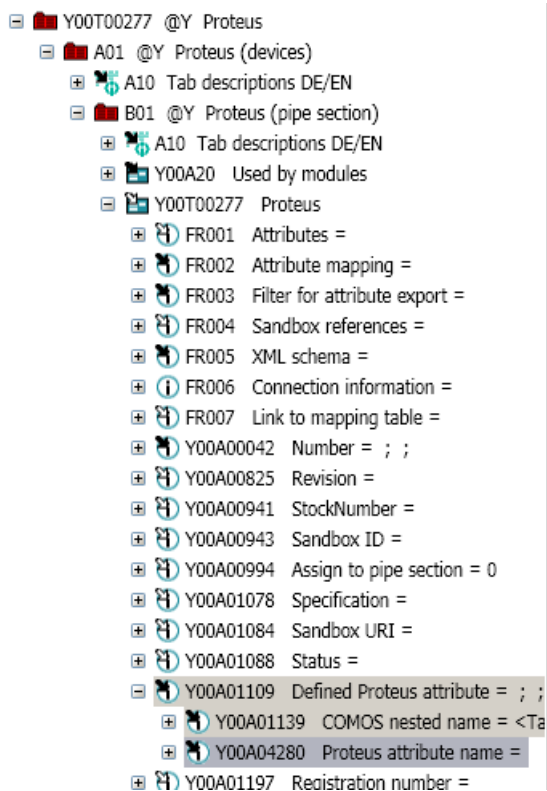
One derived attribute each for the list attribute and every row attribute. The derived attributes retain the name from the attribute catalog. The properties of the derived attributes are edited as described.

The following figure shows the procedure in principle for configuring lists:

Example:

"@40 > A20 > Y00 > A10 > B30 > Y00T00277 > A01 > B01 > Y00T00277 Proteus > Y00A01109"

Below the attribute Y00A01109 are the associated attributes Y00A01139 and Y00A04280:



All three attributes are created in @40 as catalog attributes:

- Y00A01109
Derived from "@40 > A10 > Y00 > A05 > Y00A00303 > Y00A01109 > Y00A01109"
- Y00A01139
Derived from "@40 > A10 > Y00 > A05 > Y00A01154 > Y00A01139 > Y00A01139"
- Y00A04280
Derived from "@40 > A10 > Y00 > A05 > Y00A01154 > Y00A04280 > Y00A04280"

Dynamic list attributes

You assign catalog attribute Y0003550 to dynamic list attributes as the base attribute.

Place the dynamic list attribute on a tab in the tab catalog. In the attribute properties of the placed list attribute, specify which catalog attributes are allowed as sub-attributes of the list. You must set the inheritance to "inactive" for the dynamic list attribute and the sub-attributes.

You may only create sub-attributes on the tab for an object if you have specified them on the tab in the tab catalog.

11.6.5 Attributes with the "Object query" display type (SUI query)

Requirement

- You are familiar with the overview of properties of derived attributes.
See chapter Overview of the properties of the derived attribute (Page 124).

Procedure

The following applies to attributes with the "Query" display type:

- The attributes are first created as catalog attributes with the display type "Edit field".

Attributes with the query attribute are configured at the derived query attribute on the tab.

11.6.6 Attributes with the "Button" display type

Requirement

- You are familiar with the overview of properties of derived attributes.
See chapter Overview of the properties of the derived attribute (Page 124).

Procedure

You do not create an attribute with the "Button" display type in the attribute catalog, but rather directly in the tab.

Buttons are given the name BU00n.

Example:

"@40 > A20 > Y00 > A10 > A10 > Y00T00003 > A01 > B15 > C01 > Y00T003 > BU001"

Editing of "Value" field not permitted

You cannot save any values at a button. If you need a value, in the tab you must place a separate attribute at which you can save values.

Previous attributes below buttons

Attributes which had previously been created below a button are placed parallel to the button. Visible attributes are placed in the tab within a frame in order to display the relationships.

Visibility of attributes in the Navigator and in the tab

See chapter Placing attributes invisibly (Page 131).

11.6.7 Attributes with the "Frame" or "Description" display type

Requirement

- You are familiar with the overview of properties of derived attributes.
See chapter Overview of the properties of the derived attribute (Page 124).

Impermissible actions

You do not create an attribute with the "Frame" or "Description" display type in the attribute catalog, but rather directly on the tab.

The following actions are impermissible for attributes with the "Frame" or "Description" display type:

- Setting a value
- Hard-coded access

11.6.8 Placing attributes invisibly

Visibility of attributes in the Navigator and in the tab

- Invisible attributes: "~" working area
If attributes in the engineering view are not to be visible either in the Navigator or in the tab, define this via the "~" working area (layers).
In the base project, the attributes are still visible for the administrator.
- No visibility in the tab (only for dynamic attributes)
In the case of dynamic attributes only, you can specify that these should be displayed only in the Navigator and not in the tab. To specify that dynamic attributes should only be displayed in the Navigator and not in the tab, set the ControlProperties to a zero string. Place these dynamic attributes on a tab inside a frame with an appropriate name, e.g. "Hidden system attributes for component lumping".
For all attributes, select the "Not editable (value set by script)" entry from the "Edit mode" list in the attribute properties.
- Identifying invisible attributes
Put a border around attributes that are to be invisible in engineering. In the description of the frame, specify that the attributes are to be invisible.
Example: "Invisible system attributes"
"@40 > A20 > Y00 > A10 > A10 > Y00T00001 > A02 > B01 > C01 > Y00T00001 > FR099
Invisible system attributes"

11.6.9 Placing a catalog attribute on a tab multiple times

Placing attributes multiple times using counter in the name

You can use multiple placement, for example, to create multiple attributes for user names on a tab.

The following rules apply to the allocation of names for placed attributes:

- Name of the first placed attribute: Name from the attribute catalog without any changes, e.g. "Y00A00001".
The first attribute placed is not given a counting suffix as placing multiple times is not always required and any subsequent name change causes information to be lost.
- Name of additional attributes: Name + Counter suffix "AAnn"
Start value: "AA02".

Example:

Name	Description
Y00A00001	User name 1
Y00A00001AA02	User name 2
Y00A00001AA03	User name 3

The count of the attributes in the attribute name does not have to be consecutive.

However, the count in the attribute name must agree with the count in the description.

Example: Multiple placement with gap in the count

Name of the attribute	Description of the attribute
Y00A00010	Temperature 1
Y00A00010AA04	Temperature 4

Example: Single placement with counter in the description

Name of the attribute	Description of the attribute
Y00A00025	Voltage 1

11.7 Placing a tab on an object multiple times

Requirement

- You are familiar with the overview of the administration of attributes and tabs. See chapter Overview of the administration of attributes and tabs (Page 107).

Placing a tab variant

You can place a tab on an object multiple times.

Since the name of any tab placed must be unique the following regulation applies to the name allocation when placing multiple times in this way:

For each tab placed multiple times for the object, you append the counting suffix "AAnn" additionally to the name, beginning with "AA01".

The following list shows an example for the names of the "Y00T00005" tab placed multiple times:

- Y00T00005AA01
- Y00T00005AA02
- Y00T00005AA03
- Y00T00005AA04

If the counting for the tags relates to counting objects such as connections for a device, the counting suffix for the tab must match the number counted for the connection.

Note

Gaps in the numbering are permitted for this.

The following table illustrates this assignment:

Description	Name
Tab for connection CX1	Y00T00005AA01
Tab for connection CX3	Y00T00005AA03
Tab for connection CX5	Y00T00005AA05

Placing different tab variants

You can place different variants for a tab at an object. The different tab variants all have the same name in the tab catalog. Since the name for a placed tab must be unique, you attach the counting suffix "AAnn" to the name for all placed tab variants. Use of a different counting suffix is not permitted.

The same regulations apply when using the counting suffix "AAnn" as with the placing of a tab at an object multiple times.

11.8 Dynamically created attributes

Requirement

- You are familiar with the overview of the administration of attributes and tabs. See chapter Overview of the administration of attributes and tabs (Page 107).

Definition

Dynamic attributes are attributes created during runtime automatically or using a script.

Rules

Dynamic attributes receive a base attribute from the attribute catalog by a user input. The following attributes are permissible as a base attribute:

- General catalog attribute
- Special catalog attribute (from "Y30 @Y Y00A04928 Special attributes collection")

Use the name of the catalog attribute for the dynamic attribute with single use. When a dynamic attribute is used multiple times, you determine the name based on the same rules as when counting catalog attributes.

Example: Using a catalog attribute multiple times

Name of catalog attribute XY	Name of dynamic attribute XY-DYN
Y00A00316	Y00A00316
	Y00A00316AA02

Name of catalog attribute XY	Name of dynamic attribute XY-DYN
	Y00A00316AA03
	Y00A00316AAnn

As an option you can also name the first derivation as "...AA01".

If you use the dynamic attributes for counted objects (e.g. Temperature 1, Temperature 4), the count according to the name and the count according to the description must yield the same order.

Script for creating dynamic attributes

The script for creating dynamic attributes can be found in the database at the following location:

```
"@20 > A80 > M00 > A10 > M00S00009"
```

You must not create dynamic attributes directly under an object. The following tab becomes owner of the dynamic attributes at runtime:

```
Y00T00009 "Dynamic attributes"
```

The dynamic attributes do not have to be placed on the tab. It is sufficient to set the tab as owner in the object model.

Dynamic attributes created during runtime

Dynamic attributes that are created at runtime and are thus unknown do not require a catalog attribute assigned by the user as a base attribute. Example: Dynamic attributes created when objects are imported.

See also

Interface objects for base data import (dynamic objects) (Page 68)

Dynamically created tabs (Page 134)

11.9 Dynamically created tabs

Requirement

- You are familiar with the overview of dynamically created attributes. See chapter Dynamically created attributes (Page 133).

Procedure

If tabs need to be created during runtime, you can implement this in exactly the same way as with the attributes with the tab count and a pointer in the tab catalog to the Y00T00009 tab.

You can create a variant of tab Y00T00009 in the tab catalog if this is required for an interface.

See also

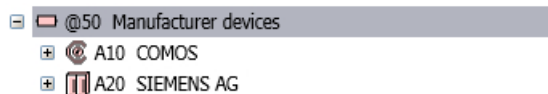
Interface objects for base data import (dynamic objects) (Page 68)

"@50 Manufacturer devices": Editing the manufacturer catalogs

12

12.1 Purpose of @50

The manufacturer catalogs are located under @50. The manufacturer catalogs are currently divided into two groups:



- A10 COMOS
Includes manufacturer catalogs for typical COMOS functions and workflows.
Example: Manufacturer devices for the COMOS PipeSpec Designer function.
- A20 SIEMENS AG
Includes Siemens manufacturer catalogs.
Example: Manufacturer devices for Siemens automation objects.

These catalogs include the base objects for the components which can be ordered such as devices (manufacturer devices) and accessories. For a base object to be considered a manufacturer device base object, a valid article number must exist in the "Order data" tab.

The branch @50 does not depend on a standard, which is why there are no standard-specific structures. Instead this branch @50 includes manufacturer-specific structures. All base objects from this node are derived from requirements from the @10 branch. This means that devices from different manufacturers can be derived from the same requirement from the @10 branch.

Creating base objects in @50

The branches in @50 are mostly empty on the layer of the manufacturer devices. You first have to create the manufacturer device base objects. You can find more information on this topic in the respective manuals:

- Creating base objects for pipe spec catalogs
The following information refers to unchanged project properties for the PipeSpec Designer. You can find more information on this topic in the "PipeSpec Designer" manual, keyword "Creating new catalog".
- Creating base objects for PIA
You can find additional information on this topic in the "EI&C Administration" manual, keyword "Administering manufacturer catalogs".
- Base objects for SAP manufacturer devices
You can find additional information on this topic in the "Platform Interfaces" manual, keyword "Importing manufacturer device catalog".
- Creating manufacturer devices for special components
Find more information on this topic in the "EI&C Operation" manual, keyword "Special components: creating manufacturer devices from requests".

12.2 Classification in the iDB system

Derivative of @10

The COMOS basic device objects for the manufacturer devices are defined in the @10 branch.

All base objects in the @50 branch are derived from requirements from the @10 branch. This means that devices from different manufacturers can be derived from the same requirement from the @10 branch.

Examples:

- COMOS base objects for automation objects
"@10 > A60 > A10 > A10 Automation objects"
 - Tabs are prepared in the COMOS base objects.
 - Simplified symbols are prepared in the COMOS base objects.
 - The COMOS base objects are identified with the "Request" option.
- COMOS base objects for logical automation objects
"@10 > A60 > A20 > A20 Logical automation components"

Permissible structures in "@50 manufacturer devices"

In the @50 branch, the COMOS basic device objects are expanded for the specific manufacturer device and grouped into a product catalog. The following options exist for the product catalog:

- General manufacturer devices plus specific manufacturer devices
 - Initial structure of a manufacturer-specific catalog with general device objects. The general device objects reference requirements in the @10 branch. An "A00 General devices" branch is created below the corresponding manufacturer node for this. Example: "@50 > A20 > A00 > General devices for product catalog".
 - Then, structure of the manufacturer-specific manufacturer device catalog with reference to the "A00 General devices" branch.
 - This procedure is recommended when requirements from the @10 branch are used multiple time in the manufacturer device catalog.
- Manufacturer-specific manufacturer device catalog with direct reference to the @10 branch. Device objects with direct reference to the @10 branch are to be changed only minimally in the @50 branch. For example, an additional tab is permitted.

Example:

"@50 > A20 > A10 > A20 > A10 > A10 > A20 > A10 > A40 ET 200M"

Permitted customized settings in the @50 branch

When a manufacturer-specific catalog with general device objects "A00 General devices" is used:

- All customized settings permitted
The inheritance from @10 is to be retained if possible

When the requirements in the @10 branch are directly referenced:

- Name, description, name template
- Entry of attributes on the tabs
- Expansions of the connectors

Symbols and classifications should be created preferentially in the @10 branch.

Use of manufacturer devices in the @30 branch

The following options exist for use of the manufacturer device catalog:

- Reference to a root node of a manufacturer device catalog
The manufacturer device catalog is completely taken over from the @50 branch
- Occasional references to individual manufacturer devices from the @50 branch

Example:

- "@30 > M40 > A50 > A60 ET 200M".
The references to the @50 branch are available on the last level of the object tree.

"@99 System settings": Editing default lists and settings

13.1 Reference: "@99 System"

"@99 > A10 Classification key"

See chapter Classifying objects (Page 183).

"@99 > A20 User interface"

Objects used to sort windows, tabs and menus in COMOS are sorted here.

Generally applicable user interfaces in "@99 > A20 > M00"

- "A10 Mapping tables"
User interfaces for linking attributes to other attributes or Excel.
- "A20 Q-Signatures"
This query displays the signatures of documents.
- "A30 Search/index object"
This is where you make the settings for search and index objects.
- "A40 'Extra' menu entries"
 - Node "@99 > A20 > M00 > A40 > A10 Query" creates the structure of the COMOS menu "Extra > Queries".
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Creating a query".
- "A50 Data import"
- "A60 'Documents' menu entries"
The objects collected in this node control some entries in the "Documents" menu. Example: Print manager.

"@99 > A30 System data"

Objects for system functions are stored here.

Generally applicable user interfaces in "@99 > A30 > M00"

- "A10 Read processors"
Various import interfaces.
- "A20 Document type mapping"
A list of objects that each represent the file extension of a document type. Depending on the document type, special import settings are available.
- "A30 ESign batch"
Base objects for controlling the electronic signatures of documents.

- "A40 Link objects"
The links specify standard objects. Example: A specification concerning which wire list should be used.
- "A50 Revision"
Controls the base objects of the revisions. The revisions are used to identify editing states of an object and make them verifiable.
See also chapter AUTOHOTSPOT.
- "A60 XML connector data"
Objects for administration of XML connectors.
 - "A10 Query pointer"
Carrier object for the "Y00T00034.Y00A00772 Query" attribute. The name of this attribute is hard-coded. If an XML connector detects an attribute of this name, the "Query pointer" functionality is activated.
- "A70 Working areas"
Base objects that represent the working areas.

"@99 > A40 Status"

See chapter Overview of status management A40 (Page 143).

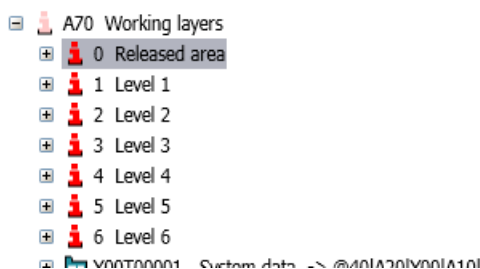
"@99 > A50 Profiles/user settings"

For each user there is an object in which the user settings are saved.

See chapter "A50 Profiles/user settings": Administration of personal settings (Page 153).

"@99 > A70 Working layers"

Objects used to control working layers are sorted here.



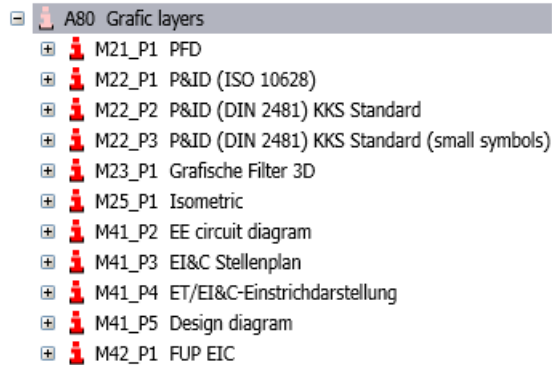
The attributes of the working layer base objects are entered with the "~" working area and are therefore invisible for everyone except the administrators.

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Administering working areas".

Keywords for the StandardDB: @WOLEVELS

"@99 > A80 Graphical layers"

Objects used to control graphical layers in interactive reports are sorted here:



You can find additional information on this topic in the "Operation" manual, keyword "Displaying/hiding or freezing content of a layer".

Keywords for the StandardDB: @GRAPHICS, @LAYERS

"@99 > B10 > A10 > @30 Tool"

This base object is a tool to administer cross-inheritance in the "@30" node. See chapter "B10 Tooling > @30 Tool": Administering cross-inheritances (Page 159).

13.2 Overview of status management A40

Overview

Status management is used to manage and display the state of objects from the point of view of engineering.

- Object-based status management must be set up first. Up to 21 base objects are created for this purpose.
Each base object manages a separate status. For example, a "consistency check" status can be set up and the user can switch between "consistent", "inconsistent", and other states for this status. The new status is written to the object, and the object thereby receives a new time stamp. See chapter "A40 Status": Object-based status management (Page 144).
- The object-based status can be expanded using queries. The status is no longer saved at the object but in a separate XML file. If the status is changed, only the query changes. The object is unchanged. See chapter "A40 Status": Query-based status management (Page 148).

You can specify for each status whether object-based or query-based status management is to be used:

- If the "Status" query is located below the status base object, the query-based status management is used.
- Otherwise, the object-based status management is used.

You can find additional information on the use of status management in the "Operation" manual, keyword "Managing object status".

13.3 "A40 Status": Object-based status management

13.3.1 Creating status base objects

Requirement

- You are familiar with the status management overview.
See Chapter Overview of status management A40 (Page 143).

Root object of status management

All templates for status management are controlled via base objects and collected below the following root: "@99 > A40 > Y10".

Up to 21 base objects can be created underneath this root object, whereby each represents a specific status. Elements representing the status values are created underneath the status base objects.

Properties of status base objects:

- Class: "Data set"
- Name: A number between 1 and 21.
If there is a gap in the digits, a corresponding gap is displayed in the properties.
- Description: The status text that is visible to the user.

Properties of status value base objects:

- Created as elements on the "Elements" tab of the status base objects
- Class: "Element"
- Name: A number from 0 to 3. All four elements must be created.
- Description: The status value text that is visible to the user.

Please note: Only 13 status base objects can be created with purely object-based status. The other 8 status base objects up to the maximum limit of 21 status base objects must be combined with query-based status management.

See also

Controlling Navigator commands via script (Page 147)

Controlling the status through attributes (Page 145)

Attributes for query-based status management (Page 152)

Status value defaults in base objects (Page 145)

Controlling the status using scripts (Page 146)

13.3.2 Status value defaults in base objects

Requirement

- The status base objects have been created.
See Chapter Creating status base objects (Page 144).

Default of object-based status

1. Open the properties of a base object.
2. Go to the "System" tab.
3. Check the "Status" input group.
Only the status of object-based status management is displayed. The number and position of the visible edit fields depend on which status objects have been created. The display can show a maximum of 13 object-based statuses. Gaps in the numbering of the names of status base objects result in gaps in the display on the "System" tab.
4. Enter one or more status values.
The status value base objects have numbers as names. These numbers can be entered in the fields of the "Status" group. These entries control which status values are visible by default for engineering objects.
The edit fields for the status value are jointly checked in. If you press the "X" button, all local inputs are deleted and all fields are inherited again.

13.3.3 Controlling the status through attributes

Requirement

- The status base objects have been created.
See chapter Creating status base objects (Page 144).

Opening properties

Properties of attributes, "General" tab

Field "Engineering object status"

The created statuses for purely object-based status management are available in this field. The attribute can be allocated to exactly one status.

Field "Engineering object status value"

A status value is determined for the above selected status. This status value is set in the event of an error.

The assignment of the attribute for status management can be removed again by pressing the "x" delete button.

Effect for status management:

The two details specified above are only evaluated when the user selects "Check" or "Set" in the planning object on the "Status" tab.

In the case of an engineering object this is followed by an evaluation of all attributes relating to this status. If an inconsistency is found in the attribute in question, the attribute is decreased to the selected status value. Attribute inconsistencies are highlighted in the properties of the engineering object using the color orange.

The following causes for inconsistency in an attribute may occur:

- A product data-related attribute infringes the template in the manufacturer device.
- A range attribute "Edit [Min Value Max]" infringes the checking of its own range. You can find additional information on this topic in the "Operation" manual, keyword "Checking via attributes".
- The value of the attribute does not correspond to that of the static link. You can find additional information on this topic in the "Operation" manual, keyword "Miscellaneous links".

If multiple status-related attributes exist in the object, the lowest status value is set for the object.

13.3.4 Controlling the status using scripts

Requirement

- The status base objects have been created.
See Chapter Creating status base objects (Page 144).

Overview

The script blocks "CheckStatus1" - "CheckStatus13" are located in the properties of a base object in the "Script" tab.

These functions are triggered if you click the "Check" or "Set" button in the "Status" tab of an engineering object for the relevant status. The "Check" or "Set" button is only available for states with object-based status management.

The text displayed in the "Info" column of the calculation log can be defined using `Workset.Lib.StatusInfo`.

13.3.5 Controlling Navigator commands via script

Requirement

- The status base objects have been created.
See Chapter Creating status base objects (Page 144).

Principle

You can generate the entries in the context menu of the Navigator individually. Only context menus for states based on object-based status management can be generated.

The component that controls the popup must have a global variable per class (in the case for "PPStatusCheck", "PPStatusSet") and the "Add" method must be run in "PopUp_Popup".

Example:

```
' ' Global variables:
Dim mPPStatusCheck As ComosPPGeneral.PPStatusCheck
Dim mPPStatusSet As ComosPPGeneral.PPStatusSet
. . .
Private Sub PopUp1_Popup()
. . .
If mPPStatusCheck Is Nothing Then
Set mPPStatusCheck = New ComosPPGeneral.PPStatusCheck
End If
mPPStatusCheck.Add PopUp1, m_CurObject

If mPPStatusSet Is Nothing Then
Set mPPStatusSet = New ComosPPGeneral.PPStatusSet
End If
mPPStatusSet.Add PopUp1, m_CurObject
. . .
End Sub
. . .
Public Sub Shutdown()
. . .
If Not mPPStatusCheck Is Nothing Then mPPStatusCheck.Shutdown
Set mPPStatusCheck = Nothing

If Not mPPStatusSet Is Nothing Then mPPStatusSet.Shutdown
Set mPPStatusSet = Nothing
. . .
End Sub
```

13.4 "A40 Status": Query-based status management

13.4.1 Purpose of query-based status management

Requirement

- You are familiar with the status management overview.
See chapter Overview of status management A40 (Page 143).

Overview

The query-based status is managed in an XML file that is located in the document directory. This file must not be deleted or locked.

Hierarchical query-based status management is an extended form of query-based status management. If the status of an object can be calculated, then hierarchical query-based status management works in the same manner as query-based status management. If the status cannot be calculated, then hierarchical query-based status management causes an object to be assigned the lowest status of all objects below it.

The following applies as well:

- Colors for status values: 1 - 10

Application

You can find additional information on this topic in the "Operation" manual, keyword "Working with the query-based status".

See also

Creating and administering a "Status" query (Page 148)
Hierarchical query-based status management (Page 149)
Query-based status management: Script examples (Page 150)
Background knowledge of XML storage of the status (Page 150)

13.4.2 Creating and administering a "Status" query

Requirement

- You are familiar with the purpose of query-based status management.
See Chapter Purpose of query-based status management (Page 148).

Creating a "Status" query

The existence of the "Status" query activates query-based status management for this status.

1. Select a status base object.
2. Select one of the two entries in the context menu:
 - "New > New query > User-defined status > Calculation"
The "Status" query is created.
 - "New > New query > User-defined status > Display"
The "StatusShow" query is created.

"@UserStatusValue" column

This query must have a column with the key name "@UserStatusValue". The values of this column are the actual status values.

Note

In the initial state the query does not return a calculated status. A script for the "@UserStatusValue" column must first be written before you can use query-based status administration.

"@UserStatusInfo" column

An additional special "@UserStatusInfo" column lists the additional information pertaining to the calculated values.

"@Filter" column

If the status calculation query has columns that begin with "@Filter", these are offered later in the "Filter" window. Status values are always offered as filters.

Note

The column with the name "@FilterUser" is used for the "For current user" context menu.

If a filter is active, this applies not only to the display of icons but also to the "Display" query. In addition, it affects the calculation of status values that are determined by the system itself (the values: -2,-1,0).

13.4.3 Hierarchical query-based status management

Requirement

- You are familiar with the purpose of query-based status management.
See Chapter Purpose of query-based status management (Page 148).

Activating hierarchical query-based status management

To activate hierarchical query-based status management, select the option "Hierarchical query-based status management" for a status in the "Attributes > System data" tab in the base data.

13.4.4 Background knowledge of XML storage of the status

Requirement

- You are familiar with the purpose of query-based status management. See Chapter Purpose of query-based status management (Page 148).

Storage of the XML file

If you select the "Calculate status" command in the context menu while the status display is active, the query is calculated with the selected start object and written to the project directory (or the working layer directory) in the form of an XML file with the name "...\\Status\\<Status number>\\1.xml".

Only those columns which begin with "@" are written. If an XML file already exists, it is updated. For that reason, it does not make any sense to change the calculation query in the meantime.

If a change to the query is unavoidable, first delete the corresponding XML file. Although the program checks the old file, it will only notice major changes to the query.

The `GetStatus(...)` COMOS method accesses values from this file during the calculation. Status values < 1 are not calculated in the file; they are calculated by the system itself.

The colors for the values up to status value 10 are reserved by the system.

The "Status per XML file" query

The "Status per XML file" query shows the current status of the XML status file. It is based on TopQDevices with the "ComosStatusByQuery.StatusShow" extender.

If no start object has been set or if the project itself is the start object, all data is displayed. Otherwise, only the data located below the entered object is displayed. No references are taken into account here.

13.4.5 Query-based status management: Script examples

Requirement

- You are familiar with the purpose of query-based status management. See chapter Purpose of query-based status management (Page 148).

Example 1: Fixed object properties

1. Create the "Status" query. The predefined columns are sufficient for this example.
2. Open the properties of the "@UserStatusValue > Status value" column.
3. Enter the following in the "Value calculation" tab:

```
Function ColumnValue(RefColObject, ColumnObject, BaseRowIndex,
BaseColumnIndex)
  ColumnValue = 99
  If Not ColumnObject Is Nothing Then
    TmpStr = ColumnObject.Label
    If Left(TmpStr,1) = "X" Then
      ColumnValue = 3
    Else
      ColumnValue = 2
    End If
  End If
End Function
```

"ColumnValue" is the status value. Status value "99" is used by default for an unset status value.

"ColumnObject" fetches the current engineering object, as in this case the row object is the engineering object. Read the label from here. If there is an "X" in the first label position, status value 3 is set. All other objects get status value 2.

Example 2: Comparison of properties

In this example the system reacts to user actions.

Objective: To label all objects that were edited after a certain date. This is done by comparing the timestamp against a specified date.

1. Create the "Status" query.
2. In addition to the predefined columns, create the default column for the date to the right of the label:
Select the "New > Timestamp > Date" command in the context menu of the label's table header. By default this column gets the name "Object_Date".

The status value is calculated as follows:

```
Function ColumnValue(RefColObject, ColumnObject, BaseRowIndex,
BaseColumnIndex)
  ColumnValue = 99
  If Not ColumnObject Is Nothing Then
    CurDate = BaseCell(BaseRowIndex, "Object_Date").Value
    If DateDiff("d", CurDate, DateSerial(2006, 03, 07)) > 0 Then
      ColumnValue = 2
    Else
      ColumnValue = 1
    End If
  End If
End Function
```

End Function

The value of the "Object_Date" column is saved in variable "CurDate" (any desired name). This value is the timestamp.

The difference between the current timestamp and the specified date is calculated with the "DateDiff" VB standard function. If the difference is greater than zero, status 2 is assigned. All other objects get status 1.

You can display the "complete status" with the above described procedure: Context menu of the white area of the Navigator, "Status display > <"Status"> > Complete".

To display your own status, change the "@FilterUser > User" default column in the status query as follows:

1. Properties, tab "Object evaluation": "Navigation library (short)": step "TimeStamp"; step "User".
You can find additional information on this topic in the "COMOS Administration" manual, keyword ""Properties" shortcut menu: "Object evaluation" tab".
2. Tab "Value calculation", calculation mode "Standard property", displays: Name (Name).
The "Status display > <"Status"> > For current user" command functions.

13.4.6 Attributes for query-based status management

Requirement

- The status base objects have been created.
See Chapter Creating status base objects (Page 144).

Overview

The following attributes can only be used in query-based status management.

Classification

Attribute	Description
Pipe spec mapping P&ID	The status management checks whether objects have been placed both in a P&ID report and also within Viper. You can find more information on this topic in the "P&ID" manual, keyword "Pipe spec mapping".
Screw calculation	Used in conjunction with the "Status screw calculation" list.
3D collision (ET mounting diagram)	The status management checks whether objects are colliding in the control cabinet. You can find more information on this topic in the "EI&C Operation" manual, keyword "Performing collision check".
COMOS <- XML -> PDMS	You can find more information on this topic in the "3D Integration Operation" manual, keyword ""Check status" operation".

Hierarchical query-based status management

See Chapter Hierarchical query-based status management (Page 149).

Status screw calculation

This list can only be used in conjunction with the "Screw calculation" classification.

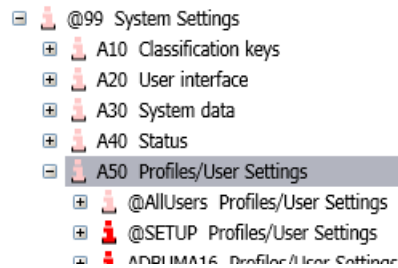
13.5 "A50 Profiles/user settings": Administration of personal settings

Requirement

- You are familiar with saving personal user settings in profile objects.
You can find additional information on this topic in the "Operation" manual, keyword ""Save settings" window".

Overview

Profile objects behave like normal base objects and can also, for example, be copied and renamed. There is the following hierarchy:



Structure	Description
"@99 > A50"	Only here are profile objects evaluated.
"@99 > A50 > @AllUsers"	Root node for generally applicable profile objects. Some general profile objects are overwritten by user-specific profile objects. Some profile objects are only available in "@AllUsers".
"@99 > A50 > @SETUP"	User settings for the standard account "SETUP".
"@99 > A50 > <User name>"	User-specific settings. If you delete the user-specific profile object, "@AllUsers" automatically applies.

Examples of profile objects

Profile object	Description	Effect
"BulkCheckInOptions"	Stores DQM options	DQM options are selected in a DVM document.
"DQMUserSystem"	Stores the settings of the query that form the "User information" window.	<ul style="list-style-type: none"> • "Plugins > PQM > User information" • Right-click the header of the query • In the context menu "Settings > Save"
"3D"	Stores some settings of the "Viper options" window.	
"Favorites"	User-specific favorites	
"FTSearchAttributes"	Profile object of the full-text search.	If this object exists and has attributes, the search is switched to the attributed search.
"FTSearchTemplates"	User-specific variant of the attributed search	
"LanguagesProfile"	If the user default is set differently to the project structure, this profile object is created.	<ul style="list-style-type: none"> • Project properties • "Languages" category > "User defaults" option box.
"Layers"	Layers You can find additional information on this topic in the "Operation" manual, keyword "Using layers".	This base object has effects on the interactive report, "Layers" button.
"MyLastDocs"	Shows the last DVM documents with which the user worked	
"NavigatorSearch"	The settings are automatically stored in the profile base object specific to the user. The settings apply to all projects.	<p>This base object has effects on searching in the Navigator:</p> <ul style="list-style-type: none"> • Right-click on an object • In the "Search" context menu • "Options" tab
"ProjectList"	Project standard table	Project selection (without the additional layers area).
"PrjSplitterPos"	Splitter position, project selection	For project selection, stores how large the areas for pure project selection and working layer selection are displayed. This setting is always saved and this takes place automatically.
"QueryUserSettings"	Settings for queries	
"SpecColorSet"	Profile for attribute colors	<p>You can define the attribute colors yourself:</p> <ul style="list-style-type: none"> • "Plugins > Basic > Define attribute color"
"StatusColorSet"	Color setting of status management, for working layers and histories	<ul style="list-style-type: none"> • Navigator context menu > "Legend" <p>See the "Properties" manual, "Legend" section. See the "Basic operation" manual, "Legend" section.</p>
"StatusProtocol_<N>"	Status calculation protocol for status <N> N stands for the status number	Query browser: Status calculation protocol For example, check the status in the Navigator when the status is enabled.
"WOList"	Working layers standard table	<ul style="list-style-type: none"> • Working layers selection in project selection or • context menu of the project root in the Navigator.

Local and global profile objects

Profile objects are stored in the system project, in the base project or in any engineering project. In the latter case you can store custom profile objects for each engineering project.

When you store profile objects on a project-specific basis, local base objects are created. In that case you can find a second local branch "@System" in the Navigator in the "Base data" tab, additional to the "@System" branch from the base data. The local branch has priority.

You must store profile objects for the working layer selection list in the base project.

13.6 "A90 Hardcoding": Using base data in script and code

13.6.1 Objective of the hardcoding object

Definition of hardcoding in COMOS

Hardcoding in COMOS means that reference is made to a base object in the script or source code with `SystemFullName`. Hardcoding means that script/source code and the referenced information must always be adapted at the same time.

The COMOS attributes of the "Reference" type represent the opposite. When the referenced base object is edited later (renamed or moved), the reference saved in the attribute is corrected by COMOS. The attribute value then contains the new name, or the attribute references to the new position of the object. This procedure is not hardcoding.

The following procedure enables you to avoid the disadvantages of hardcoding in script or source code by combining the hardcoding with reference attributes.

Overview of the procedure

- Use the prepared link object.
- There you use the prepared attributes of the type "Reference" or you create new attributes of this type.
- You reference to the base data by means of these attributes.
- You access the attribute contents by means of a .Net class.

You can move or rename the base data with this procedure without having to adapt the script or the source code.

See also

Overview of the "A90 Hardcoding" object (Page 156)

13.6.2 Overview of the "A90 Hardcoding" object

Requirement

- You are familiar with the purpose.
See chapter Objective of the hardcoding object (Page 155).

Overview of the prepared link object "A90 Hardcoding"

Storing the base object:

- "@99 > A90 Hardcoding > Y00R00070 Hardcoding object"

Common catalog attribute:

- "@40 > A10 > Y00 > A05 > Y00A06000 > Y00A06000 Hardcoding"

Base object tabs:

- Software hardcoding
This tab contains the hardcoding used in the COMOS source code.
Attribute name: Y00A06000Annnnn
- Scripting hardcoding
This tab contains the hardcoding used in the COMOS iDB scripts.
Attribute name: Y00A06000Snnnnn
- Customer hardcoding
This tab is available for customers to manage their own hardcoding.
Attribute name: Y00A06000Znnnnn

A query has been prepared on each tab. The query lists the base objects referenced in this tab. The query cannot be edited directly. Instead you use the option group "New hardcoding" to create new reference attributes.

See also

Creating a hardcoding link (Page 156)

Accessing the hardcoding object in script and source code (Page 157)

Alternative operation modes for hardcoding (Page 158)

13.6.3 Creating a hardcoding link

Requirement

- You are familiar with the overview of the hardcoding links.
See chapter Overview of the "A90 Hardcoding" object (Page 156).

Procedure

1. Select the base object for hardcoding.
See chapter Overview of the "A90 Hardcoding" object (Page 156).
2. Open the "Customer hardcoding" tab.
3. Optional: Check whether a "Hardcoding link" for the base object already exists in the list.
COMOS does not check automatically.
4. Drag a base object into the "Hardcoding link" field.
5. Optional: Enter a text in the "Comment" field.
6. Optional: Enter a text in the "Module" field.
The "Module" field describes in which COMOS module the referenced base object is used.
7. Click the "New hardcoding" button.
The query on the tab is updated.
8. Check the automatically filled fields "Object name" and "Description".
This information describes the referenced object.
9. Check the automatically filled field "Standard SystemFullName".
The SystemFullName is entered in this field whereby the hyphen "-" is used as separator.
10. Check the information in the "Identifier" column.
You use this information to access the attribute in the script or source code and thus the referenced base object.
See chapter Accessing the hardcoding object in script and source code (Page 157).
11. Optional: Lock the reference attribute.
 - Click the "Navigate to" button to the right of the "Hardcoding link" field.
 - Navigate to the attribute.
 - Open the properties of the attribute.
 - Click the "Object locking" button to set the property "Not possible to create a substructure".

In the "Customer hardcoding" tab the entry "True" is displayed in the "Locked" column.
The reference attribute can no longer be edited. This means the "Identifier" and the referenced base object remain a unit.

13.6.4 Accessing the hardcoding object in script and source code

Requirement

- You are familiar with the overview of the hardcoding links.
See chapter Overview of the "A90 Hardcoding" object (Page 156).

Availability of the "HardcodingMapper" class

HardcodingMapper is available at `Workset.Lib`. All implemented functions can be called without `CreateObject`.

Calling public functions of the "HardcodingMapper" class

You can access the hardcoding object as follows in the script:

- `Workset.Lib.HardcodingMapper.GetHardcodingForSystemFullName("Identifier")`
 - Return: String
 - Identifier: Corresponds to the value in the "Identifier" column in the query with the reference attributes
See also chapter [Creating a hardcoding link \(Page 156\)](#).
- `Workset.Lib.HardcodingMapper.GetHardcodedObject("Identifier")`
 - Return: Object
 - Identifier: Corresponds to the value in the "Identifier" column in the query with the reference attributes
See also chapter [Creating a hardcoding link \(Page 156\)](#).

13.6.5 Alternative operation modes for hardcoding

Requirement

- You are familiar with the overview of the hardcoding links.
See chapter [Overview of the "A90 Hardcoding" object \(Page 156\)](#).

Overview

- Outdated: Standard table Y40M00N00001M40
The entries of the standard table could also be addressed with the `HardcodingMapper` class. See chapter [Using hard-coded base objects in scripts \(Page 236\)](#).
- A10 Link object
This base object also uses reference attributes. But the `HardcodingMapper` class is not available. See chapter [Base references: References to database objects \(Page 245\)](#).

13.7 "B10 Tooling > @30 Tool": Administering cross-inheritances

13.7.1 Objective of "@30 Tool"

Requirement

- You are familiar with the "@30" node.
See also chapter "@30 Structures": Editing module-specific structures (Page 87).

Principle

The structure in the "@30" node is put together by means of cross-inheritance from objects of the nodes "@10", "@20" and "@50". Cross-inheritance means that a base object applies all properties of another base object by means of base object reference. The disadvantage of cross-inheritance is the fact that the hierarchical inheritance of properties and elements is interrupted at this base object. Hierarchical inheritance means that the parent object and all child objects apply the change.

Without hierarchical inheritance, all child objects must be edited themselves. The base object "@30 Tool" supports editing of the child objects by means of bulk processing:

- With the "@30 Tool", elements, tabs, documents and similar object components can be assigned in bulk and thus added.
- With the "@30 Tool", selected properties of the assigned elements can be edited.

See also

User interface reference of "@30 Tool" (Page 159)

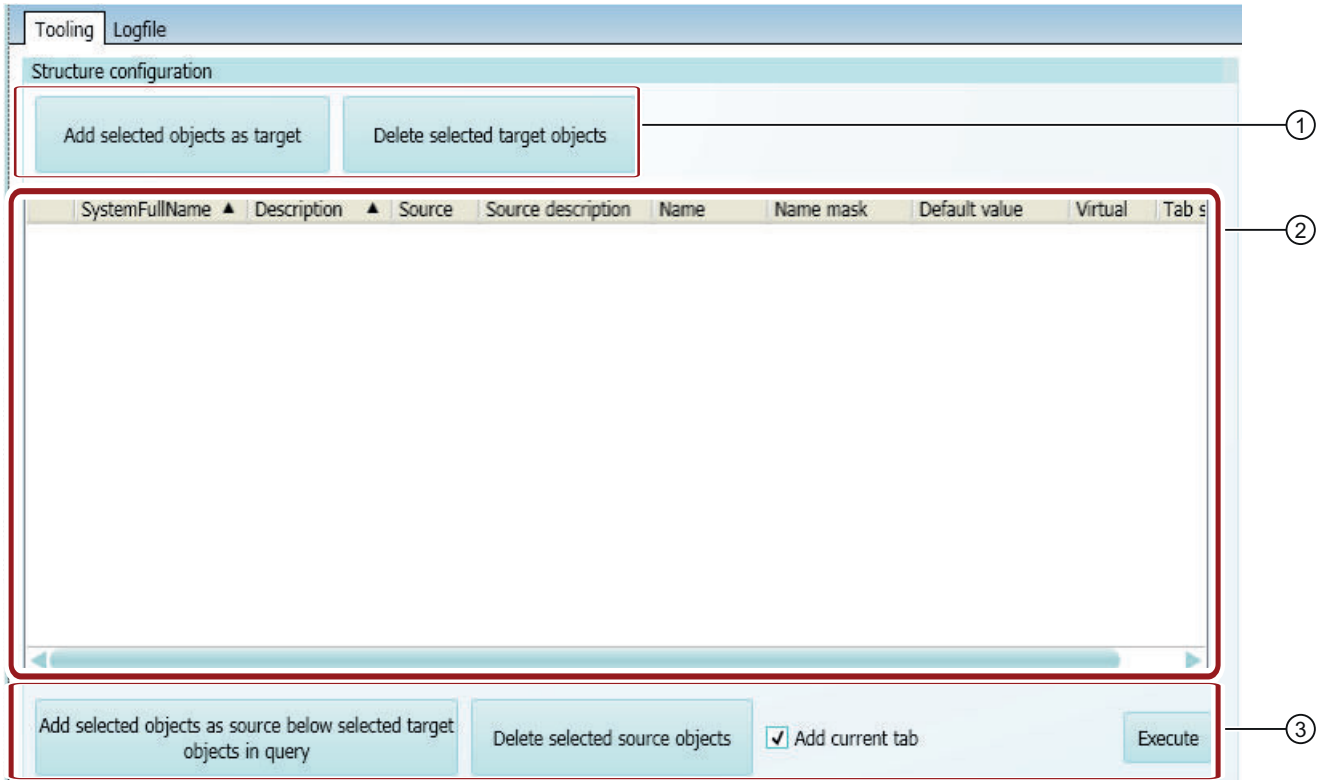
Using "@30 Tool" (Page 162)

13.7.2 User interface reference of "@30 Tool"

Requirement

- You are familiar with the objective of "@30 Tool".
See chapter Objective of "@30 Tool" (Page 159).

Reference to the area "Structure configuration" (assigning or removing object components)



①	<ul style="list-style-type: none"> • "Add selected objects as target" Base objects selected in the Navigator are added to list (2) • "Delete selected target objects" Base objects selected in the list (2) are removed from the list (2)
②	List of the target state
③	<ul style="list-style-type: none"> • "Add selected objects as source below selected target objects in query" Source objects selected in the Navigator are assigned to the selected base object in list (2) • "Delete selected source objects" The assignment to the base object is deleted in list (2) for the selected source objects. • "Add current tab" If the list (2) contains tabs, the following applies: Activated: The tabs are added as copy to the base objects. In case of identical names with an existing tab, a counter is added to the name. Deactivated: The tabs are added to the base objects. In case of identical names with an existing tab, the existing tab is discarded. • "Execute" The list of the target state is executed.

Reference to list (2)

Source	Source description	Name	Name mask	Default value	Label mask	Lab	Label default value	Virtual
SystemFullName ▲	@30 M41 A50	Description ▲	Density general					
	@50 DIN 2448-seamless p							
SystemFullName ▲	@30 M41 A50	Description ▲	Voltmeter					
	@40 Y Excel import							
	@40 Y Configuration							
	@40 Y Quick import							
SystemFullName ▲	@30 M41 A50	Description ▲	MDM_WIDM_ID					
	@40 Y Excel import							
	@40 Y Configuration							
	@40 Y Quick import							

④	Target object
⑤	Assigned substructure (in this example: tabs)

Reference to the area "Element edit" (editing selected properties)

Element edit

Add elements to list
Delete list
Save changes

Name	Default v...	Name mask	Label def...	Label name	Virtual m...	Label mask	SystemFullName
V10					Off		@10/A50/A10/B40!V10

⑥	<ul style="list-style-type: none"> "Add elements to list" An element object in the list (2) of the area "Structure configuration" is applied to the list (7). "Delete list" The list (7) is deleted. "Save changes" Changes to the list (7) are applied in list (2) of the "Structure configuration" area.
⑦	List of properties that can be edited of the element selected in list (2).

See also

Using "@30 Tool" (Page 162)

13.7.3 Using "@30 Tool"

Requirement

- You are familiar with the objective of "@30 Tool".
See chapter Objective of "@30 Tool" (Page 159).
- You are familiar with the reference of the user interface.
See chapter User interface reference of "@30 Tool" (Page 159).

Configuring base objects of the @30 branch

1. Change to the base project.
2. Select the "Base objects" tab.
3. Open the properties of the base object "@99 > B10 > A10 > Y00F00001 @30 Tool".
4. Select the "Attributes > Tooling" tab.
5. "@30 Structures" node Select one or more objects.
6. Click on the "Add selected objects as target" button in the "Structure configuration" area in the "Tooling" tab.
A base object entry is added to the list (2). The base object entry does not have any assigned source entries yet.
7. Optional: To remove a base object entry from the top list, click the "Delete selected target objects" button.
8. Select one or more base object entries in the list (2).
9. Select one or multiple objects in the Navigator in one of the following nodes:
10. "@30 Structures"
 - Elements
You can find additional information on the term "Element" in the "Administration" manual, keyword "Configuring elements".
 - Documents
Document as target object: a) A document from @30 or b) A document that was inherited from a different branch
Document as source object: a) A document from @30 or b) A document that was inherited from a different branch and checked in to (edited) @30
 - Tabs
a) A tab from @30 or b) A tab that was inherited from a @40 and checked in to (edited) @30
 - Base object as starting node of a base object branch

11. "@40 @Y Attributes and tabs"

- Tabs
A tab inherited from @40 can be replaced by a different tab from @40 having the same name.
Click the "Add selected objects as source below the target" button.
The objects selected in the Navigator are assigned for all objects that are selected in list (2).

12. Optional: Removing assigned substructures from the top list

- Select one or more entries in the substructure in the list (2).
- Click the "Delete selected source objects" button.

13. Optional: Editing properties of an element

- Select one or more elements in the list (2).
- Click the "Add elements to list" button.
The elements with the selected properties are displayed in the list (7).
- Edit the properties.

Note

Do not supplement the list (7) manually

For technical reasons the list (7) can be extended by placing the cursor in the last empty line. This procedure is not permissible and corresponding entries are also not saved.

- Click the "Save changes" button.
The edited properties are applied for the respective elements in the list (2). The elements in the Navigator still remain unchanged at this time.

14. Optional: Click the "Delete list" button in the "Define element" area.

The list (7) is emptied. When you apply new elements to the list (7) and edit them, you must not skip any superfluous old entries.

15. Optional: Select the option "Add current tab" in the "Structure configuration" area.

If the list (2) contains tabs, the following applies:

- Activated: The tabs are added as copy to the base objects. In case of identical names with an existing tab, a counter is added to the name.
- Deactivated: The tabs are added to the base objects. In case of identical names with an existing tab, the existing tab is discarded.

16. Click "Execute".

The list of the target state is executed. The objects in the Navigator are changed. Depending on the setting, a dialog for saving the log is displayed. See the following block "Save log" for more.

Save log

1. Select the "Attributes > Logfile" tab.
2. Optional: Click the "Clear log" button.

3. Edit the "Logfile configuration" area.
 - "Logfile path": Path in which the log file is saved
 - "File name": Log file name
4. Check the setting of the option "Parallel log file creation".
 - Activated (default setting):
The log file is saved automatically when you click the "Execute" button in the "Tooling" tab.
If information is missing in the "Logfile configuration" area, a dialog is displayed in which the missing information is entered.
 - Deactivated:
You must save the log file manually by clicking the "Save as log file" button.
5. Click the "Save as log file" button.
If information is missing in the "Logfile configuration" area, a dialog is displayed in which the missing information is entered.

Administration of standard tables

14.1 Overview of standard tables in the iDB

Main statement

There are five types of standard tables:

- "Attribute catalog"
The standard tables that are used by attributes are stored together here.
- "System tables"
These standard tables contain data that is used by the system.
- "Special standard tables"
These standard tables contain data that is used by external applications.
- "Mapping tables"
The various ways of searching for data or objects are mapped to each other in these standard tables.
- NLS tables
This is where the data to be used for localizing is collected.

14.2 "@ Attribute catalog table"

Aim

The standard tables that are used by attributes are placed here. This node is divided into three areas:

- "Y00 Standard table for attributes"
Standard tables of COMOS
- "Y30 Special standard tables collection"
Selection list for specific subjects
- "Z10 Standard table for customer attributes"
Freely available to customers

Storage and assigning names

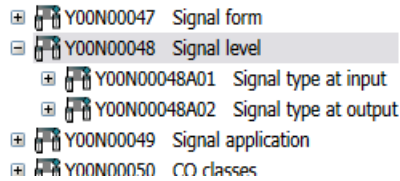
Note

All standard tables that are assigned to attributes are only copied in the tab at the derived attribute.

14.2 "@40 Attribute catalog table"

- The name of a standard table is made up as follows: Y00N + consecutive five-digit number.
Example: Y00N00001
- Special standard tables receive the prefix "Y20"
Y30Nnnnnn (e.g. Y30N00001)
- Customer-specific standard tables receive the prefix "Z10"
Z10Nnnnnn (e.g.: Z10N00001)

When standard tables belong together thematically, the variants are assembled together below a grouping standard table. Meaningful descriptions are used to distinguish the variants. The description is also selected based on the tab in which the standard table of attributes is opened. The variants receive a suffix, for example A01/A02:



The higher-level object (Y00N00048 in the above example) contains no values by definition. The subordinate standard tables contain the values.

Note

The search for standard tables takes place by means of queries. This search is textual and runs through the "Description" field of the respective standard table.

The standard tables are grouped in accordance with the same rules as with the attributes:

- Grouping in packages of 500
- Below grouping in packages of 25

Note

In the delivery state, COMOS iDB does not use standard tables that are checked in or created in the engineering project.

14.3 Y10 System tables

Aim

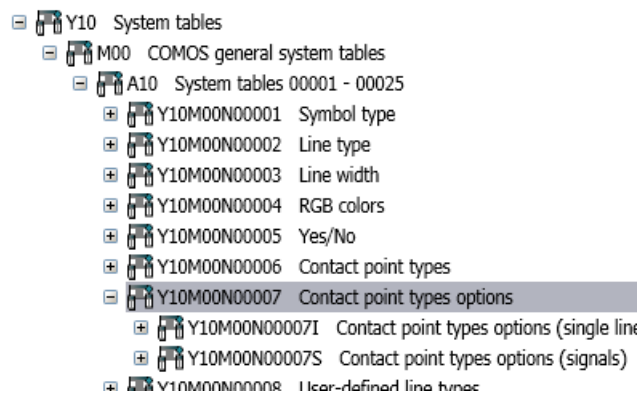
The standard tables that are only used by the COMOS software are placed here. These standard tables are not called by attributes. A standard table that is used by attributes may also be set up as a @40 standard table. This node is divided into the following areas:

- One node per COMOS module
See chapter Special cases when using letters (Page 30).
- "Z10 Customer system tables"
Freely available to customers

Storage and assigning names

- The name is made up as follows: Y10 + COMOS module number "Mxx" + "N" + consecutive five-digit number starting at 00001.
Example: Y10M00N00001

When standard tables belong together thematically, the variants are assembled together below a grouping standard table. Meaningful descriptions are used to distinguish the variants. The variants receive a suffix, for example "A0n". The suffixes "I" and "S" are used in the following example (for the drawing type identifier in each case):



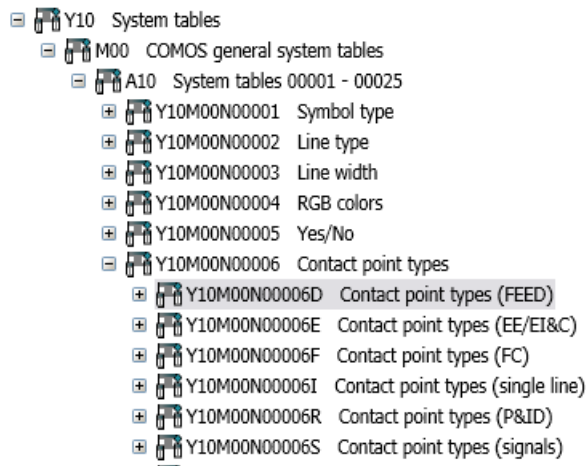
14.3.1 Y10 Standard tables for line types

There are three standard tables for line types in the COMOS iDB:

- Standard tables > Y10 > M00 > A10 > Y10M00N00002 Line type
The COMOS line types (formerly: Logocad line types) are saved here.
- Standard tables > Y10 > M00 > A10 > Y10M00N00008 User-defined line types
You can find additional information on this topic in the "Administration" manual, keyword "Structure of the standard table "User defined line types"".
- Standard tables > Y10 > M00 > A10 > Y10M00N00009 Drawing type specific line types
You can find additional information on this topic in the "Administration" manual, keyword "Line types in the standard table "Drawing type specific line types"". To make a drawing type-specific addition, more standard tables with the name of the drawing type are created below the Y10M00N00009 standard table.

14.3.2 Y10 Standard tables for connector subtypes

The name of the standard table is composed of the name of the grouping system standard table and the connector type identifier.



The connector subtypes are created as standard table values.

14.4 Y30 Special standard tables

This is where standard tables that are not iDB-compliant or only used by Y30 attributes are stored. The naming convention is analogous to the other standard tables. The prefix for special standard tables is "Y30".

14.5 Y40 Mapping tables

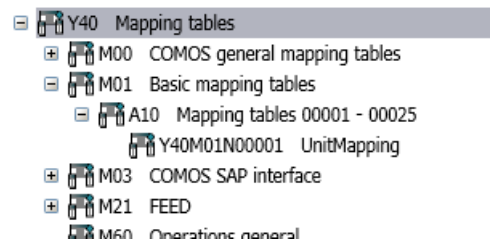
Aim

This is where standard tables in which data is mapped to each other are stored. This node is divided into the following areas:

- One node per COMOS module
See chapter Special cases when using letters (Page 30).
- "Z10 Customer mapping tables"
Freely available to customers

Storage and assigning names

- The name is made up as follows: Y40 + COMOS module number "Mxx" + "N" + consecutive five-digit number starting at 00001.
Example: Y40M00N00001



Special standard tables

- UnitMapping
You can find additional information on this topic in the "Administration" manual, keyword "Creating and editing units".

See also

Administering units (Page 171)

14.6 Y60 NLS tables

Overview

This is where NLS (National Language Support) texts are stored and managed. This node is divided into the following areas:

- "Y00 COMOS NLS tables"
Text supplied with COMOS.
- "Z10 Customer NLS tables"
Freely available to customers

See section Administering NLS texts in standard tables (Page 229) for more on this.

Administering units

15.1 Administering unit groups for the iDB

Requirement

- You are familiar with the basic technique for creating unit groups. You can find additional information on this topic in the "Administration" manual, keyword "Creating and editing unit groups".

Procedure

The unit system is structured in the style of the IEC 62720 standard.

The unit groups are named using the iDB count in increments of 5. Each unit group includes a number of units that can be converted into one another by means of a calculation. A unit is only assigned to one unit group. Example: The "Meter" unit only exists in one unit group.

Structure of unit names:

1. Prefix: Name of associated unit group followed by a period as separator
2. Main part: two letters followed by three numbers. The first letter indicates the unit system:
 - A: Metric unit system
 - B: Imperial unit system
 - C: Chinese unit system
 - D: Other unit system

The other places are counted in increments of 10 with iDB counting. Sorting is made according to the "size" of the unit, with the largest unit being indicated first.

Example: "B75.AA180 Meter" comes before "B75.AA210 Millimeter".

The associated conversion factors are configured. No decimal places are entered for the units.

See also

Using units in iDB attributes (Page 171)

15.2 Using units in iDB attributes

Requirement

- You are familiar with the administration of the unit groups. See section Administering unit groups for the iDB (Page 171).

Procedure

- A unit is not entered at the attributes in the attribute catalog.
- The unit used is selected for the derived attributes in the tabs:
 - To do this, in the "@40 > A20" node select a tab with an attribute that has a unit.
 - Click the button with the three dots in the "Unit" field.
 - The selection "Unit" is then determined in the window that opens, and the correct unit is given after selection of the unit group.

By determining the group the unit system can be switched from "Metric" to "British". If no group is selected at this point and instead the unit is selected directly, values that are not copied will not be converted.

Note

Unit groups should only be used for a good reason in exceptional cases.

There is no project-wide conversion of units for the iDB.

Administration of the labeling system and ALIAS labeling system

16

16.1 Constructing the labeling system with base object elements

Purpose of a labeling system

Labeling systems are a standard method for labeling all objects in a unit. In COMOS, labeling systems are structured using two working techniques:

- Via the automatic assignment of names or labels for newly created engineering objects. In this case, the base data is prepared so that, when the engineering objects are created, the names are assigned according to the labeling system.
- Via the automatic sorting of documents into standard-specific document folders.

Controlling labeling systems using the "Virtual" option

- The "Virtual 1-time" or "N-times" setting
The "Virtual" option determines how often the element can be created in the labeling system.

Controlling labeling systems using the "Dereferenced" option

The "Dereferenced" option is only available if the base object has been created on the "Elements" tab of a different base object.

"Dereferenced" changes the inheritance structures: If an object is created in the engineering project on the basis of this base object, only the following fields are adopted:

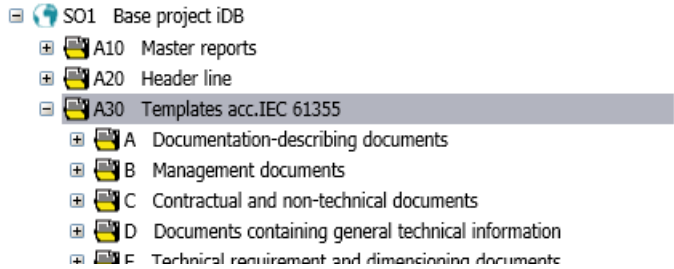
- "Name"
- "Label"
- "Description"
- "Count"

When the engineering object is created, the information that it was derived from the dereferenced base object is lost. If a change is later made to one of these four properties in the engineering view, this only affects newly created engineering objects.

Instead, the connection between the engineering object and the base object refers to a layer: The base object which is created as a "reference" for the dereferenced base object becomes the new base object of the engineering object.

Standard-specific labeling systems for documents

Standard-specific labeling systems are constructed with the help of base object structures:



Controlling labeling system using text masks

Use the following standard table:

"Standard tables > Y10 > M00 > A10 > Y10M00N00019 Masks (default)"

(Old: MASK)

The standard table contains the start value and mask (regular expression) for the text masks of objects that are used for unit labels.

- Name = FullName of the CDevice
- Value1 = Mask
- Value2 = Start value

The list can be adapted and used. The list is checked into the project and the values are modified accordingly.

More standard tables for masks can be prepared according to other naming systems. Example: "Y10M41N00016 Masks EIC".

See also

Labeling systems in the M00 branch (Page 175)

16.2 Labeling systems in the M00 branch

Labeling systems in @30 > M00

Because tagging systems are structural information, they are stored in node @30 Structures. The labeling systems are integrated into the normal subnodes in node "M00":

- [-] @30 Structures
 - [-] M00 COMOS general
 - [+] A10 General structures
 - [+] A20 Project presettings
 - [-] A30 Units
 - [+] A10 Neutral unit structures
 - [+] A20 General unit structures
 - [+] A30 Structures according to KKS
 - [+] A40 Structures according to RDS-PP
 - [+] A50 Structures according to AKZ
 - [+] A35 Categories
 - [-] A40 Locations
 - [+] A10 Neutral location structures
 - [+] A20 General location structures
 - [+] A30 Structures according to KKS
 - [+] A40 Structures according to RDS-PP
 - [+] A50 Structures according to OKZ
 - [+] A50 Devices structures
 - [+] A60 Functions

There are no central root nodes below which all components of a labeling system would be available.

The base objects used are cross-inherited from the @10, @20 and @50 nodes.

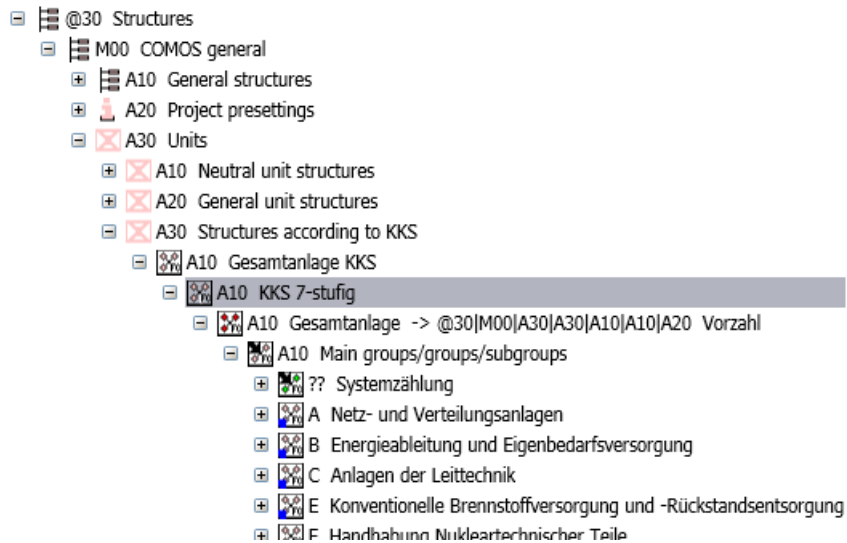
A20 Project settings

All elements which belong to the corresponding labeling system are referenced here. These must all come from the corresponding labeling system base node from node @30:

- [-] @30 Structures
 - [-] M00 COMOS general
 - [+] A10 General structures
 - [-] A20 Project presettings
 - [-] A10 Project presetting, global
 - [+] A10 Project presetting, common example
 - [+] A20 Project presetting, FEED example
 - [+] A30 Project presetting, P&ID example
 - [+] A40 Project presetting, E&IC example
 - [+] A50 Project presetting, Fluidics example
 - [+] A60 Project presetting, PCS7 AdvES example
 - [-] A70 Project presetting, KKS example
 - [+] A10 Gesamtanlage -> @30|M00|A30|A30/
 - [+] A20 Gesamtanlage -> @30|M00|A40|A30/
 - [+] A80 User defaults
 - [+] A90 Project presetting, Automation example

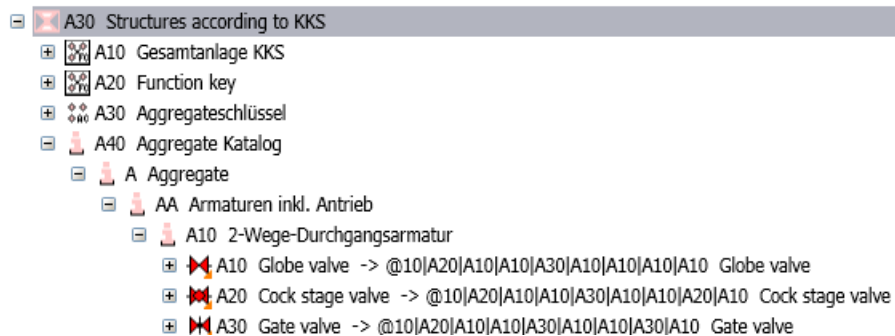
A30 Units

Unit structures for the labeling system:



- All objects from node @30 which can be created on the "Units" tab are derived from node @10.

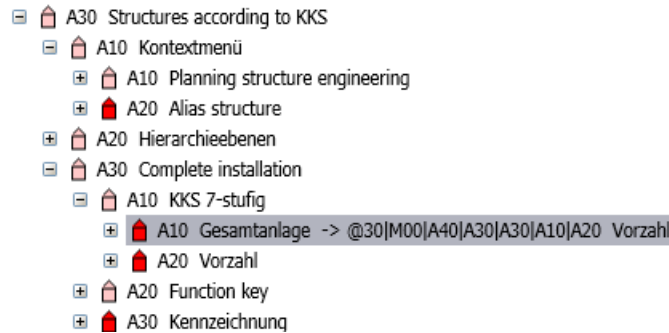
Example: "@30 > M00 > A30 > A30 > A40 > A > AA > A10"



- The unit structures are compiled into the desired structure using the base objects from node @20 as elements. This is achieved using cross-inheritance.

A40 Locations

Location structures for the labeling system:



Constructing labeling systems with base objects

1. Select the relevant root node for the units, locations, and so on.
2. Insert the base objects into the relevant root node as elements using drag&drop, or create them as new objects.
The labeling system is structured using the elements. The elements can have elements of their own.
3. Optional: You can provide the alias structure as well as the normal structure in the root object of the project.

If a different type of designation is specified by labeling systems or standards in catalogs, etc., then deviations from the standard designation, starting from the node for the corresponding substructure, are permitted.

Naming conventions

- Labeling systems can be prepared using an element structure.
- Labeling systems can be prepared using text masks.

See chapter Constructing the labeling system with base object elements (Page 173).

Controlling the creation of engineering objects using dereferencing

The objects in @30 are generated using cross-inheritance. The "Dereference" option therefore has special significance in branch @30. This option specifies that it is not the base object itself which is created, but rather the source of the base object reference. See chapter Controlling base object links using dereferencing (Page 46).

16.3 Purpose of ALIAS labeling systems

Requirement

- A basic labeling system has been set up.
See chapter Constructing the labeling system with base object elements (Page 173).

Aim

An ALIAS labeling system is an alternative labeling system to the basic labeling system. You can switch between the two labeling systems at any time.

To achieve this, the object structure of the project is rebuilt in an @ALIAS branch. The objects in the @Alias branch get a pointer to the originals. Technically, the use of alias is a link of two devices. Only the original has the entire information (attributes, icons, etc.). The alias object only has a pointer to the original, its own labeling and another location in the object tree. If an alias device is opened and the properties are edited, then the original is edited.

See also

Administration of the ALIAS labeling system (Page 178)

Using scripts to adjust ALIAS labeling (Page 180)

Labeling systems in the M00 branch (Page 175)

16.4 Administration of the ALIAS labeling system

Requirement

- You are familiar with the purpose of the alias labeling system.
See chapter Purpose of ALIAS labeling systems (Page 178).

Step 1: Labeling system

The project is planned as usual, using the basic labeling system.

Step 2: Alias root object

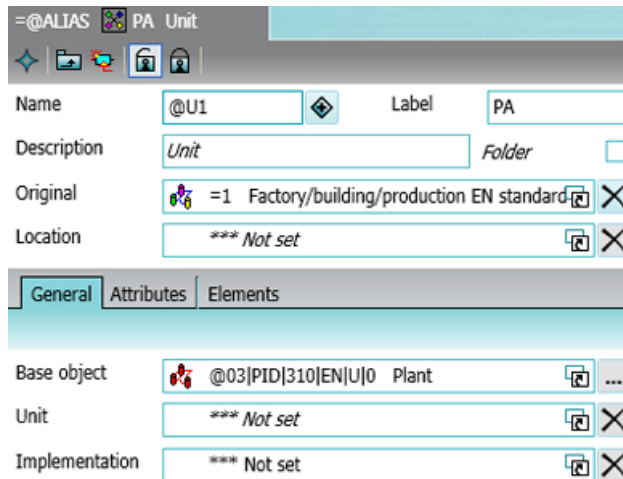
Create an @Alias branch in the engineering view:

- Tab: "Unit" and/or "Location", project root: "New > New unit" and/or "New > New location"
Name of the new unit / new location: "@Alias" (mandatory name; not case-sensitive)
All objects with system type "Device" automatically get an additional reference field.
 - Name of the reference field for the objects in the @Alias branch: "Original"
 - Name of the reference field for the originals: "Alias"

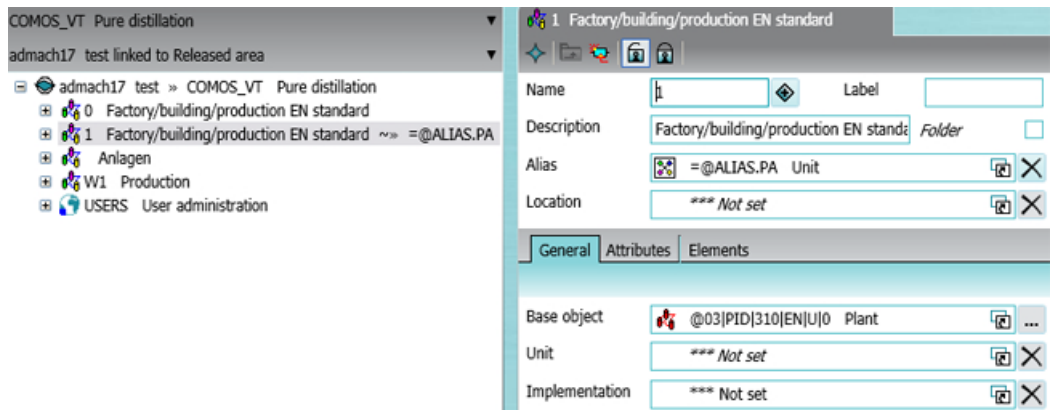
Step 3: Alias objects

Rebuild the object structure of the original data under the @Alias branch, and assign references to the original data:

1. Use "New > New unit" to create an alias unit "PA" for the original unit "W1".
2. Enter the tag from the alternative labeling system in the label property of the alias unit.
3. Set a pointer at the alias object, referencing the corresponding original. For that purpose, drag&drop the original unit onto the "Original" reference field of the alias unit:



In the properties of the original, a pointer to the alias object is automatically assigned to the "Alias" reference field:



The references appear in the Navigator and are identified by "~". As long as both objects reference each other in their "Alias" property, you can use the "Navigate > Alias" and "Navigate > Original" context menu commands to toggle between the two objects.

All engineering-relevant information continues to be maintained at the originals. Thus, the alias objects only get a pointer and a different label.

It depends on the respective requirements up to which level the original data is rebuilt in the @ALIAS branch (only units/ locations or also devices).

Note

No bulk processing for an alias labeling system

Use the properties of an object (tab with the properties) to set or edit an alias pointer or a pointer to the original.

Step 4: Activating alias

Activate the Alias function using one of the two following options:

- By script
"Workset.UseAliasForLabelFunctions =True"
The AliasLabel script commands can only be executed if the value is True.
- Via the COMOS user interface
If an object named "@Alias" is located in the engineering data directly below the project root, the "A" button is displayed in the COMOS menu bar when the project is opened.
Default when opening a project: "False".

See also

Using scripts to adjust ALIAS labeling (Page 180)

@20 > A25 Positions (Page 72)

16.5 Using scripts to adjust ALIAS labeling

Requirement

- You are familiar with the administration of an ALIAS labeling system.
Administration of the ALIAS labeling system (Page 178)

Using scripts to activate the alias function

The ALIAS system is deactivated by default. It can be activated on the menu bar using the icon or scripts:

```
Workset.UseAliasForLabelFunctions=True
```

If `Workset.UseAliasForLabelFunctions == True` applies, the name is structured using the Alias branch.

If `Workset.UseAliasForLabelFunctions == False` applies, the name is structured using the regular branch.

This makes it possible to switch the output during operation and applies to all objects.

Using AliasLabel

Either the label or the AliasLabel can be queried at the objects.

If `Workset.UseAliasForLabelFunctions = True`, then:

- If one of the `AliasLabel` commands defined in the COMOS DLL is accessed for an object (`Object_A`), the `Alias` property of `Object_A` is evaluated.
- If the object referenced in this property (`Object_B`) in turn has an `Alias` reference to `Object_A`, the `AliasLabel` command is executed for `Object_B`.
- If the `Alias` property is empty or if `Object_B` does not reference `Object_A`, the method is conducted for `Object_A` instead.

Note

It may happen that you want to call an `AliasLabel` command for an original, but that you are already at the `Alias` object and thereby mistakenly retrieve the label of the original.

To avoid this error, call the `IsAlias()` function first. The function verifies whether an object is located in the `@Alias` branch.

You can find more information on this topic in the "Class Documentation COMOS_dll" manual, keyword "Alias".

Specialties

The alias system does not influence the performance of COMOS.

If possible, the alias function should be used at the corresponding positions in scripts and also in the program.

For example: `AliasFullLabel` instead of `SystemFullLabel`

It is important to ensure that switching at the relevant positions is carried out correctly. Checking is carried out within the modules.

Classifying objects

17.1 Purpose of classification

Objectives of classification

Classification has the following objectives:

1. Search: The objective is to find a particular object or similar objects. You are supported by specialized queries when doing so.
2. Control: The objective is to assign similar reactions. In this case, either the objects themselves can react in the same way or else the environment treats all objects in the same way.
3. Standardization: The objective is to create uniform names for objects across databases and companies. This simplifies matching, exchange, and merging.

Application cases

- Cross-project queries use the classification
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Creating a query: Cross-project queries".
- Database search uses the classification.
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Overview of the database search".

No separate classification of engineering data

Only the base data (base objects, document base object, etc.) is classified. The engineering objects and engineering documents apply the classifications from the base data.

See also

Structure of classifications (Page 183)

Performing the classification (Page 186)

Classification of documents (Page 187)

17.2 Structure of classifications

Requirement

- You are familiar with the purpose of classifications.
See chapter Purpose of classification (Page 183).

Root node

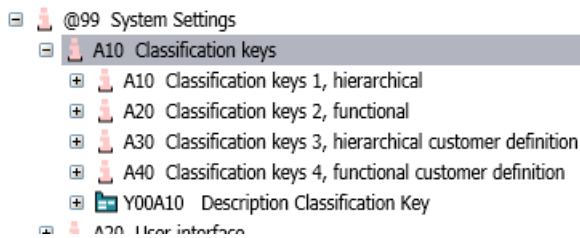
The classification keys are stored in "@99 > A10" system node.

Basic variants of classifications

COMOS uses two types of classification keys:

- Hierarchical classification
Five hierarchically dependent fields with 1 to 4 characters each.
In hierarchical classification, the subsequent layers (2, 3, etc.) can only be set in relation to the layers before them.
Default in the iDB: The first layer of the hierarchical classification is a three-digit key for the module name (M00 for Basic, M20 for Process, M40 for Automation, and so on).
- Functional classification
Five independent fields with 1 to 4 characters each.
You can assign any combination of functional classification keys.
Default in the iDB: The first functional classification of all objects is a key made up of two digits twice. The first two digits are a key for the class. The following two digits are a key for the subclass.

Both the hierarchical classifications and the functional classifications exist separately for default objects and for customer objects. For this reason, four basic variants for classifications appear below "@99 > A10":



- Classification key 1: COMOS (hierarchic)
- Classification key 2: COMOS (functional)
- Classification key 3: Customer (hierarchic)
- Classification key 4: Customer (functional)

The hierarchic classification with prefix "Z" is no longer reserved for customer-specific classifications.

It is permissible for the classification key to be set and inherited for an object with the "Structure" creation option.

Note**Changes are not permitted for classification key 1 and classification key 2**

Classification key 1 and classification key 2 are created and delivered exclusively by Siemens. Classification key 1 and classification key 2 cannot be changed and are also constant in future iDB versions. The descriptions and information texts can be changed.

Classification key 1 and classification key 2 are protected with the "lock-lock" mechanism to protect them from changes. See chapter "Object locking" and "System object" state (Page 37).

Note**Incompatible classification keys of cDB and iDB**

The components of the classification keys of the cDB consist of 2 digits.

The components of the classification keys of the iDB consist of 4 digits.

Therefore, the classification keys of cDB and iDB are incompatible. It is not possible to export objects from the cDB to the iDB without editing.

Use of the classification keys in script/memory method of the classifications

Classification keys 1 to 4 are each saved as a contiguous string. The components of the classification key are separated by a dot.

Engineering objects and engineering documents do not have their own classification key:

- Determining classification key for engineering objects (example: object debugger):
`a.CDevice.GetClassification(index)`
- Determining classification key for engineering documents (example: object debugger):
`a.CDocument.GetClassification(index)`

You can find additional information on this topic in the "Class Documentation COMOS_dll" manual, keywords:

- "GetClassification"
- "OwnerByClassification"
- "ClassificationExists"

Customer-specific classification

The naming convention for classification code 3 takes place analogous to classification key 1 and the naming convention for classification key 4 takes place analogous to classification key 2.

Relationship between classifications and class/subclass

The "Class" and "Subclass" standard tables are set in the "System" tab in the properties of the base objects. The "Class" and "Subclass" properties are also reproduced in the iDB with the help of functional classifications.

- Reproduction of class and subclass by means of functional classification
 Functional classification "@99 > A10 > A20 > N1" and all subsequent classification keys up to "P8D3"
 - Two characters for the class
 - Two characters for the subclass

Example: N6 -> dataset, B4 -> component; B5 -> participant address

Class	Subclass
N6	
N6	B4
N6	B5

The "Class" and "Subclass" standard tables are still available for reasons of compatibility.

Note

Classifications and class/subclass must be consistent

When new base objects are created, the settings for classifications and class/subclass must be consistent.

The base objects supplied in the iDB are protected against accidental modification (and thus against inconsistency) using a lock-lock mechanism. See chapter "Object locking" and "System object" state (Page 37).

For objects you create yourself, you must check and match the classification and the setting for class and subclass. This applies in particular to importing/exporting objects.

Classification keys for documents

You can find additional information on the classification keys for documents under Classification of documents (Page 187).

17.3 Performing the classification

Requirement

- You are familiar with the purpose of classifications.
 See chapter Purpose of classification (Page 183).

Classification of a base object

To perform the classification follow these steps:

1. Open the "Base objects" tab in the base project in the Navigator.
2. Select the base object which you want to classify.
3. Right-click on the base object and select the "Properties" command.
The properties of the base object are displayed.
4. In the properties, click on the "Classification" tab.
 - Hierarchical classification
Click the button with the three dots.
 - Functional classification
Select the entry in the first drop-down list.
The next drop-down list then becomes active.
Select as many functional classifications as are required, one after the other.

Classification codes cannot be entered manually in the fields. You must select entries in the drop-down fields.

Update classification

To perform a classification update, follow these steps:

1. Select one of the following options:
 - If you only want to perform the classification update for the currently open project, click the "Administrator> Update classification > Current project" menu.
 - If you want to perform the classification update for all projects in the open database, click the "Administrator> Update classification > All projects" menu.

17.4 Classification of documents

Requirement

- You are familiar with the purpose of classifications.
See chapter Purpose of classification (Page 183).
- You are familiar with the document management overview.
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Overview of document management in COMOS".

Overview of the classification for documents

Document management consists of the following independent classified components:

1. The document base object.
2. For COMOS-internal documents: the report template to the engineering document
 - Master reports
 - Subreports
 - Report templates in the style of IEC 61355 (main reports)
3. The engineering document
4. The document groups in which the report templates are sorted

For component 1: Prepared classification key for the document base objects

The document base object is classified with the hierarchical classification key for documents according to IEC 61355.

- Hierarchical classification
The hierarchical classification is below the node "@99 > A10 Classification key" in the Navigator.
The general classification code for the document base object is: "A10 > M00 > A140 Documents acc. IEC 61355"
- The classification in the module-specific node is stored for a module-specific document base object. Example:
Document base object: "@20 > A60 > A10 > A30 Document base objects for DVM"
Classification: "@99 > A10 > A10 > M65 > A030 Document base objects for DMS"
- Functional classification
Functional classification of documents: "N7 Document"

For component 2: Classification of report templates / document templates

The following reports apply the hierarchical classification key according to IEC 61355 from the document base object:

- Master reports
- Headers and footers
- Subreports

Hierarchical classification

Hierarchical classifications for document templates are formed in a DCC structure below the node "@99 > A10 > A10 > M00 > A140 Documents acc. IEC 61355" and set in the main report in accordance with your DCC structure:

- First key: Module; for most document templates: M00
- Second key: iDB count Axxx; in M00 module: "A140 Documents acc. IEC 61355"

- Third key: DCC key with two letters
- Fourth key: Matched to the name of the report template. Example: Name of the report template "FP_B02" - associated classification "B02".

The classification is set on the main report. Subreports obtain the hierarchical key for documents "M00 > A140 Documents acc. IEC 61355".

Functional classification

- First functional classification
Classification according to the document properties, "Type" field in the "General" tab.
Examples:
 - "A040 Interactive report"
 - "A050 Interactive report template"
 - "A060 Evaluating report template"
 - "A070 Evaluating report"
- Second functional classification
Classification according to the object of the document template in the iDB:
 - Report templates in node "A10 Master reports": Classification with "B980 Master report"
 - If the name has two components (for example FS_A20): Classification with "B990 Main report"
 - If the name has three components (for example DA_A10_A10): Classification with "C010 Subreport"
Example: Report templates in the node "A20 Header"
- Third functional classification
Classification according to COMOS function or COMOS module
Example: "B100 Process flow diagram (PFD)"

For component 3: Structure of the document groups for storing the report templates

The structure of the document groups for storing the report templates corresponds to a specific level of the structure of the hierarchical classification codes.

Example:

Documents tab: "A30 > A > *AA > AA_A10 > AA_A10"

Base objects tab: "@99 > A10 > A10 > M00 > A140 > *AA > A10"

In the two paths above, one node each is identified with an asterisk. The filing structure is the same as of this level.

Classification for documents under base objects

Documents under base objects inherit their classification from the `CDocument` or the `CObject` rather than from the owner. See chapter Structure of classifications (Page 183), block "Use of the classification keys in script".

Classification for engineering documents

Engineering documents adopt the classification of the document template. If no classification is set in the document template, the engineering documents adopt the classification of the base object.

Administration of documents in the iDB

18.1 Root nodes of the documents

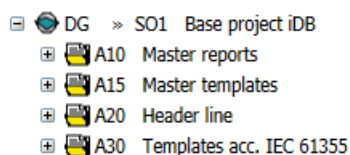
Relationship with external files

Reports consist of a report object in COMOS and an external .CRp file.

See section Physical storage of the report templates (Page 294).

Overview of storage

The documents are made available in the "Documents" tab in the Navigator. The following nodes are here:



The templates for interactive reports and evaluating reports are located in the node A30 and are sorted according to IEC 61355.

A10 Master reports

The master reports are arranged in accordance with their format. A master report is created for each format. They only contain one frame and are otherwise empty:

The "A10 Master reports" node is divided into DIN and ANSI formats. Sorting in the DIN area takes place in accordance with the ISO-5457 standard.

Example: The following 2 reports are below the A10 > A10 > A10 DIN A0 node:

- A10 Master report DIN A0 portrait
- A10 Master report DIN A0 landscape

The master reports are evaluating reports as standard. To use an interactive report, insert the base object of the corresponding evaluating report and change the type to interactive in the report properties.

A15 Master templates

The master templates are the templates for evaluating reports. The master template is a combination of master report and header and/or footer.

18.1 Root nodes of the documents

This node is divided into the following areas:

- Data sheets
- Lists

The master reports in the nodes "A10 Data sheets" and "A20 Lists" are distinguished by the standard they use. Example: "COMOS iDB Standard", "PIP Standard".

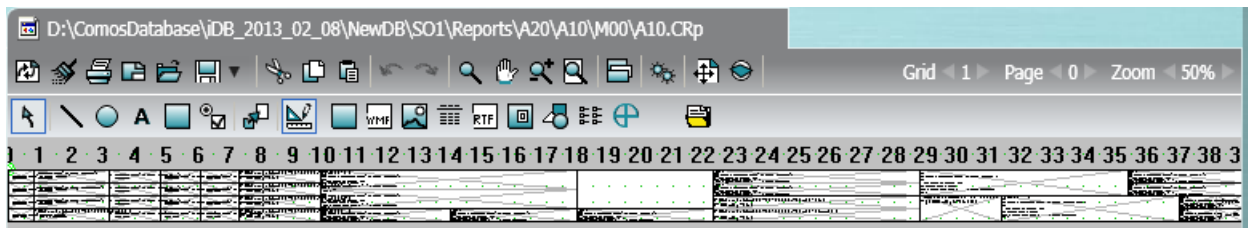
For the COMOS iDB standard:

- All data sheets: DIN A4 portrait / ANSI A portrait
- All lists: DIN A3 landscape / ANSI B landscape

A20 Header

This node contains templates for headers and footers.

The header line reports are sorted in the style of the DIN and ANSI standards and the respective modules. The header line reports include a predefined document header:



A30 Template acc. IEC 61355

The document templates are sorted in the style of the standard for classification and designation of documents (DCC) IEC 61355.

- [-] A30 Templates acc.IEC 61355
 - [+] A Documentation-describing documents
 - [+] B Management documents
 - [+] C Contractual and non-technical documents
 - [+] D Documents containing general technical information
 - [+] E Technical requirement and dimensioning documents

Each document template is grouped under a separate document folder.

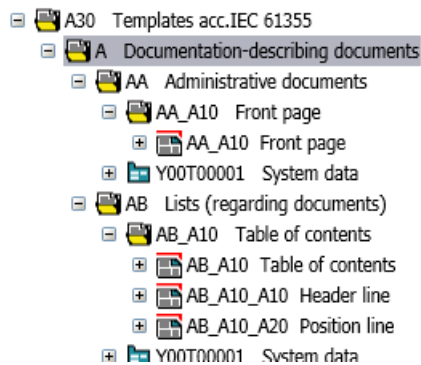
- Templates for interactive reports
The templates for interactive reports consist of master reports from the A10 node and header and footer bars from the A20 node.
- Templates for data sheets (evaluating reports)
Templates for evaluating reports use a master template from the A15 node. The master templates consist of master reports from the A10 node and header and footer bars from the A20 node.

18.2 Naming convention for report templates and document groups

The document templates are created in the structures of the node "A30 Templates acc. to IEC 61355". Each document template is grouped under a document folder which includes the main report, subreports and queries.

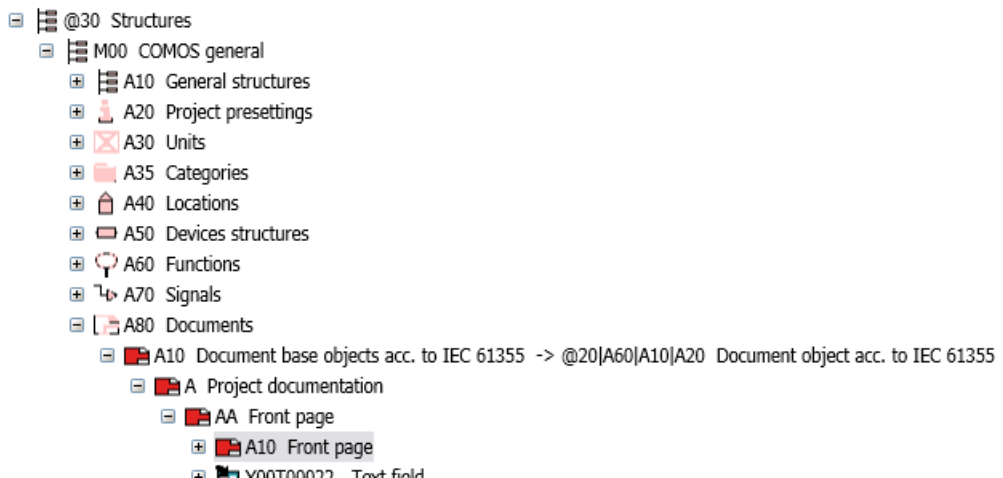
- The name of a document group is made up as follows: The first two letters of the DCC structure + "_" + COMOS iDB count.
- The main report has the same name as the document group.
- The name of a subreport and query is made up as follows: The first two letters of the DCC structure + "_" + COMOS iDB count + COMOS iDB count.
Example: AB_A10_A10 Header. See chapter Using or editing subreports (Page 194).

Example AB_A10:



A subreport can call other subreports. The subordinate subreports in the calling subreport are stored in a separate folder.

In the following figure, you can see the base object for the above report:



The respective name templates in the style of IEC 61355 are set at the base objects of the documents:

General			
Class	<input type="text" value="Document"/>	<input type="button" value="X"/>	Subclass <input type="text" value="(None)"/>
Name	<input type="text" value="A10"/>	<input type="text" value="\A\A\.<999>"/>	<input type="text" value="AA.001"/>
Label	<input type="text"/>	<input type="text"/>	<input type="text"/>
Description	<input type="text" value="Front page"/>		

18.3 Using or editing subreports

Aim

Subreports make it possible to use information numerous times.

Name of the subreport

See chapter Naming convention for report templates and document groups (Page 193).

Storing the subreport

Subreports are stored in a document folder together with the calling main report. The name of the folder corresponds to the name of the main report.

The description is used to indicate that it is a subreport.

Grouping of subreports

Subreports can be grouped in a separate subreport folder under a report folder.

The figure shows the grouping of subreports below the FB_B10_B10 folder:

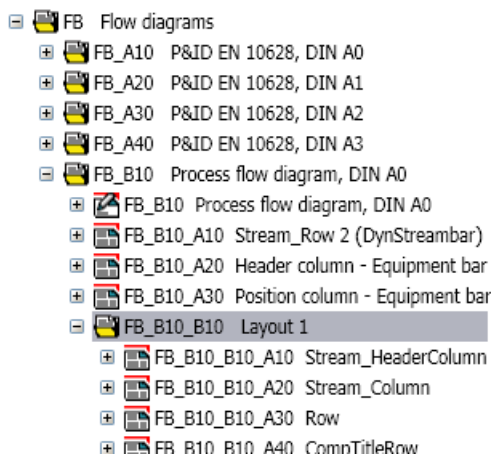


Figure 18-1 Subreport_Folder

The structure in the file system is identical.

Dynamically selected subreports

Various subreports are prepared for dynamic subreports and the user selects one of these in engineering project. The user selects an entry from a list for this. Example:

- Units tab, <Process>: "A70 Process units > FB.001 Process flow diagram"
- Open the properties of the process flow diagram, "Attributes > Material balance" tab
- "Design" list

The structure in the file system is identical.

Restricted multiple use for subreports

If the content of a subreport is required in equal measure by several reports, the following applies:

- Recommendation:
 - Do not use subreports repeatedly.
 - Make a copy of the subreport for each calling report.
 - Copy of the subreport is stored in a folder together with the calling report.
- Multiple use with customer-specific reports
With customer-specific reports, a subreport can be called by several reports. Note that a change to the subreport which is being used multiple times affects all reports which call this subreport.
Create a subreport used multiple times in a "A10 > A00" node.
References (links) can be created under the reports which are calling the subreport, which enables you to track which report has called the subreport.
 - Select the "Copy" command from the context menu of the desired subreport.
 - From the context menu of the calling report, select the "Paste link" command.
The references are stored in parallel to the calling report.

18.4 Object queries for reports

Aim

Lists are displayed for object queries, in which an object query is called from a list field.

Name and storage location

- Create a report template object for each created row.
- Create the report template object below the document group of the main report.
- Report templates can be stored and edited in working layers.

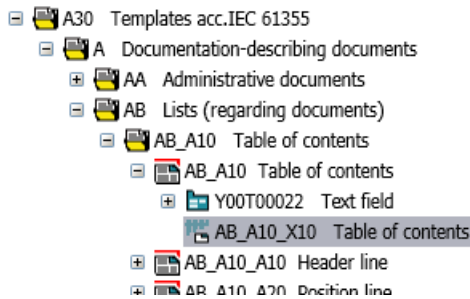
18.4 Object queries for reports

Object queries are created under the main report to which they belong. The name is made up as follows:

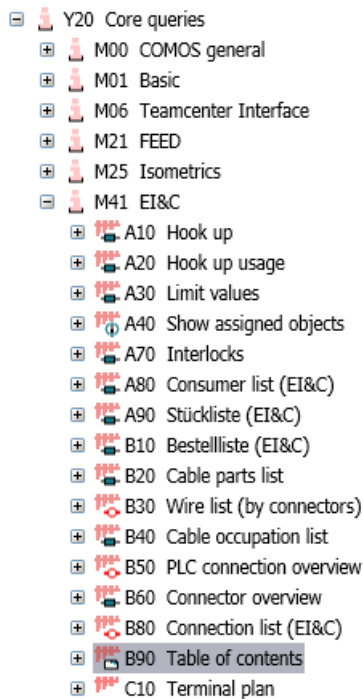
- Name of the main report + "X" (reserved letter for object queries) + count in increments of 10.

Example: AB_A30_X10

Example: AB_A10_X10



The base objects of the object queries are stored in the respective module node under "@20 > A70 > Y20 Core queries":



Configuring object queries

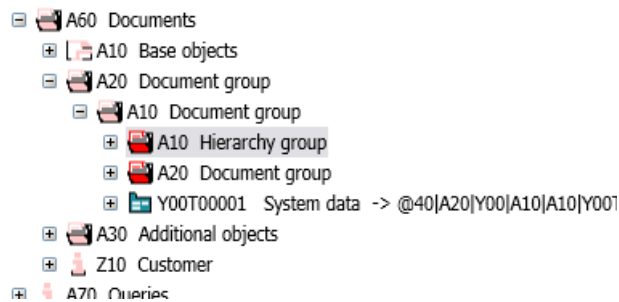
You can find additional information on this topic in the "Administration" manual, keyword "Placing a list".

18.5 Base objects for documents and document groups

18.5.1 @20 > A60 Documents

The base objects for documents are stored in the "@20 > A60 Documents" node. This node is divided into the following areas:

- "A10 Base objects"
The base objects for documents and external file types are stored in this node. Document base objects are inherited in the respective "@30 > Mxx > A80 Documents" node.
- "A20 Document groups"
The base objects for document groups are stored in this node. This node is divided into the following areas:
 - "A10 Hierarchy group"
 - "A20 Document group"

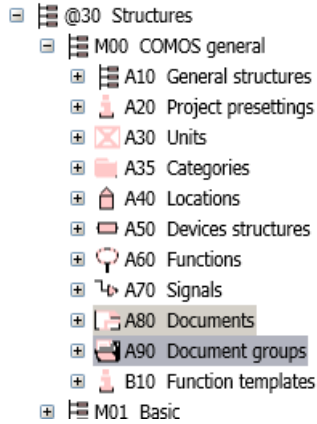


- "A30 Additional objects"
The base objects for additionally required objects are stored in this node. These objects can be used as an elements at the document base objects, for example .
Example: Note object
- "Z10 Customers"
Customer-specific document base objects

18.5.2 Basic rules for documents and document groups in @30 > M00

Overview

The structures for the base objects of documents and document groups are stored in the "@30 M00 COMOS general" node:



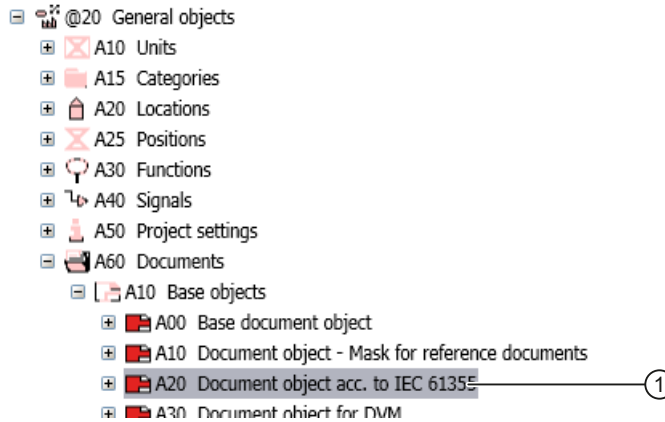
The base objects are responsible for:

- Name masks
- Classification
- Automatic referencing
- Input / storage of revision data
- Insertion of tabs
- Elements

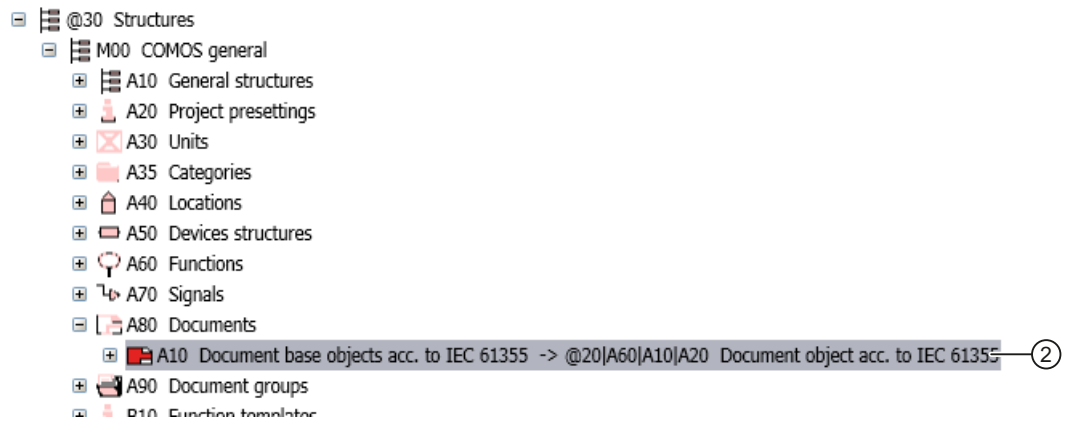
The reports used later in the planning process are mostly controlled by the report templates, and not by these base objects.

18.5.3 @30 > M00 > A80 > A10 Document base objects acc. IEC 61355

The base object (1) for documents in the style of the IEC 61355 standard is cross-inherited from the "@20 > A60 > A10" node to the "@30 > M00 > A80 Documents" node:



All objects are now derived from this base object (1) from the @20 node. The structure in the style of the IEC 61355 standard is now constructed below object (2):



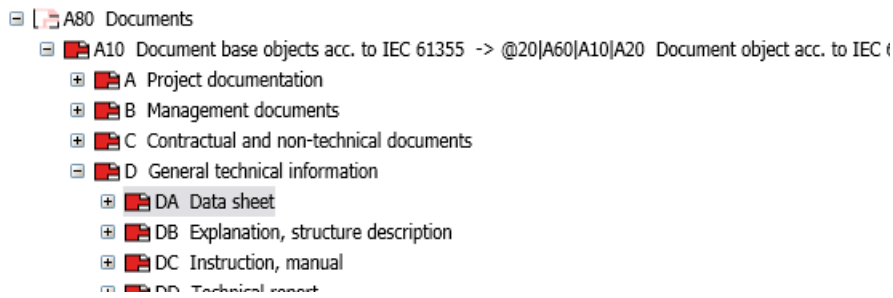
Base objects for specific documents

The document templates for specific base objects are stored below the node according to the DCC keys of IEC61355. The following settings can be made at the base object here.

- Adaptation of the name masks
- Classification key
- Insertion of tabs
- Elements

The special document base objects that are used by the document templates are stored in these structures. The documents are named with the COMOS iDB count:

18.5 Base objects for documents and document groups



If a more specific version of a document base object is required, only one new version may be created below it. Example: There might be another document base object below "A10 Component documentation".

See also

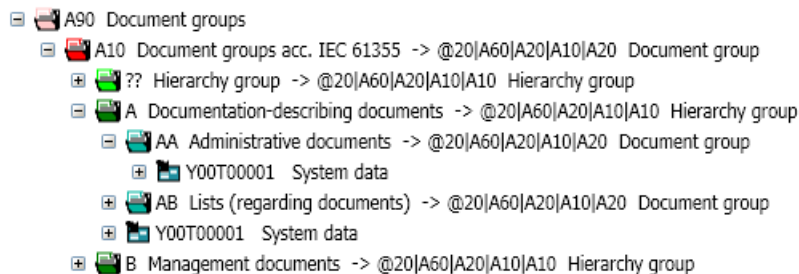
@30 > M00 > A90 Document groups (Page 200)

18.5.4 @30 > M00 > A90 Document groups

The structure below the "@30 > M00 > A90 Document groups" node is formed using the document group base objects from the "@20 > A60 > A20 > A10 Document groups" node.

The DCC structure according to IEC 61355 is displayed as elements from the following two base objects below the "@30 > A60 > A90 > A10" object:

- "@20 > A60 > A20 > A10 > A10 Hierarchy group" Source for the first level below "A90 > A10".
- "@20 > A60 > A20 > A10 > A20 Document group" Source of the second level below "A90 > A10".



The document groups are structured in accordance with the IEC 61355 standard.

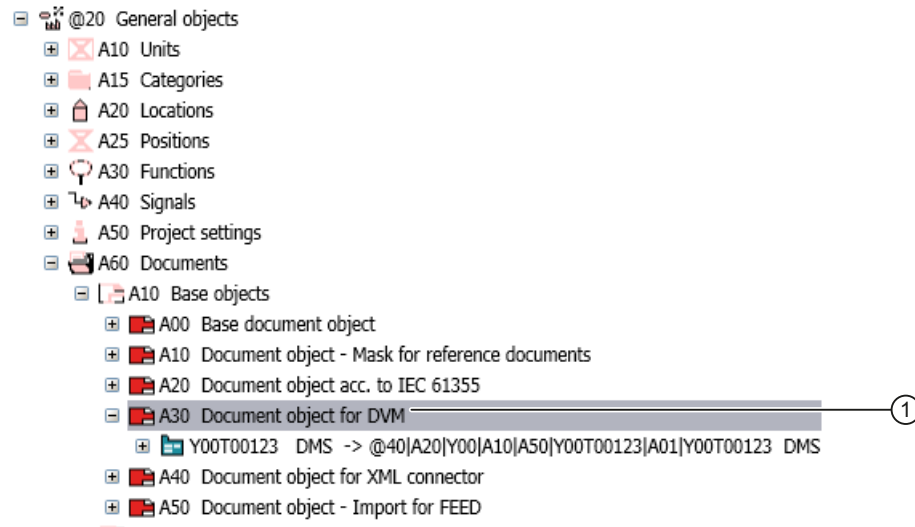
See also

Document structure A90 below the modules (Page 103)

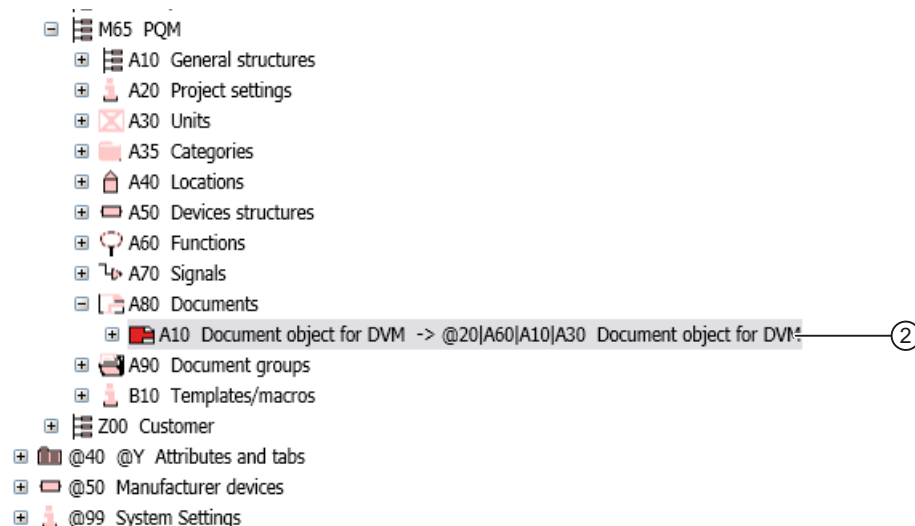
@30 > M00 > A80 > A10 Document base objects acc. IEC 61355 (Page 199)

18.5.5 Managing documents without IEC 61355 labels (@30 > xxx > A80)

Document base objects used for documents which do not belong to the IEC 61355 structure are managed in the relevant module node (@30 > Mxx > A80). The relevant base object is cross-inherited to the module node from the "@20" node:



The specific base objects can then be constructed below this.



18.6 Classification of documents

Classification keys for documents are constructed in a DCC structure below the following node:

- "@99 > A10 > A10 > M00 > A140 Documents"

See section Classification of documents (Page 187).

18.7 Layout recommendations for report templates

Static versus dynamic texts

Use dynamic texts preferably. To do this, you create COMOS attributes in which the texts are entered. In the report template, you display the value of the attribute.

Text fields

- Text fields must be large enough to accommodate all languages.
- Abbreviations are practical for very long attribute descriptions.
- Abbreviations are not permitted in text fields that are not an attribute description.
- The text fields should have an identical text alignment. Default: Left center.
- Text alignment for numbers: Default: Right-aligned
- Select identical display. Default: "Fit oversized texts".

"Text parameters" for texts

- All fields must have a uniform font, font size and a uniform font style (for example, Arial 8 normal).
- "Bold" font type is permitted.
- "Bold" is required for structure elements.
- The "Italic" font type is not permitted. This font type is reserved for revised text.
- Display: Default: "Show texts unchanged"

Lines and borders

- Line width: 0.35 mm to 0.7 mm.

Headers

- Text alignment for headers: Left-aligned
- Row height: 5 mm
- Font in cells: "Arial"
- Font size: "8"

Text fields for min/max values

The "Min/Max" text fields should have the following settings:

- Type: "Item property"
- The suffix "item (0)" for the Min text field.

- The suffix "item (1)" for the Max text field.
- Property: "Value"

Text parameters for revision fields

Revision fields are text fields that display revision texts.

- There is exactly one revision field per attribute.
- Type: "Script"
- Script: "MaxRevisionIndexOf"
- Text alignment: Left-aligned
- Font size: "7"

Page numbers

- The page numbering should be the same for data sheets and lists.
- Use `PageNo` and `MaxPageNo`

Using physical units of measure

- Data sheets
Create fields for physical units. The fields should not be operable.
The default physical unit is read by the attribute.
- Lists
Physical units are selected with queries.

18.8 Using external document files

See Chapter iDB-specific rules for using external files (Page 293).

18.9 Company logos

Principle

All company logos used on a report must have a reference to an external file.

Set this reference as follows:

- Project properties: "General settings > Project"
 - Company logo
The assigned company.
 - Customer logo
The assigning company.

Calling in the script

You get the file name for these logo files via `Project.GetDocumentDirectory + "\" + Project.Spec(ATT_NAME).DisplayValue`, where `ATT_NAME` is "COM01.COM0004" for the company logo and "COM01.COM0010" for the customer logo.

18.10 Displaying AutoCAD drawing in the report

Objective

A dxf file belonging to a component is displayed in a data sheet (an evaluating report) The following steps are required for this:

- Base project: Preparation of the report template
- Engineering project
 - Import of the dxf file below the data sheet
 - Set imported dxf file as reference in an attribute of the pump

Example in the iDB

- Component: "@30 > M00 > A50 > A10 > A20 > A10 > A10 Pump, general"
Data sheet is prepared.
- Data sheet: "A30 > D > DA > DA_D04 > DA_D04 Data sheet pump"
Picture box and script prepared

Base project: Preparation of the report template

1. Open the base project.
2. Open the report template of the evaluating report.
3. Create a picture box.
In the above-indicated example, the picture box is on page 3 of the report template. Use "Page" to navigate to page 3.

4. Open the properties of the picture box.
5. Use the following script:

```
Sub OnCreate()  
  FileName = ""  
  Set objCatalog = Document.owner  
  If Not objCatalog Is Nothing Then  
    Set objAtt = objCatalog.Spec("Y00T00156.Y00A01111")  
    If Not objAtt Is Nothing Then  
      Set objDoc = objAtt.LinkObject  
      If Not objDoc Is Nothing Then  
        FileName = objDoc.FullFilename  
      End If  
    End If  
  End If  
End Sub
```

The script queries the "Y00A01111" attribute of the pump in the engineering project and displays the dxf file entered there.

Creating engineering data

1. Change to an engineering project.
2. Create a pump.
3. Create the data sheet below the pump.
4. Find the dxf file belonging to this pump in the file explorer.
5. Using drag&drop, move the dxf file from the file explorer below the data sheet. The "Choose a document import mode" window appears.
6. Select the "Create copy" option.
7. Open the properties of the pump.
8. Select the "Attributes > General information" tab.
9. Drag&drop the imported dxf file from the Navigator into the "Sketch AutoCad" field.

Additional information

Preparing symbols for AutoCAD export: See chapter Preparing symbols for AutoCAD export (Page 53).

18.11 Creating headers and footers

Headers and footers can be created as a subreport. To store the files, see chapter Using or editing subreports (Page 194). You can find additional information on editing subreports in the report template in the "Administration" manual, keyword "Placing a subreport".

18.12 Interaction of tabs and report templates

Dynamic content in data sheets

Data sheets are divided into areas. Example:

- Header
- General information
- Process data
- Design data
- Footer

Information for the header and footer is created as attribute base object for the report template.

The other areas take their information from tabs. There is a "Process data" tab for the "Process data" report area. If the attributes in the tab are divided into groups, the area on the report template is divided further.

When you display a comment field in the data sheet, create a comment field on each tab.

The order of the tabs at the engineering object must be identical to the order of the fields on the report template. The "Sorting text" field is used in the properties of the tabs for this.

Use of VSUI attributes

Check the VSUI attributes after changes are made to the names of the tabs.

Example: Properties of a function below a position: "Attributes > System data", "VSUI attributes" field

18.13 Drawing type-specific symbol display

See Chapter Drawing type-specific symbol display (Page 50).

18.14 Important information on reports

Create all reports under the corresponding device/equipment.

Select the names for all reports according to the names of the report templates.

Developing scripts for the iDB

19.1 Basic principles of scripts

Definitions

This section contains guidelines and recommendations for creating scripts when using the script language Microsoft Visual Basic Script VBS or VB Script for short).

These guidelines and recommendations apply to scripts within the COMOS iDB, but they also form the basis for scripts that are created as part of customer-specific COMOS-Customizing in COMOS databases.

Any deviations from the guidelines defined in this section are only permissible where customer-specific guidelines are prescribed for scripts within their COMOS database(s).

We make a distinction between the following terms within this document:

- **Method:**
The term "method" is used in this document as a collective term for functions and procedures. Explicit reference is provided where certain conventions only apply to one function or procedure.
- **Function:**
The term "function" is used in this document for methods that provide a return value.
- **Procedure:**
The term "procedure" is used in this document for methods that do not provide a return value.

Personal information in the header

The iDB does not contain example names in scripts and examples. The fields "Changed by:" and "Created by:" are empty or removed. In practice, we recommend that you use this information.

Additional information

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Developing scripts".

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Script editor".

19.2 Script library

19.2.1 Structure of the script library

Exceptions from central storage

All scripts must be stored at a central location so that they can be called by the corresponding usage points.

The following scripts may be exceptions:

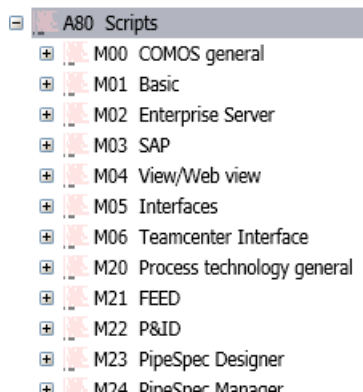
- Script calls
- DLL calls
- Setting constants
- Manageable scripts with only a few rows of instructions
- Scripts in object queries
- Option scripts in reports
- Symbol scripts
- Scripts in standard tables

Root node of the scripts (@20 > A80)

The scripts are saved in script objects below the @20 > A80 node. Sub nodes exist below this node for the different modules and module groups. The script objects are collated under structure objects for 50 script objects in each case under this node for improved manageability.

A script object is a base object with the class set as "Action " and subclass set as "Script".

The structure objects receive their name in accordance with the COMOS iDB count (A10, A20, Ann, etc.):



Up to 9 UserScriptBlocks can be defined at a script object. The method which is called via CallScriptLib at the object is called the main method. The Main Method is saved in UserScriptBlock1 and the sub-methods are saved in the further UserScriptBlocks:

Implemented	Functions
<input checked="" type="checkbox"/>	' UserScriptBlock1: Function GetNewDynamicAtt(PARAMS)
<input checked="" type="checkbox"/>	' UserScriptBlock2: Function GetAttributeBySystemFullName(P/
<input checked="" type="checkbox"/>	' UserScriptBlock3: Function CreateDynamicAttribute(objPoolAI
<input type="checkbox"/>	Sub Connect(Connector)
<input type="checkbox"/>	Sub Disconnect(Connector)
<input type="checkbox"/>	Sub OnDocObjCreate(DocObj)
<input type="checkbox"/>	Sub OnDocObjDelete(DocObj)

Note

Each method is saved in its own UserScriptBlock.

See also

Determining use of a script block (Page 209)

19.2.2 Determining use of a script block**Requirement**

- You are familiar with the structure of the script library. See chapter Structure of the script library (Page 208).

Procedure

- Optional: Update the list of usages. See chapter Using the "Search for script usages" tab (Page 242).
- Select a script object below "@20 > A80 Scripts".
- Open the properties of the script object.
- Select the "Attributes > Script usages" tab.
- Check the columns "SystemFullName" and "Description" in the top table. All other information is only of importance for in-house use at Siemens.
- Optional: Select the "Navigate > Attribute" command in the context menu in the "SystemFullName" column of the top table and check the attribute.

19.2.3 Methods in the script library

In order to identify the name of the method within the "Script" tab this should be added to the first row of commentary separated by a colon and a space. Below is the example "Function GetPagePathLib":

Script example

```
' UserScriptBlock1: Function GetPagePathLib(PARAMS)
Function GetPagePathLib(PARAMS)
' #####
' ###
' ### Header
' ###
' #####
```

Main methods must supply a return value. Main methods must therefore be declared to be functions (Main function).

Public methods are called via CallScriptLib and they only have one transfer parameter, which is called "PARAMS" (see Assigning transfer parameters (Page 214)). Private methods are called within the same script object and they are not suitable for CallScriptLib. Only one public method (main function) should generally be present for each script object.

Note

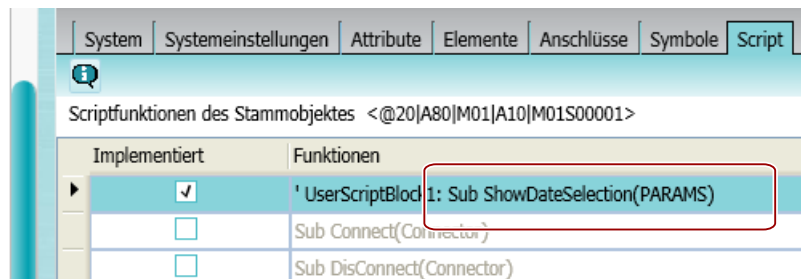
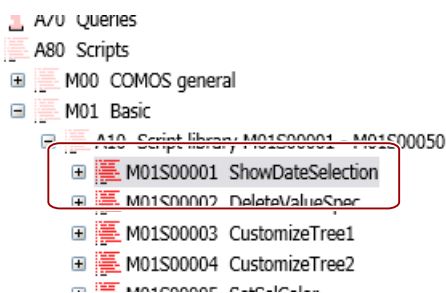
Only functions should be used to process status information on the execution of methods as procedures do not supply a return value.

19.2.4 Naming of script objects

The name of the object is composed as follows:

- Module number <Mxx>
- Letter "S"
- five-digit number

The description corresponds with the name of the main function.



19.3 Structures/ conventions

19.3.1 Overview of the structures and conventions

The source text for a script should be well structured so as to increase legibility and therefore facilitate a user's understanding of the source text and guarantee improved maintainability.

Note

If conventions mention methods then these rules relate both the functions (function) as well as to procedures (sub). This is unless explicit reference is made to the fact that they only apply to one or the other.

Note

The following conventions are derived from Microsoft's programming guidelines from 2011 and they have been adapted to COMOS.

The source text for a script must be structured as follows:

- Header block
- Variable declarations
- Variable initialization
- Implementation

See also

Header block (Page 211)

Variable declaration (Page 213)

Variable initialization (Page 214)

Assigning transfer parameters (Page 214)

19.3.2 Header block

Header block

All methods must start with a header block which must be inserted directly under the first row of the method. The header block contains descriptive information on the functionality of the method and on the change history in implementation.

With this the header block for a method must contain the following information:

- Description of the method
- Information on the method variables

- Information on the method return value
- Information on changes with implementing the method

The description of the method should only contain the purpose of the method (i.e. what the method does, not how it does it – as the implementation may change with time):

Header block

```
' #####
' ###
' ### Description: <description of the method (function/procedure)>
' ### Variables: <variable name>: <description of the variable>
' ###
' ' ### Returns: <description of the return value>
' ### Changed by: <yyyy-mm-dd> - <surname, first name> (<company short name>)
' ### <description of the changes>
' ###
' ' ### Created by: <yyyy-mm-dd> - <surname, first name> (<company short name>)
' ###
' #####

' ### Declaration of the variables:
Dim <variable name>
...

' ' ### Initialization of the variables:
Set <variable name> = ...
...
<variable name> = ...
...

' ' ### Get parameters:
Set <variable name> = PARAMS(<Index>)
...
<variable name> = PARAMS(<Index>)
...
```

When creating the method the header block must be supplemented with the following information on creation:

- Date created
- First name and surname of the creator
A name is not entered in scripts of the iDB.
- Company short name

The header block must be supplemented by the following change information following each change in the method implementation:

- Date changed
- First name and surname of the creator
A name is not entered in scripts of the iDB.
- Company short name
- Description of change

Note

The abbreviation "(iDB)" must be provided as the company short name within the iDB to document that this is the delivery status.

Script example for creation information**' UserScriptBlock1: Function GetPagePathLib(PARAMS)**

```
Function GetPagePathLib(PARAMS)
' #####
' ###
' ### Description:  Creates a text containing page and path of the given device
' ### Variables:   PARAMS(1): Calling device as object
' ###              PARAMS(2): Concerned document as object
' ###              PARAMS(3): Helper DLL as object
' ### Returns:     Reference string (page and path) between device and document
' ### Created by:  2011-11-30 - iDB
' ###
' #####
```

Script example for change information**' UserScriptBlock1: Function GetPagePathLib(PARAMS)**

```
Function GetPagePathLib(PARAMS)
' #####
' ###
' ### Description:  Creates a text containing page and path of the given device
' ### Variables:   PARAMS(1): Calling device as object
' ###              PARAMS(2): Concerned document as object
' ###              PARAMS(3): Helper DLL as object
' ### Returns:     Reference string (page and path) between device and document
' ### Changed by:  2011-12-02 - iDB
' ###              Added checks on Nothing
' ### Created by:  2011-11-30 - iDB
' ###
' #####
```

19.3.3 Variable declaration

The variable declaration should join the header block directly with a space separating the two.

Variable declaration

```
' ### Declaration of the variables:
  Dim <variable name>
  ...
```

Inserting just one declaration is recommended for each row. Each additional declaration should be inserted accordingly in the subsequent rows.

19.3.4 Variable initialization

The variable initialization should join the variable declaration directly with a space separating the two.

Variable initialization

```
' ### Initialization of the variables:
  Set <variable name> = ...
  ...
  <variable name> = ...
  ...
```

19.3.5 Assigning transfer parameters

All main methods in the script objects of the central script library must feature one transfer parameter precisely known as "PARAMS" in the declaration. This is a field which contains all transfer parameters which have been transferred when calling the method.

' UserScriptBlock1: Function GetPagePathLib(PARAMS)

```
Function GetPagePathLib(PARAMS)
' #####
' ###
' ### Description:  Creates a text containing page and path of the given device
' ### Variables:   PARAMS(1): Calling device as object
' ###              PARAMS(2): Concerned document as object
' ###              PARAMS(3): Helper DLL as object
' ### Returns:    Reference string (page and path) between device and document
' ### Created by:  2011-11-30 - iDB
' ###
' #####
```

These transfer parameters should first of all be assigned to a previously declared variable before they are used:

Assignment

```
' ### Get parameters:
  Set <variable name> = PARAMS(<Index>)
  ...
  <variable name> = PARAMS(<Index>)
  ...
```

19.3.6 Return values of functions

Functions should always supply an explicitly defined return value.

Background: Standard return values which have been used with no explicit return value assigned should not be accessed.

If no value or object were to be assigned to the function name within the source text then standard return values would be used which are defined as follows:

- A numerical function returns a 0
- A character string function returns a string of zero length ("")
- A function that returns an object returns "Nothing".

In order to return a value or an object explicitly from a function then a value or an object is assigned to the function name (by Set <Function name> = ...). Within the function there can be any desired number of these assignments at any desired points in the source text.

In the event of an error the point up to which a function was completed or what the result of calling the function was can both be determined via the return value for a function.

19.3.7 Calling central scripts

Call of UserScript blocks

The scripts that are located in the UserScript blocks of the script objects are called via a CallScriptLib call at the usage points.

Example: Script call to a function event

Example of a script call to a function event at the object

```
Function GetConnectorSpecification(objDevice, strConnectorName, strAttName)
' #####
' ###
' ### Description: Gives the attribute object of a connection depending on
' ### its linked attribute.
' ' ### Variables: objDevice: Owner of the connected connector
' ### strConnectorName: Name of the connected connector
' ### strAttName: Name of the linked attribute
' ### Returns: Attribute object of the connection.
' ' ### Created by: 2012-01-30 - iDB
' ###
' #####

' ### Get corresponding connector specification
Set GetConnectorSpecification = Project.Workset.Lib.CallScriptLib( _
    "@20|A80|M22|A10|M22S00002", _
    "GetConnectorLinkedAttribute3Way", _
    objDevice, strConnectorName, _
    strAttName, ThisObj)

End Function
```

Example: Scripts in UserScript block

Example of a calling script in the UserScript block at a script object

' UserScriptBlock1: Function GetPagePathLib(PARAMS)

```

Function GetPagePathLib(PARAMS)
' #####
' ###
' ### Description:  Creates a text containing page and path of the given device
' ### Variables:   PARAMS(1): Calling device as object
' ###              PARAMS(2): Concerned document as object
' ###              PARAMS(3): Helper DLL as object
' ### Returns:    Reference string (page and path) between device and document
' ### Created by:  2011-11-30 - iDB
' ###
' #####

' ### Declaration of the variables:
Dim colObjDocObjects
Dim intDocObjCnt
Dim objDev
Dim objDoc
Dim objDocCurrent
Dim objDocObjCurrent
Dim strReference

' ### Initialization of the variables:
intDocObjCnt = 0
strReference = ""

' ### Get parameters:
Set objDev = PARAMS(1)
Set objDoc = PARAMS(2)
Set objGlbDll = PARAMS(3)

' ### Get all DocObjs
Set colObjDocObjects = objDev.BackPointerDocObjs

If Not colObjDocObjects Is Nothing Then
    intDocObjCnt = colObjDocObjects.Count

    For intI = 1 To intDocObjCnt

        Set objDocObjCurrent = colObjDocObjects.Item(intI)
        Set objDocCurrent = objDocObjCurrent.owner

        ' ### Check SymbolType of the corresponding document
        If objDocCurrent.SymbolType = "M41_P2" Then
            strReference = objGlbDll.GU_BuildReferenceString(objDocObjCurrent, _
                objDocCurrent, 0)
        End If

    Next 'intI

End If

' ### Return reference string
Set GetPagePathLib = strReference

End Function

```

19.3.8 Comments in the script

Syntax of the comments

The general comments in the script begin as follows:

```
' ### ...
```

The source text for a method should indeed be documented and this can be achieved by way of comments. All comments must be in English. Care should be taken here that comments are informative but are not too lengthy.

This is how comments are provided on the general sequence for the method, in particular for each start of large sections of the method (e.g. "Building the object collection", "Processing all objects in the object collection"). Furthermore, comments should be provided for particularly complex portions of the source text (e.g. "This loop processes all rows in the table while the sub-loop enters the attribute values of the devices into the right columns").

Commenting on each individual portion of the source text must be avoided.

As well as the addition of change information to the header block, the source text where the changes took place must also be commented on as follows:

Commenting on a change at a single instruction row:

At this point, the comment row inserted inside the header block, which contains the change date, the first and last name of the person who made the change, and the company short name, is copied in front of the instruction row which is to be changed.

The change is then described in a subsequent comment row.

Example 1

```
' ### Changed by: <yyyy-mm-dd> - <surname, first name> (<company short name>)  
' ### <description of the change>  
  <source code>
```

Commenting on a change at a block of instruction rows:

At this point, the comment row inserted inside the header block, which contains the change date, the first and last name of the person who made the change, and the company short name, is copied in front of the first instruction row which is to be changed and after the last instruction row which is to be changed.

The words "Start" and "End" are then added to the "Changed by:" text to show the beginning and end of the change.

The change is then described in one of the comment rows following the first change comment row.

Example 2

```
' ### Start Changed by: <yyyy-mm-dd> - <surname, first name> (<company short name>)
' ### <description of the changes>
  <source code>
  <source code>
  <source code>
' ### End Changed by: <yyyy-mm-dd> - <surname, first name> (<company short name>)
```

19.3.9 Formatting the script

Row length and breaks

Rows should not be so long that they require a horizontal scroll bar. Rows should generally have a maximum of 80 characters so that new lines and indentations that are motivated semantically are also retained in the printout (limited to the safe printable area).

If a row is too long it should be broken up in accordance with the following rules:

- Break following a comma (e.g. with parameter lists)
- Break after operator
- The text in the new row should be indented in such a way that similar source text points are at the same level in both rows

It must be ensured that any instructions which cover multiple rows must be separated by an underscore (with a preceding space in all cases). This demonstrates that the instruction continues in the subsequent rows, as VBScript separates commands with spaces and not with special characters such as a semi-colon (;).

Line breaks – example

```
Function ShowSampleOfMultiLineCommands(intInputValue1, _
                                       intInputValue2, _
                                       intInputValue3, _
                                       intInputValue4)

    intOutputValue = intInputValue1 ^4 + _
                    4 * intInputValue2 ^3 + _
                    12 * intInputValue3 ^2 + _
                    24 * intInputValue4

    ShowSampleOfMultiLineCommands = intOutputValue

End Function
```

Note

Error messages with script errors may point to an incorrect row as this type of row continuation is interpreted as a row by the Interpreter.

Indenting the source text

Blocks of source text (contents of a method, a loop, an If...Then...Else instruction, etc.) must be indented. This makes it clear to the reader which parts of the source text belong to which control structures (loops/branchings). Comments should be indented in exactly the same way as the source text which is being commented on. The standard indentation depth is one tab. A tab increment of 2 is recommended for indentations using the tab key.

Indentation – example

```
Sub ShowIndentationSample(intInputValue)

    blnStop = False

    While (blnStop = False)

        intInputValue = intInputValue / 2

        If intInputValue < 10 Then
            Output "Done"
            blnStop = True
        Else
            Output "Not done yet: " & intInputValue
        End If
    Wend

End Sub
```

Blank spaces

Using blank spaces and line breaks considerably improves the legibility of the source text. Blank spaces are created using the space bar and tab key, line breaks using the enter key. Blank spaces and line breaks help to create a meaningful structure and facilitate the reader's understanding of the source text.

Blank spaces should be used:

- Between a key word and the left bracket which belongs to it. This applies to If, Elseif, For, While, etc.
- Between the right bracket and the key word following it
- Before and after an operator

Blank spaces should not be used:

- Between a method name and the following left bracket for the arguments
- After a left bracket
- Before a right bracket

Examples

```
intTotalQuantity = intTotalQuantity + intCurrentQuantity
If (intTotalQuantity < 10) Then Exit Sub
Function GetObject(PARAMS)
```

19.3.10 Use of parentheses for If...Then...Else instructions

If there are multiple operators in a printout, each part is evaluated in a pre-defined sequence. This evaluation sequence is known as operator precedence.

In order to simplify the readability of the source text in control structures in the form of If...Then...Else instructions, the operator precedence should be bypassed by enclosing individual operations in brackets. Operations in brackets are always executed before operations outside of brackets. However, the normal operator precedence applies within the brackets.

Examples

```
If (dblNominalCurrent * dblConcurrencyFactor) < dblMaximumCurrent Then
    ...
End If

dblTotalWeight = dblUnitWeight * intQuantity
dblAllowedWeight = dblMaximumWeight - dblEmptyWeight

If (dblTotalWeight > dblAllowedWeight) OR _
    (dblTotalWeight > dblMaximumTensileLoad) Then
    ...
End If
```

19.3.11 Use of variants for If...Then...Else instructions

Different syntax variants (single and multiple rows) are possible for control structures in the form of If...Then...Else instructions. The following syntax should be used – depending on the usage case – in order to guarantee manageability and to be able to react with greater flexibility to subsequent source text changes:

Single-row variant

If in the source text additional source text follows on from the key word "Then" in the same source text row (excluding comments) then this is a single-row If instruction.

The single-row variant should only then be used if an Exit instruction follows the key word "Then" which directly terminates any further execution of a method or loop.

Examples

```
If intFuseCnt = 0 Then Exit Function

If objDevFuse Is Nothing Then Exit Function

If Not objDevFuse.IsSuccessorOf(objDevCabinet) Then Exit Function

If Not ((intVoltageLevel = 230) AND (intFrequency = 50)) Then Exit Function
```

Multiple-row variant (block variant)

If in the source text no more additional source text follows on from the key word "Then" in the same source text row (excluding comments) then this is a multiple-row If instruction which must be terminated with an End If instruction.

19.4 Naming

The multiple-row variant should always then be used if further instructions follow the key word "Then" which do not directly terminate further execution of a method or loop.

Examples

```
If (intVoltageLevel = 230) AND (intFrequency = 50) Then
    lngFuseSubSymbol = 1
ElseIf (intVoltageLevel = 400) AND (intFrequency = 50) Then
    lngFuseSubSymbol = 2
Else
    lngFuseSubSymbol = 0
End If
```

19.3.12 Merging of character strings

Merging character strings is known as concatenation and it should generally be implemented with a "&" instead of a "+". When the operator "+" is used VB Script converts numerical values to a plus.

19.4 Naming

19.4.1 Naming for methods

The name for a method should be meaningful so that the purpose can be deduced from the name. Methods are given English names.

Note

Method names may not include any spaces.

The method name should begin with a verb as they execute an action. The first letter for the method name must be written in upper case. If the method name consists of individual words written together then the relevant first letter of a word must be written in upper case (Camel Case).

Example:

```
CloseDialog, SearchPositionsBelowUnit
```

19.4.2 Naming for variable names

The name for a variable should be meaningful so that the purpose can be deduced from the name. With this the variable name must start with a prefix as described under variable types. Names for variables must be in English. The first letter for the variable name must be written in lower case. If the variable name consists of individual words written together then the relevant first letter of a word must be written in upper case (Camel Case).

As is the case with methods, all abbreviations that are used must be consistent in the overall script (and in all script where possible).

Furthermore a series of prefixes must be used to identify the variable type and other variable properties.

Exception: A simple intI, intJ, intK etc. suffices for simple count variables such as those used in a For loop.

See also

Variable types (Page 223)

Other properties (Page 224)

COMOS object types (Page 224)

19.4.3 Variable types

The following prefixes must be used to identify the variable type:

Type	Prefix	Example
Boolean	bln	blnFound
Byte	byt	bytColour
Dictionary (key word pair)	dic	dicColors
Date (Time)	dtm	dtmStart
Double	dbl	dblTolerance
Error	err	errOrderNum
Float	flt	fltTemperature
Integer	int	intQuantity
Long	lng	lngDistance
Single	sng	sngAverage
String	str	strFirstName
Object	obj	objCurrent

Note

Count variables in loops must also follow these rules.

The following thus applies: For intK = 1 To 8

See also

Naming for variable names (Page 222)

19.4.4 Other properties

The following prefixes are used for other variable properties. Where possible they should be used in addition to the prefixes for variable types.

Type	Prefix	Example
Array (Field)	arr	arrStrNames
Collection (Collection)	col	colObjFuses
Constant (Constant)	const	constIntDefaultValue

Global variables must be used with caution. Where global variables are required they must be declared in the "ScriptBlockParameter" Scriptblock.

See also

Naming for variable names (Page 222)

19.4.5 COMOS object types

The following prefixes must be used in addition to the variable type prefix "obj" in order to identify COMOS object types:

COMOS object type	Prefix	Example
Attributes (Attribute)	Att	objAttTemperature
CDevice (Base object)	CDev	objCDevPump
CDocument (Report template)	CDoc	objCDocPID
Connector (Connector)	Con	objConSLI
Device (Engineering object)	Dev	objDevCurrent
Document (Document)	Doc	objDocPID
Query (Object query)	Qry	objQryPartslist
Standard table (Standard table)	StdT	objStdTBridgeType
Standard value (Standard table value)	StdV	objStdVCableBridge

COMOS object type	Prefix	Example
Tab (Tab)	Tab	objTabTechnicalData
Working overlay (Working layer)	WoO	objWoOReleaseOverlay

See also

Naming for variable names (Page 222)

19.5 Incorporating COM components

19.5.1 Overview of the technology in the COM components

COM components are understood as a collection of one or more COM classes. A COM component is generally a binary file (.dll, .ocx, .exe). However, there are also COM components in the form of source text files known as Windows Script Components.

Each COM class provides an interface which following successful instantiation can be used to implement the methods and properties offered by the COM classes.

The "Component Object Model" ("COM" for short) is not location-dependent, meaning that the COM components can be accessed independently of their actual location.

Once a COM component has been identified by the operating system it can be incorporated within the source text with the following instruction:

```
Set <ObjectInstanceName> = CreateObject (<ObjectClassName>[,  
<ServerName>])
```

Note

The method "CreateObject" returns an object reference which must be assigned to a variable.

Only COM-enabled components can be incorporated with the method "CreateObject". COM-enabled components are COM components and .NET components which are defined as COM visible.

19.5.2 Declaration components of the method "CreateObject"

- **<ObjectClassName>:**
The ObjectClassName of the object to be created is made up of the <ComponentName>.<ClassName>.
- **<ServerName>:**
Optional parameter. The name of the network server on which the object is to be created. If <ServerName> is an empty string (""), the local computer is used.

19.5.3 Memory storage management or cleaning with incorporated COM components

The COMOS Script Engine should be supported with memory storage management or cleaning where instances of COM classes have been produced from incorporated COM components.

If instances of COM classes have been produced from incorporated COM components then the memory assigned to these instances should be released where the instances are no longer being used.

The following instruction must be used to release the memory assigned:

```
Set <ObjectInstanceName> = Nothing
```

NOTICE
Instances
The instance of the COM components must be cleaned before a method is left.

19.6 Avoiding runtime errors

Procedure

The following steps help to avoid runtime errors:

- **Validity check of input variables**
All input variables transferred to the method using transfer parameters should be checked for validity. This also ensures that, during further processing of the input variables, no runtime errors can occur due to missing input variable types or input variable values.
- **Checking for the existence of objects**
Before accessing the properties of an object (for example, the description of an attribute or the name of a device), a check must be performed first to determine whether the object exists. Only then is access permitted.

Example:

```
If Not objDevThermometer Is Nothing Then
  Set objAttTemperature =
objDevThermometer.spec("Y00T00005.Y00A07285")
  If Not objAttTemperature Is Nothing Then
    intTemperature = objAttTemperature.value
  End If
End If
```

19.7 Improve script performance

The following fundamental rules should be observed in order to improve script performance and thereby reduce the runtimes.

- Define objects which are used frequently as variables beforehand

Example:

```
Set objWorkset = Project.Workset
```

- Define a collection as a variable before accessing its elements

Example:

```
Set colObjDevTerminals = objDevTerminalStrip.EnObs("E")
```

- Define the number of elements in a collection before iteration as a variable

Example:

```
intFuseCnt = colObjDevFuses.Count  
For intI = 1 To intFuseCnt  
    ...  
Next 'intI
```

- Define the current object as a variable with multiple access to different properties of an object in iteration (For loops, etc.)

Example:

```
For intI = 1 To intFuseCnt  
    Set objDevActualFuse = colObjDevFuses.Item(intI)  
    Set objAttFrequency = objDevActualFuse.Spec("Y00T00003.Y00A00363")  
    Set objAttManufacturer = objDevActualFuse.Spec(Y00T00025.Y00A00370")  
  
    If Not objAttFrequency Is Nothing Then  
        intFrequency = objAttFrequency.Value  
    End If  
  
    If Not objAttManufacturer Is Nothing Then  
        strManufacturer = objAttManufacturer.Value  
    End If  
  
Next 'intI
```

- Define attributes as a variable for a check for their existence

Example:

```
Set objAttNominalVoltage = objDevFuse.Spec("Y00T00003.Y00A00837")  
  
If Not objAttNominalVoltage Is Nothing Then  
    ...  
End If
```

- Define a tab as a variable prior to querying multiple attributes of a tab

Example:

```
Set objAttTabTechnicalData = objDevFuse.Spec("Y00T00003")
Set objAttFrequency = objAttTabTechnicalData.Spec("Y00A00363")
Set objAttNominalVoltage = objAttTabTechnicalData.Spec("Y00A00837")
Set objAttNominalCurrent = objAttTabTechnicalData.Spec("Y00A00838")
```

- Limit the search quantity as far as possible (define classes and base objects and leave out recurrent search procedures) and do not generally use the Devices Collection when searching for certain engineering objects within a node.

Examples:

Instead of a search using the methods "Devices", use methods such as OwnElements, EnObs, ScanDevices, ScanDevicesWithXObject, Scan etc. Stop classes should be defined where possible when using recurrent searches.

- Where possible execute comparisons based on object and not on string operations.

Examples:

```
<Object A> Is <Object B>
<Object A>.IsSuccessorFrom(<Object B>)
<Object A>.IsInheritSuccessorFrom(<Object B>)
```

19.8 Localizing texts in scripts with NLS

19.8.1 Overview of the use of NLS texts

Aim

You can create text and messages in COMOS using script; for example, you can create notes for the user by means of "Messagebox". With the technique described below, you can create such text and messages in more than one language. To do so, you use NLS (National Language Support).

Procedure

Multilingual text is administered as follows:

1. The text to be translated is collected in standard tables. Each separate text to be translated has its own entry.
See section *Administering NLS texts in standard tables* (Page 229).
2. The entries in the standard table are translated.
See section *Administering NLS texts in standard tables* (Page 229).
3. An entry is accessed from the standard table in the script and it is determined which translation is to be used.
The following options are available:
 - Opening in the script delivers the text in the language that is currently set as the project language (database language).
 - Opening in the script delivers the text in the language that is currently set as the interface language.
 - Opening in the script delivers the text in the language that is found by means of an ID provided.See section *Functions for localizing texts* (Page 232).

See also

Call example of an NLS text (Page 234)

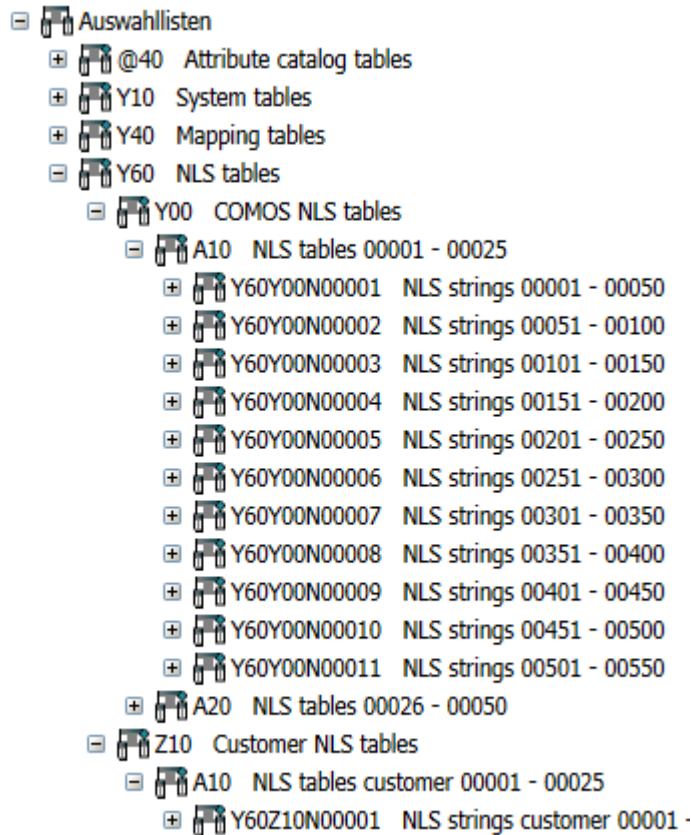
19.8.2 Administering NLS texts in standard tables

Requirement

- You are familiar with the overview of using NLS texts.
See chapter *Overview of the use of NLS texts* (Page 228).

Storage and assigning names

The texts are administered in standard tables under the node "Y60 NLS tables":



It is essential to observe the following structure:

Y60 NLS tables: Shared folder for all NLS tables

- Y00: Branch with the text supplied. These objects cannot be edited.
 - "A10 NLS tables 00001 - 00025": First sub-folder for up to 25 standard tables
 - "Y60Y00N00001 NLS Text 00001 - 00050": First standard table with up to 50 entries
- "Z10 Customer NLS tables": Branch with the customer-specific text.
 - "A10 NLS tables customer 00001 - 00025": First customer sub-folder for up to 25 standard tables
 - "Y60Z10N00001 NLS Text Customer 00001 - 00050": First customer standard table with up to 50 entries

The standard tables with the text are identified by their name. Every standard table under "Y60 NLS tables" must therefore have a unique name. The following name conventions apply:

- Standard tables
 - The structure is used as a prefix in the name.
 - COMOS standard tables
 - Prefix for the folder "Y60" + subfolder "Y00" + "N" + five-digit consecutive number starting at "00001"
 - Example: "Y60Y00N00001"
 - Customer standard table
 - Prefix for the folder "Y60" + subfolder "Z10" + "N" + five-digit consecutive number starting at "00001"
 - Example: "Y60Z10N00001"
- Standard table values
 - Standard name: Capital letter "A" + three-digit number.
 - Example: A010
 - The names are incremented in tens, with only 50 standard table values permitted for each standard table.
 - The text to be translated is entered in the "Description" column in multiple languages.
 - Optional: Customer-specific entries in a COMOS standard table:
 - Prefix "Z" followed by a number with at least two numerical values or a "_Z" suffix
 - Value 1: The value in this column is the attribute value (as compared to the display value). You need to ensure that this entry is unique and that the numerical value is correct.

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Administration of selection lists".

Translate NLS texts

1. Open the function "Extra > Translation > Bulk translation".
2. Drag the NLS table from the Navigator into the "Start object" field.
3. Set the "Search technique" option to "Recursive".
 - The list contains the entries from the standard table.
4. Edit the entries in the language columns "German", "English (USA)", and so on.
5. Recommendation: Value 1 should be unique and not have any special meaning.
 - Value 1 is internally a buffer to store the value at the attribute. The correct content for the attribute (e.g. "Yes") is stored in value 2. This way you can change value 2 without losing the selected information (entries) of the standard tables.
6. Recommendation: Each standard table should contain a "Not defined" value. The first name of the list is "A000" and value 1 should be "0".
7. Confirm your entries.

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Manual translation with bulk translation".

See also

Y60 NLS tables (Page 169)

19.8.3 Functions for localizing texts

Requirement

- You are familiar with the overview of using NLS texts. See section Overview of the use of NLS texts (Page 228).

Call

An NLS text is created with one of these functions:

- `GetInternationalTextFromNLSByProjectLanguage`
- `GetInternationalTextFromNLSByGUILanguage`
- `GetInternationalTextFromNLSByLanguageID`

Explanations for function `GetInternationalTextFromNLSByProjectLanguage`

```
Public Function GetInternationalTextFromNLSByProjectLanguage (ByVal  
NLSStdTableName As String, ByVal NLSTextID As String, ParamArray  
PlaceholderValues()) As String
```

- **Parameter 1**
The first parameter provides the name of the associated NLS table. (The system full name of the standard table is not specified here.)
- **Parameter 2**
The second parameter provides the name of the associated standard table value.
- **Parameter 3**
The third parameter provides a list of strings that replace placeholders within the tests. An example is given later.

The result is the selected text in the project language. If the text was not translated into this language, then the process for all translatable COMOS texts applies.

Explanations for function `GetInternationalTextFromNLSByGUILanguage`

```
Public Function GetInternationalTextFromNLSByGUILanguage (ByVal  
NLSStdTableName As String, ByVal NLSTextID As String, ParamArray  
PlaceholderValues()) As String
```

- **Parameter 1**
The first parameter provides the name of the associated NLS table. (The system full name of the standard table is not specified here.)
- **Parameter 2**
The second parameter provides the name of the associated standard table value.
- **Parameter 3**
The third parameter provides a list of strings that replace placeholders within the tests. An example is given later.

The result is the selected text in the language of the user interface. If the text was not translated into this language, then the process for all translatable COMOS texts applies.

Explanations for function `GetInternationalTextFromNLSByLanguageID`

```
Public Function GetInternationalTextFromNLSByLanguageID (ByVal
NLSStdTableName As String, ByVal NLSTextID As String, ByVal
LanguageID As Integer, ParamArray PlaceholderValues()) As String
```

- **Parameter 1**
The first parameter provides the name of the associated NLS table. (The system full name of the standard table is not specified here.)
- **Parameter 2**
The second parameter provides the name of the associated standard table value.
- **Parameter 3**
The third parameter provides the number of the required language as string. (Example: "1" for German and "2" for English.)
- **Parameter 4**
The fourth parameter provides a list of strings that replace placeholders within the tests. An example is given later.

The result is the selected text in the language of the specified number. If the text was not translated into this language, then the process for all translatable COMOS texts applies.

Placeholders for text using "PlaceholderValues"

Explanation for optional parameter "ParamArray PlaceholderValues()":

The placeholders %0%, %1%, ..., %n% can be entered in texts for NLS texts. The values that are later specified in these placeholders are administered as a list of strings.

NLS text in the description at the standard table value "A010" at the standard table "Y60Z10N00001":

```
"Name: %0% Systemfullname: %1%"
```

The following call in the object debugger for any object "a":

Output

```
Project.Workset.Lib.GetInternationalTextFromNLSByProjectLanguage ("Y6
0Z10N00001", "A010", a.Name, a.SystemFullName)
```

"a" is the object in this case: @20|A80|M00|A10|M00S00001.

The call has the following result:

```
"Name: M00S00001 Systemfullname: @20|A80|M00|A10|M00S00001"
```

This method is defined this way for all three functions for the last parameter.

Example

```
Project.Workset.Lib.GetInternationalTextFromNLSByProjectLanguage ("<N
LSStdTableName>", "<NLSTextID>")
```

Example for a MessageBox:

```
strBox =
```

```
Project.Workset.Lib.GetInternationalTextFromNLSByProjectLanguage ("<N
```

```
LSStdTableName>", "<NLSTextID-MsgBox>")
strHeader =
Project.Workset.Lib.GetInternationalTextFromNLSByProjectLanguage("<N
LSStdTableName>", "<NLSTextID-MsgBoxHeader>")
Project.Workset.Lib.ComosMsgBox strBox, , strHeader
```

Comments and parameters for example:

- <NLSStdTableName>:
Name of the NLS standard table (e.g. "Y60Y00N00001")
- <NLSTextID-MsgBox>:
Name of the message text to be addressed (e.g. "A010") This value is output in the Messagebox text area.
- <NLSTextID-MsgBoxHeader>:
Name of the message text to be addressed (e.g. "A020") This value is output in the Messagebox header.

19.8.4 Call example of an NLS text

Requirement

- You are familiar with the overview of using NLS texts.
See section Overview of the use of NLS texts (Page 228).

Examples

Call examples from the object debugger:

- `Project.Workset.Lib.GetInternationalTextFromNLSByProjectLanguage("Y60Y00N00001", "A030")`
- `Project.Workset.Lib.GetInternationalTextFromNLSByGUILanguage("Y60Z10N90001", "A010", a.SystemFullName)`
(Note: A matching NLS text must first be created for this example in the database and "a" must be a matching object.)
- `Project.Workset.Lib.GetInternationalTextFromNLSByLanguageID("Y60Y00N00001", "A030", "1")`
In this example, the value "1" corresponds to the first language column, i.e. German.

19.9 Prevent pop-up window

Any parts of the program which use pop-up windows should only be used where absolutely necessary. This includes message boxes (MsgBox), input boxes (InputBox), progress bars (ProgressBar). These control elements interrupt the program flow and may therefore lead to errors in COMOS.

There is a separate function for message boxes in COMOS which should be used where a pop-up message is unavoidable.

The COMOS-specific function is called with the `Project.Workset.Lib.ComosMsgBox` command, with the transfer parameters being identical to the classic "MsgBox" function.

Contrary to the classic "MsgBox" function the "ComosMsgBox" can be suppressed by the COMOS Enterprise Server (execution in silent mode) and it does not therefore wait for an answer from the user, e.g. when importing files.

19.10 Logging script actions

When you write scripts, you have the following options for logging the sequence of the script:

"OutputDebugString" method

Outputs can be created via script from COMOS using the following instruction:

```
Project.Workset.Lib.Output(<Text to be output>)
```

A program such as the Microsoft program "DBMon.exe" (DBMon - Debugging Monitor), which is available in the "Current\BIN" subdirectory of the COMOS installation, can be used to view all logged execution information.

The messages displayed using DBMon.exe comprise both local error messages as well as information exchanged among other COMOS users via CVS (Cache Validation Service). In general, all messages which are produced by a program using the "OutputDebugString" method are displayed by DBMon.exe.

The "Output" method to be used in COMOS for outputting relevant messages produces an output via the "OutputDebugString" method defined inside kernel32.dll.

"Object test" tab

The "Object test" tool can be used to provide the user with execution information at the COMOS user interface stage.

The user can make outputs and inputs within the "Object Test" tool on a certain object using the following instructions:

Error outputs

```
<Object>.AddToErrorObjects(<ErrorCode of the ErrorObject>, <Error  
description of the ErrorObject>)
```

Additional handling of ErrorObjects is described in the COMOS help guide.

Note

Management of ErrorObjects that are displayed in the "Object Test" tool is implemented per user and per project.

The user cannot influence the "degree of problem" property for the corresponding input in the Object Test.

The user cannot influence the ErrorObjectMode for the corresponding input in the Object Test tool. The inputs created by the user using the methods stated above are always given the definition "3 - external" by the system for the ErrorObjectMode

19.11 Information about the Object Debugger

The object debugger is used to act directly on objects by means of a script. You can, for example, use the object debugger to test scripts or query the available methods and properties of objects.

Call: "Extra > Object debugger" menu

You can find additional information on this topic in the "Administration" manual, keyword "Object debugger".

19.12 Using hard-coded base objects in scripts

Use of the "A90 Hardcoding" object

See chapter "A90 Hardcoding": Using base data in script and code (Page 155).

Outdated: Hardcoding for block base objects by means of standard table

"@30 > M40 > A50 > A10 > A10 > A10 Automation objects"

In the current versions of the iDB, the standard table has been moved to the node "Y99 Obsolete standard tables".

The hard coding for these base objects is managed in the standard table "Y40 > M00 > A10 > Y40M00N00001 > Y40M00N00001M40".

This standard table contains the changeable Name and the SystemFullName in each case. The "-" character is used as the delimiter.

The SystemFullName of a base object is retrieved from the standard table in which the value in the "Name" column is passed to the `HardcodingMapper` function (here with the example value "A010"):

```
Set mapper =  
CreateObject("Comos.SolidBaseUtilities.HardcodingMapper")  
Dim mapping  
mapping = mapper.GetMappingFromStandardTable("Y40|M00|A10|  
Y40M00N00001|Y40M00N00001M40", "A010")
```

19.13 iDB script editing

19.13.1 Opening script editing

The script editing exports and imports iDB script blocks. These functions enable you to edit the script blocks outside of COMOS. In addition, script editing offers an overview of the usage of script blocks in the COMOS projects.

1. Select the "Plugins > Basic > Script editing" command in the menu bar.
2. Select one of the tabs:
 - See chapter Using the "Search and export of scripts" tab (Page 239).
 - See chapter Using the "Import scripts" tab (Page 241).
 - See chapter Using the "Search for script usages" tab (Page 242).

The "Search and export of scripts" tab and the "Search for script uses" tab use the same option area:

- See chapter Reference of the traversal search settings (Page 237).

19.13.2 Reference of the traversal search settings

Requirement

- The script editing is opened.
See chapter Opening script editing (Page 237).

User interface

Search and export of scripts	Scripts import	Search for script uses
Traversal settings		
Search below	<input checked="" type="radio"/> Project <input type="radio"/> Start object **** Not set	
Object range	<input checked="" type="radio"/> All objects <input type="radio"/> Current working layer objects	
Object types	<input checked="" type="radio"/> All types <input type="radio"/> Selected types <input type="checkbox"/> Base objects <input type="checkbox"/> Attributes <input type="checkbox"/> Documents <input type="checkbox"/> Engineering objects <input type="checkbox"/> Base objects DevSymbols <input type="checkbox"/> Standard tables Dev	
Search mode	<input checked="" type="radio"/> Search manager <input type="radio"/> Search and scan manager	
<input type="checkbox"/> Deactivate message boxes		

Overview

- "Search below"
 - "Project"
All objects with script-capable object class are considered.
 - "Start object"
The search starts at the set start object.
- "Object area"
 - "All objects"
Searches takes place in the project in the released area.
 - "Current working layer objects"
Search takes place in the open working layer.
- "Object types"
 - "All types"
The effect corresponds to the procedure using the "Selected object types" option and selecting all object system types listed on the right.
 - "Selected types"
The search is limited to the object system types selected on the right.
- "Search mode"
 - "Search manager"
This setting uses the database search. This setting makes sense at large amounts of data, for example when a search is carried out in the database.
You can find additional information on the search manager in the "Administration" manual, keyword "Using the search manager".
 - "Search and scan manager"
This setting uses the database search and the ScanManager. This setting is only advisable for small amounts of data and is recommended especially when the search is limited to the "Attributes" object type.
You can find more information on the scan manager in the "Class Documentation COMOS_dll" manual, keyword "IComosDScanManager".
- "Deactivate message boxes"
Activated: The feedback messages of the COMOS kernel are deactivated.

Use of the traversal search settings

- See chapter Using the "Search and export of scripts" tab (Page 239).
- See chapter Using the "Search for script usages" tab (Page 242).

19.13.3 Using the "Search and export of scripts" tab

Requirement

- The script editing is opened.
See chapter Opening script editing (Page 237).
- The "Search and export of scripts" tab is opened.

Alternative 1: Exporting all scripts in one step

1. Set the traversal settings.
See chapter Reference of the traversal search settings (Page 237).
2. Click the "List all scripts" button in the "Export all scripts" group.
The "Search folder" window opens.
3. Select the folder in which the folder structure with the vbs files should be created.
The "COMOS Traversal Status" window opens.
4. Check the "COMOS Traversal Status" window.
"Status": The entry "Finished" is displayed in this field once the function is completed.
Optional: Click the "Cancel" button. The button disappears when the "Finished" status is displayed.

Alternative 2: List scripts first, check and optional export

1. Set the traversal settings.
See chapter Reference of the traversal search settings (Page 237).
2. Click the "List all scripts" button in the "List all scripts" group.
The "COMOS Traversal Status" window opens.
3. Check the "COMOS Traversal Status" window.
"Status": The entry "Finished" is displayed in this field once the function is completed.
Optional: Click the "Cancel" button. The button disappears when the "Finished" status is displayed.
The script blocks are listed in the "Hit list" area.

4. Check the found scripts.
 - Select an entry in the "Hit list" area. The script is displayed in the "Script" area.
 - Select the "Navigate to object" command in the context menu of a list entry.
5. Optional: Select an option for saving the scripts according to the hit list.
 - Option 1: Click "Export hit list to Excel" in the "Save results" area.
The hit list is saved in an Excel list.
 - Option 2: Click "Export search result scripts" in the "Save results" area.
The "Export search results" window opens. Select an option: "Export to Excel" or "Export to text files".
If you have selected the "Export to Excel" option, Excel opens. A script is entered per Excel row.
If you have selected the "Export by text files" option, the "Search folder" window opens. Select a folder. A folder structure is created according to the SystemFullName of the script blocks. The script blocks are exported as vbx text files.

The progress and the completion of the export is shown in the "Save results" area.

Alternative 3: List scripts according to the regular expression, check them, followed by optional export

1. Set the traversal settings.
See chapter Reference of the traversal search settings (Page 237).
2. Searching and listing scripts with regular expression
 - Enter a regular expression In the "Pattern" field.
In order to enter several expressions simultaneously use "Return" and enter each regular expression in a separate row.
 - Optional: Click the "Case-sensitive" option.
 - Click the "Find matching scripts" button.

The script blocks are listed in the "Hit list" area.
3. Check the found scripts.
 - Select an entry in the "Hit list" area. The script is displayed in the "Script" area.
 - The expression searched for is colored in the script.
 - Select the "Navigate to object" command in the context menu of a list entry.
4. Optional: Select an option for saving the scripts according to the hit list.
 - Option 1: Click "Export hit list to Excel" in the "Save results" area.
The hit list is saved in an Excel list including script blocks.
 - Option 2: Click "Export search result scripts" in the "Save results" area.
The "Export search results" window opens. Select an option.
If you have selected the "Export by text files" option, the "Search folder" window opens. Select a folder. A folder structure is created according to the SystemFullName of the script blocks. The script blocks are exported as vbx text files.

The progress and the completion of the export is shown in the "Save results" area.

Example: Listing COMOS script library

1. Select the following base object: "@20 > A80 Scripts"
2. Drag the base object to the "Start object" field.
3. Activate the "All objects" setting for the "Object area" option.
4. Activate the "Base objects" setting for the "Object types" option.
5. Activate the "Search manager" setting for the "Search mode" option.
6. In the "Patterns" field enter a text that appears in all script blocks, for example, "ScriptInfoHeader".
7. Click the "Find matching scripts" button.
The list area "hit list" is being filled. When you select an entry of the hit list, the associated script appears in the "Script" field.

Reimporting scripts

See chapter Using the "Import scripts" tab (Page 241).

19.13.4 Using the "Import scripts" tab**Requirement**

- You are familiar with the overview of the script editing.
See chapter Opening script editing (Page 237).
- Depending on the workflow:
 - The folder structure and the vbs files must have been created by an export from COMOS.
See chapter Using the "Search and export of scripts" tab (Page 239).
Folder structure and name of the vbs files must be unchanged.
 - The Excel file with the scripts must have been created by an export from COMOS.
- The "Import scripts" tab is opened.

Reimporting vbs text files

In this workflow, the script blocks were exported as individual vbs text files.

1. Click the "Reimport scripts" button.
The list area "Import log" is emptied.
The "Search folder" window opens.
2. Select a folder and confirm your selection with "OK".
The folder structures are being evaluated in the selected folder and the matching base data structures are searched in COMOS.
If a matching base data structure and a vbs file is found, the vbs file is imported into the script block.
A row is created in the "Import log" list area for each correct vbs file.
The row includes the following information:
 - SystemFullName of the script object
 - Number of the script block
 - Status: "File is not modified" / "Script imported successfully"
3. Optional for a row with the entry "Script imported successfully": Select the "Navigate to object" command in the context menu.

Reimporting an Excel file

1. Click on the "Reimport Excel" button.
The list area "Import log" is emptied.
The "Select an Excel file of scripts to be imported" window opens.
2. Select an Excel file and confirm your selection with "Open".
The Excel file is being evaluated.
A row is created in the Excel file in the "Import log" list area for each correct row. Faulty entries in the Excel file are skipped without feedback.
The row includes the following information:
 - SystemFullName of the script object
 - Number of the script block
 - Status: "Script is not modified" / "Script imported successfully"
3. Optional for a row with the entry "Script imported successfully": Select the "Navigate to object" command in the context menu.

19.13.5 Using the "Search for script usages" tab

Requirement

- You are familiar with the overview of the script editing.
See chapter Opening script editing (Page 237).

Procedure

1. Set the traversal settings.
See chapter Reference of the traversal search settings (Page 237).
2. Drag&drop an object in the "Search parameters" area to the "Script root" field.
All following work steps have an effect on this object and the objects below.
Depending on the traversal settings and the set root node, the next work step can take a long time.
3. Select one of the following options:
 - "Delete all script uses"
The "Attributes > Script uses" tab of the script blocks is edited as follows:
 - All entries are deleted. This does not delete the script use, but the script uses are no longer documented in this tab.
 - "Evaluation" with the "Without changes in the script library" option
Script uses are checked. The test results are documented in the areas "Script library" and "Script uses".
 - "Evaluation" with the "With changes in the script library" option
Script uses are checked. The "Attributes > Script uses" tab of the script blocks is edited as follows:
 - Script uses that no longer exist are deleted.
 - New script uses are entered.All actions are documented in the areas "Script library" and "Script uses".
4. Optional: Click the "Save log to text file" button.
The texts in the "Script Library" area and the "Script uses" area are saved in a text file.

See also

Structure of the script library (Page 208)

Editing project properties

20.1 Global project options

Global options

The properties of the object are saved in an XML file which can only be changed by Siemens.

You have the following options for specifying templates for a COMOS project:

- Project properties, "General settings > Project > Links > Project structure"
- Creating attributes for the released area of the working layers and processing them using scripts

Specifying the scope for the option

- Database scope
Edit mode of the attribute set to "Editable - only on base data side".
- Project scope
Edit mode of the attribute set to "Editable - normal".

See also

Customer-specific project properties (Page 256)

20.2 Base references: References to database objects

Aim

Base references determine the base objects or root objects used for certain tasks ("hardcoding"). Example: The "Cable catalog" base reference specifies the base object below which the the cable objects are located. Base references save you from having to enter specific path details in scripts. Customizations are easy to make as they involve changing the base reference.

Example:

```
"@99 > A30 > M00 > A40 > A10 Link object"
```

Using the FullName

Base references are created as reference attributes. The display should be set to "FullName of the linked object" in the attribute display.

See also

Alternative operation modes for hardcoding (Page 158)

20.3 Project properties reference

You can find the reference of the project properties in the "COMOS Administration" manual, keyword "Overview of the project properties".

Customer-specific additions

21.1 General rules for customer-specific supplements

Preference of the standards

The structure rules of the iDB must be observed. In particular:

- The existing structures must be given preference.
- The existing objects must be given preference.
This rule applies in particular to attributes.
- Standard objects may not be deleted.

Hierarchical structures and name rules for customer-specific supplements

The following options are available for customer-specific application cases:

- Objects that do not originate from @20 or @10 can also be created in the customer area of the @30 branch.
There are two variants in this case:
 - Completely independent structures in the "@30 > Z00" node
 - Expansions to the standard nodes by means of subnodes of the "Z" type
Standard objects can also be linked in this way. If linking is carried out to standard objects, the iDB design rules described in this manual have to be observed. This ensure that the updateability of the objects is retained.
You can find additional information on the topic of database updating in the "COMOS Administration" manual, keyword "Objective and general conditions of database update".
- Attributes in the "Z" area can be changed freely.

Storage updating when changing manual activities and automated activities

COMOS provides tools for the automatic generation of customer-specific base data. For further information see the block "Automatic editing of the customer-specific base data" at the end of this topic. COMOS saves the names which are no longer available for the generation of objects in the name memory for automatic editing.

If you create the base data manually, the names created in the process are not taken into consideration in the name memory. During the next automated generation of base data COMOS could try to generate names that are already used.

21.2 Inserting customer-specific base objects in the iDB

You have to update the name memory during storage updating when changing manual activities and automated activities:

1. Open the properties of the following object:
"@40 @Y Attributes and tabs"

2. Check the following attributes:

- "Attribute search and creation" tab: "Configuration"
- "Create tabs" tab "Configuration"

These two attributes specify the name area for which the name memory is updated.

3. Select the following tab:
"Attributes > Configuration"

4. Click "Attributes" or "Tabs" in the "Memory update" group.

The name memory of the attributes or of the tabs is updated. Depending on the database this can take some minutes.

Managing cross inheritances with "B10 Tooling"

See chapter "B10 Tooling > @30 Tool": Administering cross-inheritances (Page 159).

Manual editing of the customer-specific base data

- See chapter Inserting customer-specific base objects in the iDB (Page 46).
- See chapter Inserting customer-specific base objects in the iDB (Page 248).
- See chapter Creating customer-specific attributes and tabs (Page 251).
- See chapter Creating customer-specific documents (Page 253).

Automated editing of the customer-specific base data

- See chapter Importing attributes and tabs in bulk (Page 257).
- See chapter Attributes and tabs are generating individually automatically (Page 285).

Manual editing of the customer-specific project properties

- See chapter Customer-specific project properties (Page 256).

21.2 Inserting customer-specific base objects in the iDB

Requirement

- The general rules for customer-specific supplements are known.
See also chapter General rules for customer-specific supplements (Page 247).

Prepared z10 customer nodes

There is a customer node prepared in each of the main nodes @10, @20, @30, and @40 (starting from the second layer):



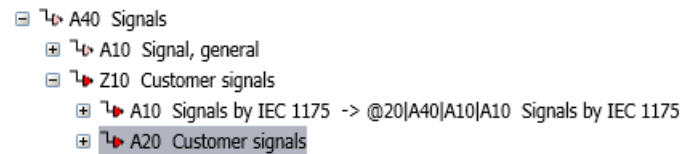
You can create more customer nodes within the existing COMOS iDB structures using the letter Z and COMOS iDB counting.

Creating base objects in a "Z10" customer node

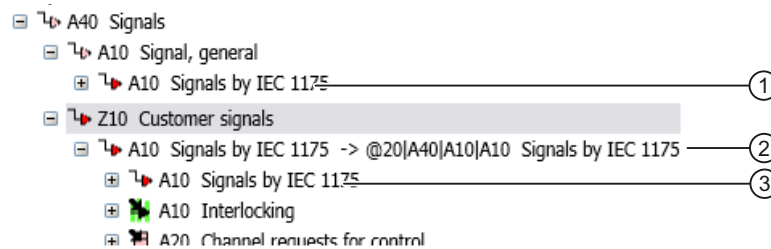
You have the following options below a customer node:

1. Own base object

The base object can also be a modified copy of a standard object from the COMOS iDB.



2. Derived base object from a cross-inherited object from the COMOS iDB



- (1) Standard object in COMOS iDB
- (2) Cross-inherited base object (reference in the "Base object" field)
- (3) Child object of base object (2)

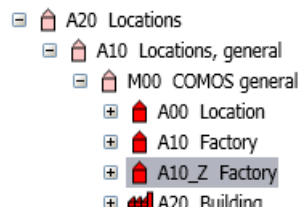
3. Change to standard object

With this method, the changes are lost each time the iDB is updated.

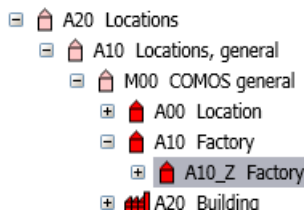
Alternative: Customer nodes with suffix "_Z"

It is possible to insert customer nodes labeled with the suffix "_Z" into the standard structures. The following options are available:

1. Creating a copy that is parallel to the standard object and entering the suffix "_Z":



2. Creating a child object below the standard object and entering the suffix "_Z":



Customer-specific attributes

The attributes of the customer-specific objects are not created at the base object, but instead in the customer area of @40. See chapter Creating customer-specific attributes and tabs (Page 251).

Customer-specific connectors and elements

- Editing at the base object (checking in)
Connectors which are checked in and edited can be overwritten by changes to the iDB supplied by Siemens. Therefore, checked-in and edited connectors must be checked after each update to the iDB.
- Newly inherited
New connectors which have been created by inheritance can lead to duplicate connectors with identical names after the iDB is updated. These connectors must be merged or synchronized after each update to the iDB.

Customization of descriptions, symbols, and scripts

These customizations can be resolved by the object matcher when the iDB is updated.

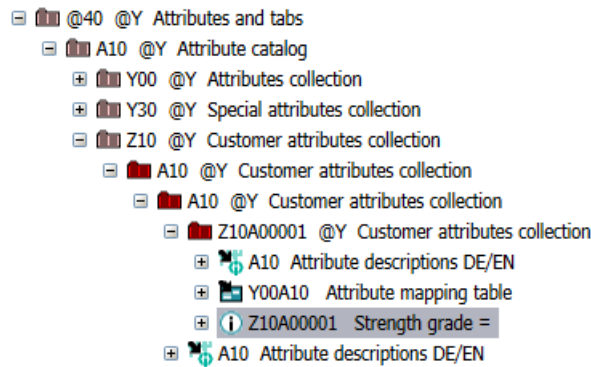
21.3 Creating customer-specific attributes and tabs

Requirement

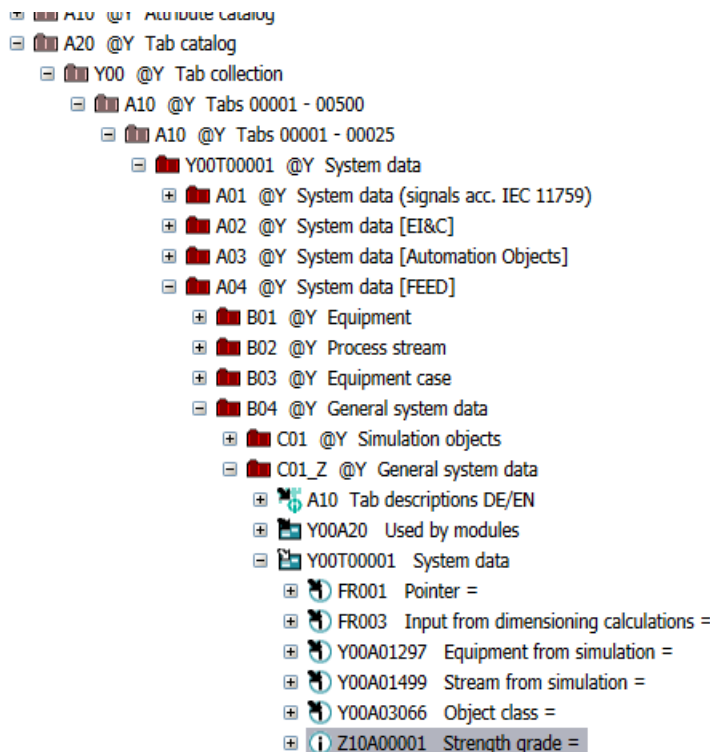
- The general rules for customer-specific supplements are known.
See also chapter General rules for customer-specific supplements (Page 247).

Customer attributes and customer tabs

The customer attributes are created in the Z area with the same structure as in the iDB. The concept of carrier attributes is used:



The supplied standard tabs are extended by means of the customer attributes. The customer-specific variants of the tabs are created to this purpose:



21.3 Creating customer-specific attributes and tabs

The variant can be generated by two methods:

- Creating and customizing a copy



- Creating and customizing a subvariant



The customizations remain unchanged when the iDB is updated.

Using customer-specific tabs

The tabs are assigned to the base objects ("attaching"). The following alternatives are available to you:

- Attached to a standard object
- Attached to an object copy
- Attached to a derived object



21.4 Creating customer-specific documents

Requirement

- The general rules for customer-specific supplements are known.
See also chapter General rules for customer-specific supplements (Page 247).

Customer-specific connectors

The following alternatives are available for creating customer-specific documents:

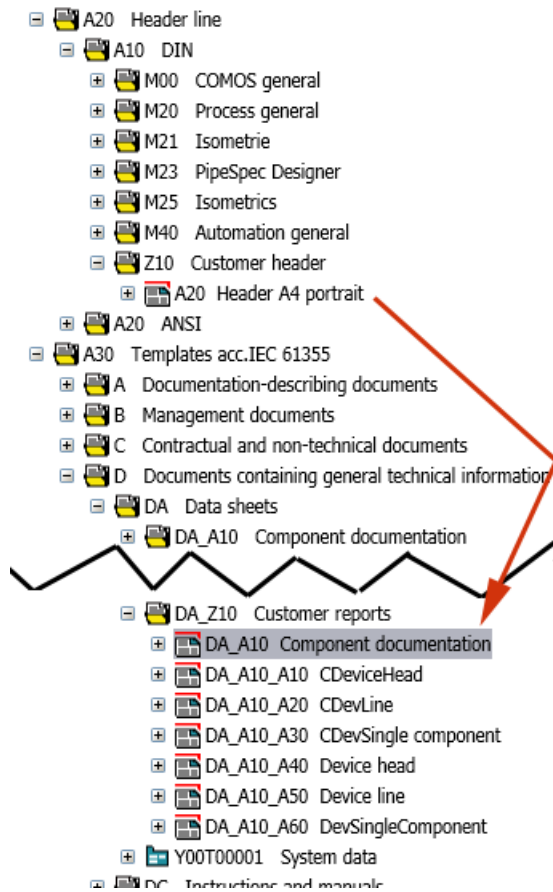
- Copying
- Changing the actual object
With this procedure, customer-specific changes may be lost when the iDB is updated.

Procedure:

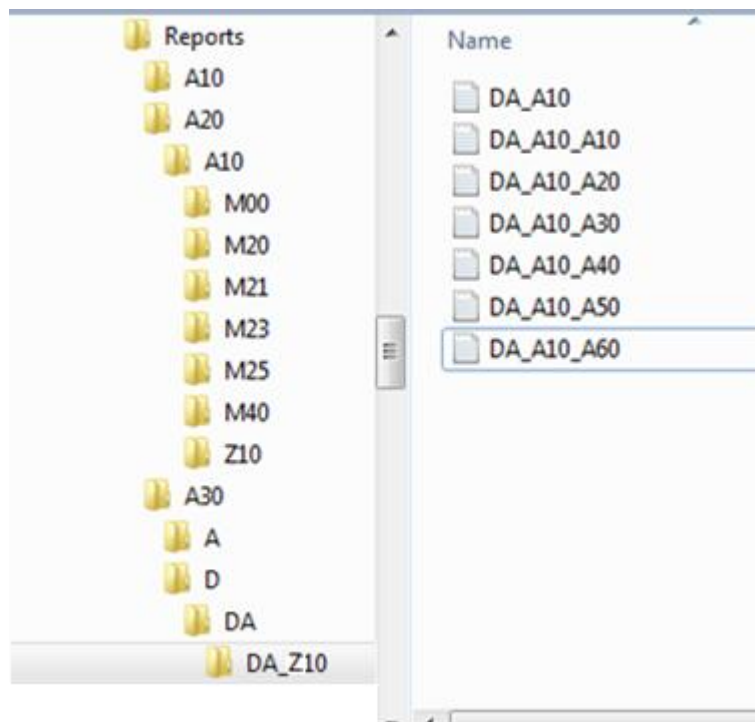
- The standard examples of reports are copied and stored in the customer areas
- A customer layer is created in the structure in the style of IEC 61355, parallel to the structure keys

21.4 Creating customer-specific documents

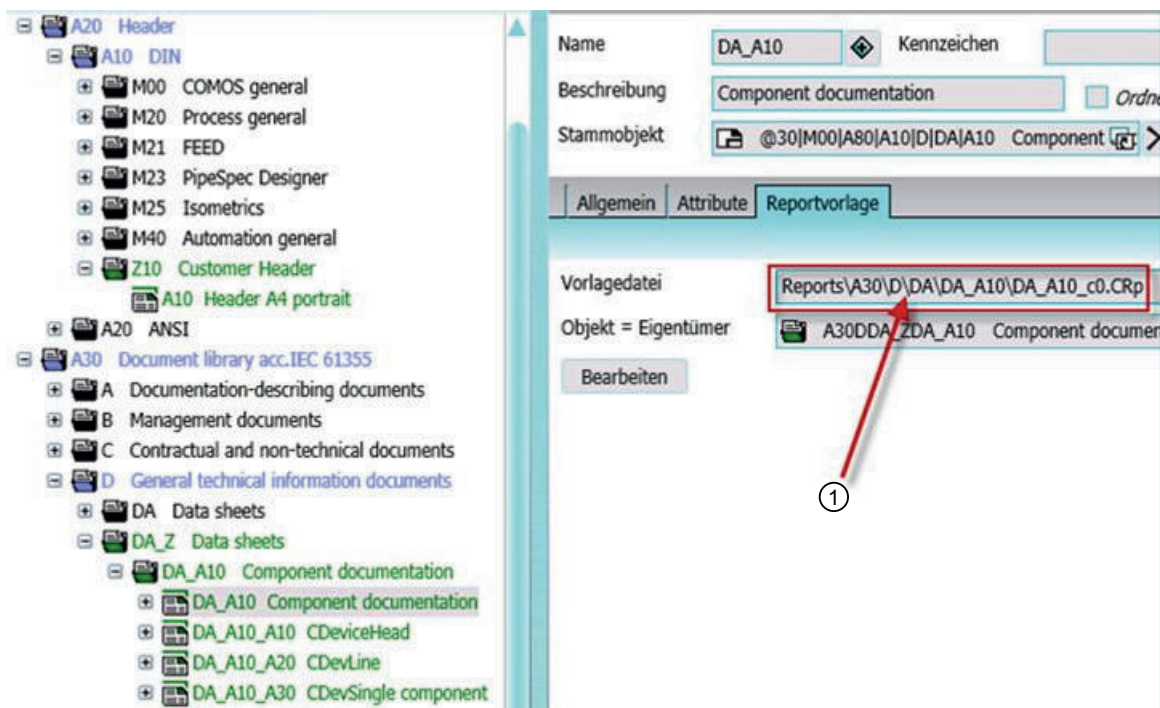
- The customer reports are located below the customer layer
 - The customer reports use unmodified standard master reports and customized headers where possible
 - The customer headers are created in their own customer area in the header node.
 - All other customizations are set at the customer reports in the structure in the style of IEC 61355. Customer subreports are also created at this position.
- Example:



The directory structure is created in the same way:



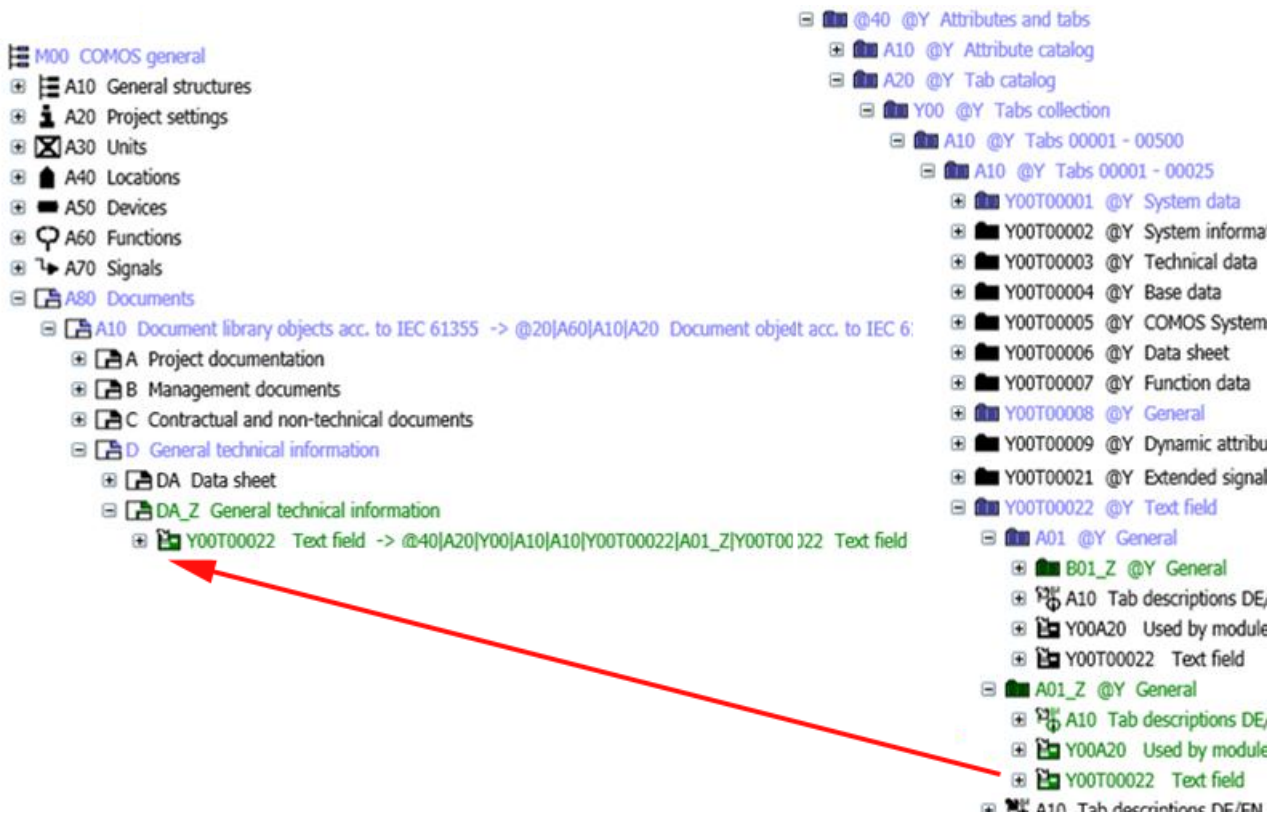
Using a report template on the document:



1. Enter the path in the Z10 area in this field. The document name is DA_Z10, for example, but it can also be expanded.

Document base objects are extended in the same way:

21.5 Customer-specific project properties



For changes to tabs, a customer variant is created and this is copied at the document base object. See section Creating customer-specific attributes and tabs (Page 251).

21.5 Customer-specific project properties

Requirement

- The general rules for customer-specific supplements are known. See also chapter General rules for customer-specific supplements (Page 247).

Overview

The customer-specific "Customer settings" category is invisible in the project properties in the delivery state. The "Customer settings" category only becomes visible when at least one customer-specific tab has been created in the project properties.

In order to create customer-specific project properties, proceed as follows:

1. Switch to the base project.
2. Select the "@30 > M00 > A20 > A10 Project presetting, global" base object.
3. Open the properties of "A10 Project presetting, global".
4. Change to the "Attributes" tab.

5. Create a new tab.
6. Create whichever attributes you choose on this tab.
7. Confirm your entries.
The "Customer settings" category appears in the project settings with a subcategory that has the same name as the tab. The attributes created on the tab are available in the subcategory.

21.6 Importing attributes and tabs in bulk

21.6.1 Aim of the iDB import

Requirement

- You are familiar with the administration of attributes and tabs.
See chapter Overview of the administration of attributes and tabs (Page 107).
- The general rules for customer-specific supplements are known.
See also chapter General rules for customer-specific supplements (Page 247).

Scope of the import

This operation type is used to edit the attribute catalog and the tab catalog in bulk based on external specifications.

The following information is imported in the attribute catalog:

- Description of the attribute
The description determines the purpose of the attribute.
- Name of the attribute
 - Standard case: The attribute name has no additional meaning in COMOS and only has to be unique. The name is created automatically during the import according to efficiency criteria.
 - Optional: Name generation is controlled or the name is specified for specific attributes individually.
- Storage path in the attribute catalog
For basics on the storage structure see chapter Structuring variants of the iDB attribute import (Page 266).
- Optional: Information for the "Attribute mapping" tab in the properties of the base objects

21.6 Importing attributes and tabs in bulk

The following information is imported in the tab catalog:

- Description of the tab
The description determines the purpose of the tab.
- Name of the tab
 - Standard case: The name is created automatically during the import according to efficiency criteria.
 - Optional: Name generation is controlled or the name is specified for specific tabs individually.
- Storage path in the tab catalog
For basics on the storage structure see chapter Structuring variants of the iDB attribute import (Page 266).

After these two work steps, you can place the attributes in the tabs of the tab catalog:

- Mapping of a catalog attribute to one or more catalog tabs
- Setting properties of the placed attribute

See also

Overview of the import of attributes and tabs (Page 258)

21.6.2 Overview of the import of attributes and tabs

Requirement

- You are familiar with the purpose of the import.
See chapter Aim of the iDB import (Page 257).

Bulk import

The bulk import includes the following work steps:

1. Import of attributes: Prepare Excel workbook "A30 Attributes & tabs V1.0"
 - Selection of a structuring variant
See chapter Structuring variants of the iDB attribute import (Page 266).
 - Edit the Excel sheet
See chapter Preparing the "Attributes" Excel table for attribute import (Page 269).
2. Import of attributes: Preparing and using the base object "(@40) @Y Attributes and tabs"
See chapter Importing attributes by means of "(@40) @Y Attributes and tabs" (Page 273).
3. Import of tabs: Prepare Excel workbook "A30 Attributes & tabs V1.0"
 - Check types and variants
See chapter Structuring variants and types of the iDB tab import (Page 277).
 - Prepare the Excel sheet for import of tabs
See chapter Preparing the "Tabs" Excel sheet for the tab import (Page 279).

4. Import of tabs: Preparing and using the base object "(@40) @Y Attributes and tabs"
See chapter Importing tabs by means of "(@40) @Y Attributes and tabs" (Page 284).
5. Mapping of the attributes to the tabs
The attributes are only mapped in the Navigator (which means in the object model) in this step. When you open the properties of the tabs you cannot see the attributes yet.
 - Mapping of the tab
 - Optional: Setting of attribute properties
According to iDB rules, the properties of placed attributes may be adapted.

This work step is prepared and executed together with the import of the tabs.

See chapters Preparing the "Tabs" Excel sheet for the tab import (Page 279) and Importing tabs by means of "(@40) @Y Attributes and tabs" (Page 284).

Alternative: Quick import of tabs during first import

During the first import, you can edit the tab catalog with only two pieces of information:

- "Naming area"
- "Quick import path"

See chapter Quick import of tabs (Page 282).

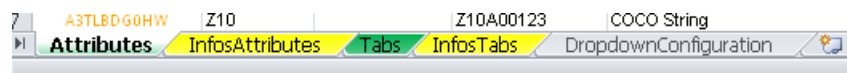
21.6.3 Overview of the Excel workbook

Requirement

- You are familiar with the overview of the import.
See chapter Overview of the import of attributes and tabs (Page 258).

User interface

This overview describes the following document object in the iDB: "(@40) @Y Attributes and tabs > A30 Attributes & tabs V1.0(.xls)".



"Attributes"	Data for import of attributes. See chapter Excel workbook: "Attributes" Excel sheet (Page 260).
"InfosAttributes"	Info text: This Excel sheet describes the applications of the "Attributes" Excel sheet. You can find the current information on the use of the Excel sheet in the "Legend" group. The following applies at the editorial deadline: <ul style="list-style-type: none"> • Light blue, blue, cyan: One or multiple columns are filled by the customer, depending on the application • Light red, dark red, yellow: These columns are written by COMOS. Check the current information and examples for the different structuring variants 1 to 5.
"Tabs"	Data for import of tabs. See chapter Excel workbook: "Tabs" Excel sheet (Page 263).
"InfosTabs"	Info text: This Excel sheet describes the applications of the "Tabs" Excel sheet. You can find the current information on the use of the Excel sheet in the "Legend" group. The following applies at the editorial deadline: <ul style="list-style-type: none"> • Light green, green, dark green: One or multiple columns are filled by the customer, depending on the application • Light red, dark red, yellow: These columns are written by COMOS.
"DropdownConfiguration"	Definitions for the other Excel sheets. Must not be edited.

See also

Preparing the "Attributes" Excel table for attribute import (Page 269)

Preparing the "Tabs" Excel sheet for the tab import (Page 279)

Overview of the naming convention in the iDB (Page 27)

21.6.4 Excel workbook: "Attributes" Excel sheet

Requirement

- You are familiar with the overview of the Excel workbook.
See chapter Overview of the Excel workbook (Page 259).

Description of the "Attributes" Excel sheet

SystemUID	Search area	Naming area	Name	German (DE)	English (EN)
-----------	-------------	-------------	------	-------------	--------------

Below are the columns of the first grouping level:

- "SystemUID"
Filled in by COMOS. UID of the attribute
- "Search area"
Specifies one or more nodes below which the system looks for duplicates. An object is recognized as duplicate when the description is identical in all entered languages. A difference can be any character, including spaces, hyphens etc. The semicolon is used as separator for multiple entries.
- "Naming area" and "Name"
Interaction: If a path is listed in the "Owner" column, the naming area also specifies the start node of the path.
The naming area can be specified with the "Naming area" column or the "Name" column.
Standard case: "Naming area" filled, "Name" initially blank
 - Column "Naming area": The value specifies the name space and the prefix of the automatically generated name. Examples: "Y00" for the Siemens name space or "Z00" for a customer name space.
 - Column "Name": Blank
The name is automatically entered in the "Name" column in a later work step. The aim is that the "Naming area" column as well as the "Name" column is filled after updating the master list.

Special case: "Naming area" initially empty, "Name" filled

 - Column "Naming area": Blank
 - Column "Name" filled: This value is the name of the attribute. Example of an object name: "Z00A12345"
An entry in the "Name" column has priority over an entry in the "Naming area" column. The name is subject to different validity checks. See also chapter Overview of the naming convention in the iDB (Page 27). For some applications, the prefix of the name must match the prefix of the owner object. The "Special naming (Y30)" option deactivates the name check.
- "German (DE)", "English (EN)", "Chinese (ZH)"
Specifies the description of the created attributes.
- "Owner"
Mandatory field for attributes without "Main attribute" label: The object under which a carrier object is created for an attribute.
For some application cases: Path and owner under which a carrier object is created for an attribute. The path is specified relative to the name space.

21.6 Importing attributes and tabs in bulk

- "iDB structured"
"Packaging" and "Special naming (Y30)" must be deactivated for this option.
The carrier objects of the attributes are created according to the iDB naming rules and below semantic collector nodes. The prefix of the object names must match the prefix of an owner object. If no owner object exists for a prefix, it is created during the import.
- "Packaging"
"iDB structured" must be deactivated for this option.
The carrier objects of the attributes are created below a two-stage structure of collector objects (package folders). If a collector object is missing in a stage, it is created during the import.
- "Special naming (Y30)"
"iDB structured" must be deactivated for this option.
Deactivates the name check.
- "Uses"
Number of utilizations of the attribute within the checked name areas below "@40 @Y > A20 Tab catalog". Filled in by COMOS.
- "Creation properties"
Date stamp "Generation" of the attribute; name of the last editor; comment. With the exception of the "Comment" field these specifications are filled in COMOS.
- "Intern"
For internal use by SIEMENS.
- "Attribute settings"
Activates the evaluation of grouped columns "AM_xxx"
 - "AM_xxx"
Sets values for the "Attribute mapping table" for the base objects of the carrier objects.
- "Synchronizing"
Specifies the type of the synchronization. See chapter Controlling repeated synchronization (Page 275).
- "Change date"
Filled in by COMOS. Date and time of the last update.
 - "Status information"
Status of the last import. An individual entry about the implementation of the last import is entered for each row in this column.

For information on administration of the Excel sheet, see chapter Preparing the "Attributes" Excel table for attribute import (Page 269).

Amendment to the "Attributes" Excel sheet: "iDB structured" grouping

- "Main attribute"

Specifies an attribute as main attribute. A "Main attribut" is created under a folder. All entries that are not labeled as "Main attribute" are automatically considered to be subattributes and need an "Owner" or a "Main attribut".
- "iDB levels"
 - The columns "MainAttribute_DE", "MainAttribute_EN" and "MainAttribute_ZH" contain information on the "Main attribut" in German, English and Chinese. An attribute with the "Main attribute" label only contains this information. The columns "Level_1_xx", "Level_2_xx" and so forth are blank.
 - The columns "Level_1_DE", "Level_1_EN" and "Level_1_ZH" contain information on "Level_1" and therefore for a subattribute of level 1. An attribute without the "Main attribute" label contains information in the columns "MainAttribute_xxx" and in the columns "Level_1_xx".
 - The other columns contain the information for the lower levels.

21.6.5 Excel workbook: "Tabs" Excel sheet

Requirement

- You are familiar with the overview of the Excel workbook. See chapter Overview of the Excel workbook (Page 259).

Description of the "Tabs" Excel sheet

A	B	C	D	E	F	G	H	I	J
SystemUID	Tab name	Naming area	Owner	Variant	Search area	Attributes	Type	German (DE)	English

- "SystemUID"

Filled in by COMOS. UID of the tab
- "Tab name"
 - Standard case: The column "Tab name" is initially blank. The name is created automatically based on the name space specified in the "Naming area" field. Examples: "Y00" for the Siemens name space or "Z00" for a customer name space. The name is automatically entered in the "Tab name" column in a later work step.
 - Special case: Column "Tab name" filled: Entered manually and thus binding name of the tab

- "Type"
Defines the type of the created tab:
 - "T" (= Tab)
Import of a tab
 - "V" (= Variant)
Import of a tab variant
 - "A" (= Placing attribute)
Placing of an attribute in a tab variant

See also chapter Structuring variants and types of the iDB tab import (Page 277).
- "German (DE)", "English (EN)", "Chinese (ZH)"
Specifies the description of the created tabs.
- "Packaging"
The carrier objects of the tabs are created below a two-stage structure of collector objects. If a collector object is missing in a stage, it is created during the import. For "Type" = V and "Type" = A this column remains blank. If "Packaging" is not activated, customers can define their own structures or the tabs are created flat below the respective main node.
- "Special naming (Y30)"
Deactivates the name check.
- "Quick import"
Activates the quick import for this row. A string is entered during quick import; it contains the path information as well as the name of the object to be created.
- "Attribute settings"
Additional information for "Type" = A. According to iDB design rules, it is permitted to adapt the properties of attributes when these are placed in tabs.
- "Module", "Comment"
Filled in by COMOS.
- "Synchronizing"
Specifies whether the Excel sheet is updated by COMOS.
- "Change date"
Filled in by COMOS. Date and time of the last update.
 - "Status information"
Status of the last import. An individual entry about the implementation of the last import is entered for each row in this column.

For information on administration of the Excel sheet, see chapter Preparing the "Tabs" Excel sheet for the tab import (Page 279).

Amendment to the "Tabs" Excel sheet: "Type" grouping

- "Naming area"
Interaction: If a path is listed in the "Owner" column, the naming area also specifies the start node of the path.
The naming area can be specified with the "Tab Name" column or the "Naming area" column.
Standard case: "Tab name" initially empty, "Naming area" filled
 - Column "Tab name": Blank
The name is automatically entered in the "Tab name" column in a later work step. The aim is that the "Tab name" column as well as the "Naming area" column is filled after updating the master list.
 - Column "Naming area" filled: The value specifies the name space and the prefix of the automatically generated name. Examples: "Y00" for the Siemens name space or "Z00" for a customer name space.Special case: "Tab name" filled, "Naming area" initially blank
 - Column "Tab name": This value is the name of the attribute. Example of an object name: "Z00A12345"
An entry in the "Tab name" column has priority over an entry in the "Naming area" column. The name is subject to different validity checks. See also chapter Overview of the naming convention in the iDB (Page 27). For some applications, the prefix of the name must match the prefix of the owner object. The "Special naming (Y30)" option deactivates the name check.
 - Column "Naming area": Blank
The name area is automatically entered in the "Naming area" column in a later work step. The aim is that the "Tab name" column as well as the "Naming area" column is filled after updating the master list.
- "Owner"
For some application cases: Path and owner under which a carrier object is created for a tab. The path is specified relative to the name space.
- "Variant"
Additional information for Type = "V". Defines the object for which a variant is created.
- "Search area"
Specifies one or more nodes below which the system looks for duplicates. An object is recognized as duplicate when the description is identical in all entered languages. A difference can be any character, including spaces, hyphens etc.
- "Catalog attributes"
Additional information for Type = "A". Defines the catalog attribute to be placed.
- "Placed attributes"
Additional information for "Type" = A. Contains the name of the placed attribute. Filled in by COMOS.

21.6.6 Structuring variants of the iDB attribute import

Requirement

- You are familiar with the purpose of the iDB attribute import. See chapter Aim of the iDB import (Page 257).
- You are familiar with the overview of the import of attributes and tabs. See chapter Overview of the import of attributes and tabs (Page 258).

Storage of attributes directly below carrier objects

Two methods are available in COMOS to prepare attributes in the base data:

- Standard case: CDevice > Tab > Attribute
- Special case: CDevice > Attribute
The attribute catalog and therefore the iDB attribute import uses this special case:
 - In the Navigator, the original of the attribute is located in the attribute catalog below the CDevice.
The designation "Owner of this attribute" in this case does not refer to a tab but to a CDevice.
 - The attribute is not placed on any tab. You will not see the attribute when you open the properties of the CDevice. When customer-specific attributes are imported, the CDevice will not even have a tab for attributes.

Supported structuring variants

General information on designation of owner objects is available in chapter Special sub-branches with Y prefix (Page 24).

The following options exist for managing the base data of the attributes in the iDB:

- 1) Sorting of customer attributes in the Siemens pool, i.e. in the node "Y00 @Y Attributes collection > A05 Semantic attribute collection"
See also chapter Structuring variant: A05 Semantic structuring (Page 113).
- Separate administration of customer attributes in the customer folder, e.g. in the folder "(Z00) @Y Customer attributes collection":
See chapter Structuring variant: Z10 Folder for customer sorting (Page 116).
 - 2) Use of the "semantic structuring"
See also chapter Structuring variant: A05 Semantic structuring (Page 113).
 - 3) Free structuring
 - 4 a) Use of the "packaged structuring"
Sorting in packages of 500 and below that in packages of 25
See also chapter Structuring variant: A10 Packaged structuring (Page 114).
 - 4 b) Mixed:
First: Free structuring
Below: Packages of 25
- 5) Separate administration of customer attributes in the folder "Y30 @Y Special attributes collection"
(outside of the iDB naming)
 - 5 a) Without additional packaging
 - 5 b) With additional packaging
 See also chapter Structuring variant: Y30 Special attributes (Page 115).

Examples

In the examples below, the name "Z00" can also be replaced with "Z10", "Z20", etc.

1) Sorting of customer attributes into the Siemens pool, e.g. "A05 Semantic attribute collection"

"@40 @Y Attributes and tabs > A10 @Y Attribute catalog > Y00 @Y Attributes collection"

- "A05 @Y Semantic attribute collection"
 - Y00A00001 @Y Word stem 1
 - Y00A0000m @Y Attribute m
 - Z00A0000m Customer attribute
 - Y00A0000n @Y Attribute n
 - Z00A0000n Customer attribute
 - Y00A00002 @Y Word stem 2
 - ...

2) Separate administration of customer attributes in the customer folder with semantic sorting

"@40 @Y Attributes and tabs > A10 @Y Attribute catalog > Z00 @Y Customer attributes collection"

- "A05 @Y Semantic attribute collection"
 - Z00A00001 @Y Word stem 1
 - Z00A0000m @Y Attribute m
 - Z00A0000n @Y Attribute n
 - Z00A00002 @Y Word stem 2
 - Z00A0000x @Y Attribute x
 - Z00A0000y @Y Attribute y
 - ...

3) Separate administration of customer attributes in the customer folder with free structuring

"@40 @Y Attributes and tabs > A10 @Y Attribute catalog > Z00 @Y Customer attributes collection"

- "A10 @Y Attribute folder 1"
 - Z00A00001 @Y Attribute 1
 - Z00A00002 @Y Attribute 2
 - ...
- "A20 @Y Attribute folder 2"
 - Z00A0000m @Y Attribute m
 - Z00A0000n @Y Attribute n
 - ...

4a) Separate administration of customer attributes in the customer folder with packaged structuring

"@40 @Y Attributes and tabs > A10 @Y Attribute catalog > Z00 @Y Customer attributes collection"

- "P01 @Y Attribute collection 00001 - 00025"
 - Z00A00001 @Y Attribute 1
 - Z00A00002 @Y Attribute 2
 - ...
 - Z00A00025 @Y Attribute 25
- "P02 @Y Attribute collection 00026 - 00050"
 - Z00A00026 @Y Attribute 26
 - ...

4b) Separate administration of customer attributes in the customer folder with free structuring and packaged structuring

"@40 @Y Attributes and tabs > A10 @Y Attribute catalog > Z00 @Y Customer attributes collection"

- "A10 @Y Attribute folder 1
 - P01 @Y Attribute collection 00001 - 00025
 - Z00A00001 @Y Attribute 1
 - Z00A00002 @Y Attribute 2
 - ...
 - Z00A00025 @Y Attribute 25
 - P02 @Y Attribute collection 00026 - 00050
 - Z00A00026 @Y Attribute 26
 - ...
- "A20 @Y Attribute folder 2
 - ...

5) Separate administration of customer attributes in the folder "Y30 @Y Special attributes collection"

"@40 @Y Attributes and tabs > A10 @Y Attribute catalog > Y30 @Y Special attributes collection"

- "Z00 @Y customer-specific attribute collection
 - A10 @Y Attributes 00001 - 00025
 - Any name 1 @Y Attribute 1
 - Any name 2 @Y Attribute 2
 - A20 @Y Attributes 00026 - 00050
 - ...

21.6.7 Preparing the "Attributes" Excel table for attribute import**Requirement**

- You are familiar with the overview of the Excel workbook.
See chapter Overview of the Excel workbook (Page 259).
- You are familiar with the structuring variants of the import.
See chapter Structuring variants of the iDB attribute import (Page 266).

Standard inputs

1. Select the following document object in the iDB:
"@40 > A30 Attribute & tabs V1.0"
2. Open the Excel workbook

3. Select the "Attributes" Excel sheet
4. Create a row for each attribute that is to be imported.
5. Enter the description:
 - Mandatory values: "German (DE)", "English (EN)"
 - Optional values: "Chinese (ZH)"

Inputs for a structuring variant

1. Structuring variant 1 "Import with iDB structuring in the delivery area":
 - Column "Search area": Enter the value "Y00".
 - "Naming area" / "Name": See section "Standard inputs".
 - Column "Owner":
 - Standard case: Column "Main attribute" deactivated:
Standard case for adding customer-specific attributes in the Y00 delivery area. Enter the name of the existing Siemens object (e.g. "Y00A00744") in the "Owner" column. All CDevices below the node "@40 > A10 > Y00" have a unique name, so no path is entered.
 - Special case: Column "Main attribute" activated:
Leave "Owner" empty.
 - Column "Main attribute"
 - Standard case: Column "Main attribute" deactivated:
 - Special case: Column "Main attribute" activated: Because the structuring variant 1 uses the name space "Y00", a main attribute must have the name prefix "Y00".
 - Column "iDB structured": Set an "x".
 - "Packaging" column and "Special naming (Y30)" column: Blank
 - .
2. Structuring variant 2 "Import with iDB structuring in the customer area"
 - Column "Search area": Enter the name of the customer node (e.g. "Z30")
 - "Naming area" / "Name": See section "Standard inputs".
 - Column "Owner":
 - Column "Main attribute" activated:
Leave "Owner" empty.
 - Column "Main attribute" deactivated:
Enter the name of the existing customer object (e.g. "Z30A00744"). All CDevices below a name space (in this example "Z30") must have a unique name, so no path is entered.
 - Column "Main attribute"
 - Column "Main attribute" activated: Use automatic naming by filling the "Naming area" column (in this example "Z30") and leaving the "Name" column blank.
 - Column "iDB structured": Set an "x".
 - "Packaging" column and "Special naming (Y30)" column: Blank
 - .

3. Structuring variant 3 "Import with specified structuring"

- Column "Search area": Enter the name of the customer node (e.g. "Z30").
- "Naming area" / "Name": See section "Standard inputs".
- Column "Owner":
Enter the path and the name of the customer object. Start the path with the separator "|" and then place this separator between each additional level. The path starts from the name space. Missing folder nodes are created during the import.
If "Owner" is blank, the objects are created directly below the root node according to the naming area.
- Column "Main attribute": Blank
- Column "iDB structured": Blank
- "Packaging" column and "Special naming (Y30)" column: Blank

4. Structuring variant 4a "Import in packages"

- Column "Search area": Enter the name of the customer node (e.g. "Z30")
- "Naming area" / "Name": See section "Standard inputs".
- Column "Owner": Blank
- Column "Main attribute": Blank
- Column "iDB structured": Blank
- Column "Packaging": Set an "x".
- Column "Special naming (Y30)": Blank

21.6 Importing attributes and tabs in bulk

5. Structuring variant 4b "Import in packages with specified structuring"
 - Column "Search area": Enter the name of the customer node (e.g. "Z30")
 - "Naming area" / "Name": See section "Standard inputs".
 - Column "Owner": Enter the path and the name of the customer object. Start the path with the separator "|" and then place this separator between each additional level. The path starts from the name space.
 - Column "Main attribute": Blank
 - Column "iDB structured": Blank
 - Column "Packaging": Set an "x".
 - Column "Special naming (Y30)": Blank
 - .
6. Structuring variant 5 "Y30 Import"
 - Column "Searching area": Enter the value "Y30".
 - Column "Naming area": Blank
 - Column "Name": Enter the name.
 - Column "Owner": Enter the path and name of the customer node (e.g. "Z10"). You cannot create carrier objects for attributes directly below "Y30".
 - Column "Main attribute": Blank
 - Column "iDB structured": Blank
 - Optional for structuring variant 5 b: Column "Packaging": Set an "x".
 - Column "Special naming (Y30)": Set an "x".
 - .

Optional inputs and additional steps in COMOS

1. Optional: Enter a second value in the "Search area" column. Separator: ";".
When two values are entered, the system looks for duplicates below both nodes. When a duplicate is found, the respective error text is entered in the "Status information" column.
2. Optional: "Attribute settings"
The following values are written to the "Attribute mapping table" tab for each base object.
 - Open the column group "Attribute settings" in the Excel sheet.
 - Enter a value in one or multiple columns "AM_xxx".
If a value already exists in COMOS, a comparison is conducted.
3. Optional: Column "Synchronizing"
See chapter Importing attributes by means of "(@40) @Y Attributes and tabs" (Page 273).
4. Confirm your entries.
5. Close the Excel workbook.
An open Excel workbook results in access conflicts in the further work steps.

6. Open the base object properties "(@40) @Y Attributes and tabs".
 - Open the properties of the object and import the attributes.
See chapter Importing attributes by means of "(@40) @Y Attributes and tabs" (Page 273).
7. Check the import:
 - Open the Excel workbook
 - Select the "Attributes" Excel sheet
 - Check the column "Change date > Status information"
 - If necessary, check the other light red and dark red columns.

See also

Overview of the import of attributes and tabs (Page 258)

Excel workbook: "Attributes" Excel sheet (Page 260)

21.6.8 Importing attributes by means of "(@40) @Y Attributes and tabs"**Requirement**

- The Excel sheet has been prepared.
See chapter Preparing the "Attributes" Excel table for attribute import (Page 269).
- If required: The name memory has been updated.
See chapter General rules for customer-specific supplements (Page 247).

Procedure

1. Open the base object properties "(@40) @Y Attributes and tabs".
2. Select the "Attributes > Configuration" tab.
You need the group "Excel list settings" for the following workflow.
3. Click the "Set reference" button in the "Excel list" field.
4. Select an Excel workbook.
Based on the template "@40 > A30 > Attributes & tabs V1.0", you can prepare additional Excel workbooks. Select a prepared Excel workbook.
5. Click the "Refresh worksheets" button.
The standard table entered in the "Standard table" field is updated.
Default: "Y10 > M00 > A20 > Y10M00N00036 Worksheets Excel list". This standard table updates all lists and drop-down lists in all tabs that access the Excel sheets:
 - "Configuration" tab, "Settings Excel list" group: The Excel sheets are displayed in the list.
 - "Configuration" tab: The Excel sheets are displayed in the "Worksheet" drop-down lists.
 - "Excel import" tab: The Excel sheets are displayed in the "Worksheet" drop-down lists.

6. Change to the "Excel import" tab.
You need the group "Attribute import" group for the following workflow.
7. Select an Excel sheet in the "Worksheet" drop-down list.
8. Check the options:
 - "Check duplicates"
This function checks whether attributes that are to be imported already exist. The duplicate check must have been prepared in the Excel sheet. See also chapter Preparing the "Attributes" Excel table for attribute import (Page 269).
The "Search area" column in the Excel sheet determines the nodes below which the search for duplicates takes place. If you want to search below the nodes "Z00" and "Y00", for example, enter the following in the "Search area" column: "Y00;Z00".
An object in the iDB pool is recognized as a duplicate when the description is identical in all entered languages (And search).
An object is not a duplicate when the description is identical in one language, but the description is different in another language. A difference can be any character, including spaces, hyphens etc.
 - "Used attributes"
Checks during import whether uses are present in the node "@40 @Y > A20 Tabs catalog". The specification of name areas can be used to further limit where used attributes are to be searched for.
Interaction with the option "Check duplicates": The found usages form the basis of the double check.
 - "Update Excel list"
The import information (UIDs, date, etc.) is entered in the Excel sheet.
Recommendation: Activate this option.
 - "Consider Y00 node"
The attributes in the pool "@40 > A10 > Y00" are entered in the Excel sheet.
This option can result in a long runtime of the import when used for the first time.
9. Click on the "Import" button.
 - Missing attributes are created based on the Excel sheet.
 - Differences between Excel and COMOS are displayed in the "Status" column in the Excel workbook.

Controlling repeated synchronization

When the Excel workbook is used for a longer period of time, differences between COMOS and the Excel workbook can arise. The deviations can be eliminated by means of synchronization. See chapter Controlling repeated synchronization (Page 275).

21.6.9 Controlling repeated synchronization

Requirement

- The Excel sheet has been prepared.
See chapter Excel workbook: "Attributes" Excel sheet (Page 260).
See chapter Excel workbook: "Tabs" Excel sheet (Page 263).

Controlling repeated synchronization

When the Excel workbook is used for a longer period of time, differences between COMOS and the Excel workbook can arise. In this case, proceed as follows:

1. Check whether the "Update Excel list" option is enabled.
2. Open the Excel workbook.
3. Select the "Attributes" Excel sheet or the "Tabs" Excel sheet.
4. Open the "Synchronizing" column group.
5. Check the "Compare" column.
The "Compare" column contains information on whether or not there are differences between COMOS and the Excel sheet. If necessary, set a filter in the "Compare" column.
6. Edit the "Synchronizing" column:
Interaction: During synchronization, the "SystemUID" column is evaluated to identify an attribute.
 - Blank
Only information that is valid in COMOS and in the Excel sheet is accepted. Differences are documented in the Excel sheet.
 - "E" (= Excel)
The information in the Excel sheet takes precedence. COMOS is corrected.
 - "C" (= COMOS)
The information in COMOS takes precedence. The Excel sheet is corrected.
 - "O" (= Off)
This row is not synchronized. This setting reduces the import time.

Status messages at existing entries

The following status messages require that the line has already been imported once in COMOS. A value is already entered in the "SystemUID" column before the import.

21.6 Importing attributes and tabs in bulk

"Attributes" Excel sheet or "Tabs" Excel sheet after the import:

Input in the "Compare" column	Input in the "Synchronizing" column	Effect
Blank	Blank	<ul style="list-style-type: none"> • Deviating descriptions <ul style="list-style-type: none"> – The "Different" entry is assigned to the "Compare" column – The name of the languages with deviating descriptions is assigned to the "Status information" column • Identical descriptions <ul style="list-style-type: none"> – The "Already in COMOS" entry is assigned to the "Compare" column
"Already in COMOS"	Blank	<ul style="list-style-type: none"> • Deviating descriptions <ul style="list-style-type: none"> – The "Different" entry is assigned to the "Compare" column – The name of the languages with deviating descriptions is assigned to the "Status information" column • Identical descriptions <ul style="list-style-type: none"> – Column "Status information": The column contains an information that the object exists in COMOS.
"Already in COMOS"	"E"	<ul style="list-style-type: none"> • Deviating descriptions <ul style="list-style-type: none"> – Column "Compare": All the previous specifications are deleted – COMOS is updated, Excel remains unchanged – The "Synchronizing" column remains unchanged • Identical descriptions <ul style="list-style-type: none"> – The object in COMOS and the line in the Excel sheet are deleted.
"Different"	"E"	<ul style="list-style-type: none"> • Deviating descriptions <ul style="list-style-type: none"> – Column "Compare": All the previous specifications are deleted – COMOS is updated, Excel remains unchanged – The "Synchronizing" column remains unchanged • Identical descriptions <ul style="list-style-type: none"> – The object in COMOS and the line in the Excel sheet are deleted.
Blank	"E"	No effect. The line is not analyzed.
"Already in COMOS"	"C"	<ul style="list-style-type: none"> • Deviating descriptions <ul style="list-style-type: none"> – Column "Compare": All the previous specifications are deleted – Excel is updated, COMOS remains unchanged – The "Synchronizing" column remains unchanged • Identical descriptions <ul style="list-style-type: none"> – Column "Status information": The column contains an information that the object exists in COMOS.

Input in the "Compare" column	Input in the "Synchronizing" column	Effect
"Different"	"C"	<ul style="list-style-type: none"> • Deviating descriptions <ul style="list-style-type: none"> – Column "Compare": All the previous specifications are deleted – Excel is updated, COMOS remains unchanged – The "Synchronizing" column remains unchanged • Identical descriptions <ul style="list-style-type: none"> – Column "Status information": The column contains an information that the object exists in COMOS.
Blank	"C"	No effect. The line is not analyzed.

Status messages at deleted objects

The following status messages require that the object has been deleted in the meantime in COMOS.

Input in the "Compare" column	Input in the "Synchronizing" column	Effect
"Already in COMOS"	Blank	The "Not in COMOS" entry is assigned to the "Compare" column
"Different"	Blank	The "Not in COMOS" entry is assigned to the "Compare" column
Blank	Blank	The "Not in COMOS" entry is assigned to the "Compare" column
"Not in COMOS"	"E"	Line is deleted in Excel sheet
"Not in COMOS"	"C"	Line is deleted in Excel sheet
"Not in COMOS"	Blank	No effect. The line is not analyzed.

21.6.10 Structuring variants and types of the iDB tab import

Requirement

- You are familiar with the purpose of the iDB tab import. See chapter Aim of the iDB import (Page 257).
- You are familiar with the overview of the import of attributes and tabs. See chapter Overview of the import of attributes and tabs (Page 258).

Storage of tabs below carrier objects

In the default case, the catalog of tabs has the following structure:

- A carrier object
 - The tab is prepared with the correct name here but not filled yet.
 - Below: The base object with the filled tab. These base objects are also referred to as variant base objects.

See chapter Storage structure and names of the tabs (Page 120).

Supported structuring variants

General information on designation of owner objects is available in chapter Special sub-branches with Y prefix (Page 24).

The following options exist for managing the base data of the tabs in the iDB:

1. Sorting of customer tabs in the Siemens pool, i.e. in the node "(Y00) @Y Tab collection"
2. Separate administration of customer tabs in the customer folder, e.g. in the folder "(Z10) @Y Customer tab collection"
 - Without additional packaging
 - With additional packaging
Sorting in packages of 500 and below that in packages of 25
3. Sorting of customer tabs in the folder "(Y30) @Y Special tab collection" (outside of the iDB naming)
 - Without additional packaging
 - With additional packaging
Sorting in packages of 500 and below that in packages of 25

See also chapter Overview of the nodes in "@40 > A20" (Page 119).

Tab types

The import of tabs can be implemented according to the iDB standard by means of the tab types:

1. Type T: Tab
This type of import creates the carrier objects with the named but empty tabs.
2. Type V: Variant
This type of import creates the variant base objects. The tab is still empty immediately after the import, but this tab will contain the attributes later.
3. Type A: Attribute
Placing of attributes in the tab of the variant base object. For technical reasons, the attribute is mapped to the tab according to the object model, but it is not visible yet in the tab in the properties of the base object. Use drag&drop to move the attributes from the Navigator to the tab.

Examples

In the examples below, the name "Z00" can also be replaced with "Z10", "Z20", etc.

1. Sorting of customer tabs into the Siemens pool

"@40 @Y Attributes and tabs > A20 @Y Tab catalog > Y00 @Y Tab collection"

- "A10 @Y tabs 00001 - 00500 > B70 @Y tabs 00376 - 00400"
 - "Y00T00400 @Y Hardcoding"
 - "A01 @Y Hardcoding function > B03 @Y Customer hardcoding"

2. Separate administration of customer tabs in the customer folder

"@40 @Y Attributes and tabs > A20 @Y Tab catalog > Z10 @Y Tab collection"

- "A10 Folder_1 > Z00T00001 @Y Name_1"
 - "A10 @Y Name_2"

3. Sorting of customer tabs in the folder "Y30 @Y Special tab collection"

"@40 @Y Attributes and tabs > A20 @Y Tab catalog > Y30 @Y Special tab collection"

- "Z10 Folder_1 > Z00T00001 @Y Name_1"
 - "A10 @Y Name_2"

21.6.11 Preparing the "Tabs" Excel sheet for the tab import**Requirement**

- You are familiar with the overview of the Excel workbook.
See chapter Overview of the Excel workbook (Page 259).
- You are familiar with the types and variants.
See chapter Structuring variants and types of the iDB tab import (Page 277).

Standard inputs

1. Select the following document object in the iDB:
"@40 > A30 Attribute & tabs V1.0"
2. Open the Excel workbook
3. Select the "InfosTabs" Excel sheet
 - Check the current information in the "Legend" group.
The following applies at the editorial deadline:
Light green, green, dark green: One or multiple columns are filled by the customer, depending on the application
Light red, dark red, yellow: These columns are written by COMOS.
Blue: These columns are part of the "Quick import" application and are not needed here.
For information on the "Quick import" application case, see chapter Quick import of tabs (Page 282).
 - Check the current information and examples for the different tab variants 1 to 4.

21.6 Importing attributes and tabs in bulk

4. Select the "Tabs" Excel sheet
5. Create a row for each tab that is to be imported.
6. Enter a value in the following columns for each tab that is to be imported:
 - Mandatory values: "German (DE)", "English (EN)"
 - Optional values: "Chinese (ZH)"
7. Specify the name space and the name:
 - Standard case:
 - Column "Naming area": The value specifies the name space and the prefix of the automatically generated name. Examples: "Y00" for the Siemens name space or "Z00" for a customer name space.
 - Column "Tab name": Blank
 - Special case:
 - Column "Naming area": Blank
 - Column "Tab name" filled: This value is the tab name as well as the name of the CDevice to which the tab belongs. Example of an object name: "Z00T12345"

Inputs for a type

1. Column "Type": "T" (carrier objects)
Note on the structuring variants: The carrier objects with the empty tabs can be created according to the structuring variants introduced in chapter Storage structure and names of the tabs (Page 120).
 - Optional: Column "Owner"
Enter the path and the name of the owner object. Start the path with the separator "|" and then place this separator between each additional level. The path starts from the name space. Missing folder nodes are created during the import.
 - Column "Variant":
 - Column "Attributes": Blank
 - Optional: Column "Packaging": setting "x"
 - Optional: Column "Special naming (Y30)": setting "x"
Dependency: Set a name in the "Tab name" column.
 - Column "Quick import": Blank
 - Column "Attribut settings": Blank
 - .
2. Column "Type": "V" (import of tab variants)
Note on the structuring variants: The structuring variants are meaningless for the tab variants because the carrier object is specified.
 - Column "Tab name": Name of the carrier object
The carrier object must already exist or it must have been created in a preceding work step during the current import operation.
 - Column "Owner": Blank
 - Optional: Column "Variant" with path specification
See also chapter Excel workbook: "Tabs" Excel sheet (Page 263).
Enter the path and the name of the owner object. Start the path with the separator "|" and then place this separator between each additional level. The path starts from the carrier object. Missing objects are created during the import.
 - Column "Attributes": Blank
 - Column "Packaging": Blank
 - Optional: Column "Special naming (Y30)": setting "x"
Dependency: Set a name in the "Tab name" column.
 - Column "Quick import": Blank
 - Column "Attribut settings": Blank
 - .
3. Column "Type": "A": Mapping of the attributes to the tab variants
 - Column "Tab name": Name of the tab variant
 - Column "Owner": Blank

21.6 Importing attributes and tabs in bulk

- Optional: Column "Variant" with path specification
This information is not required when all variants are located directly below the carrier object.
- Column "Attributes"
The name of the attribute to be placed. If this column is empty, an attribute with matching description texts is searched for in the stored name spaces.
- Column "Packaging": Blank
- Column "Special naming (Y30)": Blank
- Optional: Column "Attribut settings" (set properties of the attributes)
Open the column group "Attribut settings" in the Excel table.
Set an "x" in the "Attribut settings" column.
Enter a value in one or multiple columns.

Optional inputs and additional steps in COMOS

1. Optional: Column "Synchronizing"
See chapter Importing tabs by means of "(@40) @Y Attributes and tabs" (Page 284).
2. Close the Excel workbook.
3. Open the base object properties "(@40) @Y Attributes and tabs".
4. Open the properties of the object and import the attributes.
See chapter Importing tabs by means of "(@40) @Y Attributes and tabs" (Page 284).
5. Check the import:
 - Open the Excel workbook
 - Select the "Tabs" Excel sheet
 - Check the column "Status information"

21.6.12 Quick import of tabs

Requirement

- You are familiar with the preparation of the Excel workbook.
See chapter Preparing the "Attributes" Excel table for attribute import (Page 269).

Procedure

Quick edit can only be used initially.

1. Select the following document object in the iDB:
"@40 > A30 Attribute & tabs V1.0"
2. Open the Excel workbook.

3. Select the "InfosTabs" Excel sheet.
 - Check the current information in the "Legend" group.
The following applies at the editorial deadline:
Light green, green, dark green: One or multiple columns are filled by the customer, depending on the application
Light red, dark red, yellow: These columns are written by COMOS.
Blue: These columns are part of the "Quick import" application.
 - Check the current information and examples for the tab variant 5.
4. Select the "Tabs" Excel sheet.
5. Create a row for each tab that is to be imported.
6. Enter a value in the following columns for each tab that is to be imported:
 - Mandatory values: "German (DE)", "English (EN)"
 - Optional values: "Chinese (ZH)"
7. Specify the name space:
 - Column "Naming area": The value specifies the name space and the prefix of the automatically generated name. Examples: "Y00" for the Siemens name space or "Z00" for a customer name space.
 - Column "Tab name": Blank
8. Open the "Quick import" column group.
 - Column "Quick import": Set an "x".
 - Column "Quick import path"
Input pattern: <carrier object>;<path and name of the tab variant>;<description of the attribute>
Example: "Technical data;Technical data V1.Technical data V2:Filter identification"
The description of the attribute is searched for in the name space in all languages.
 - Column "Owner": Blank
 - Column "Variant": Blank
 - Column "Attributes": Blank
 - Column "Packaging": Blank
 - Column "Special naming (Y30)": Blank
9. Optional: Column "Synchronizing"
See chapter Importing tabs by means of "(@40) @Y Attributes and tabs" (Page 284).
10. Close the Excel workbook.
11. Open the base object properties "(@40) @Y Attributes and tabs".

21.6 Importing attributes and tabs in bulk

12. Open the properties of the object and import the attributes.
See chapter Importing tabs by means of "(@40) @Y Attributes and tabs" (Page 284).
13. Check the import:
 - Open the Excel workbook
 - Select the "Tabs" Excel sheet
 - Check the column "Status information"

21.6.13 Importing tabs by means of "(@40) @Y Attributes and tabs"

Requirement

- The Excel workbook has been prepared.
See chapter Preparing the "Tabs" Excel sheet for the tab import (Page 279).
- If required: The name memory has been updated.
See chapter General rules for customer-specific supplements (Page 247).

Procedure

1. Open the base object properties "(@40) @Y Attributes and tabs".
2. Select the "Attributes > Configuration" tab.
You need the group "Excel list settings" for the following workflow.
3. Click the "Set reference" button in the "Excel list" field.
4. Select an Excel workbook.
Based on the template "@40 > A30 > Attributes & tabs V1.0", you can prepare additional Excel workbooks. Select a prepared Excel workbook.
5. Click the "Refresh worksheets" button.
The standard table entered in the "Standard table" field is updated. Default: "Y10 > M00 > A20 > Y10M00N00036 Worksheets Excel list". This standard table updates all lists and drop-down lists in all tabs that access the Excel sheets:
 - "Configuration" tab, "Settings Excel list" group: The Excel sheets are displayed in the list.
 - "Configuration" tab: The Excel sheets are displayed in the "Worksheet" drop-down lists.
 - "Excel Import" tab: The Excel sheets are displayed in the "Worksheet" drop-down lists.
6. Change to the "Excel import" tab.
You need the group "Tab Import" for the following workflow.
7. Select an Excel sheet in the "Worksheet" drop-down list.

8. Check the options:
 - "Check duplicates"

This function checks whether tabs that are to be imported already exist. The duplicate check must have been prepared in the Excel sheet. See also chapter Preparing the "Tabs" Excel sheet for the tab import (Page 279).

An object in the iDB pool is recognized as a duplicate when the description is identical in all entered languages (And search).

An object is not a duplicate when the description is identical in one language, but the description is different in another language. A difference can be any character, including spaces, hyphens etc.
 - "Used attributes"

Checks during import whether uses are present in the node "@40 @Y > A20 Tabs catalog". The specification of name areas can be used to further limit where used attributes are to be searched for.

Interaction with the option "Check duplicates": The found usages form the basis of the double check.
 - "Update Excel list"

The import information (UIDs, date, etc.) is entered in the Excel sheet.

Recommendation: Activate this option.
 - "Customizing variant naming"

Disabled: Standard variant count (example: "A01> B01 > C01")

Enabled: iDB count (example "A10 > A10 > A10")
9. Click on the "Import" button.
 - Missing tabs are created based on the Excel sheet.
 - Differences between Excel and COMOS are displayed in the "Status" column in the Excel workbook.

Controlling repeated synchronization

When the Excel workbook is used for a longer period of time, differences between COMOS and the Excel workbook can arise. The deviations can be eliminated by means of synchronization. See chapter Controlling repeated synchronization (Page 275).

21.7 Attributes and tabs are generating individually automatically

21.7.1 Aim of individual editing of the @40 catalog

Requirement

- You are familiar with the administration of attributes and tabs.
See chapter Overview of the administration of attributes and tabs (Page 107).
- The general rules for customer-specific supplements are known.
See also chapter General rules for customer-specific supplements (Page 247).

Overview

This chapter describes the following workflow:

1. You search for a catalog attribute in COMOS.
See chapter Finding attributes with "(@40) @Y Attributes and tabs" (Page 18).
2. You notice a missing catalog attribute.
3. You create the missing catalog attribute by means of the base object "(@40) @Y Attributes and tabs".
See chapter Create individual attribute with "(@40) @Y Attributes and tabs" (Page 288).
4. Optional: You create an additional catalog tab with the object "(@40) @Y Attributes and tabs".
See chapter Create individual tab with "(@40) @Y Attributes and tabs" (Page 289).
5. You place the catalog attribute in a catalog tab.
6. Optional: Refreshing the naming area
See chapter Refreshing the naming area (Page 291).

Alternatively, you can use bulk processing of the catalog. See also chapter Importing attributes and tabs in bulk (Page 257).

21.7.2 Configure "(@40) @Y Attributes and tabs" for individual creation

Requirement

- You are familiar with the purpose of individual processing.
See chapter Aim of individual editing of the @40 catalog (Page 285).
- The "Configuration" tab is open.
- If required: The name memory has been updated.
See chapter General rules for customer-specific supplements (Page 247).

Settings in the "Configure attribute creation" group

1. Check the entry in the "Standard table" field.
This standard table determines which cells are visible in the configuration list.
To enter additional rows in the list, open the standard table.
 - Create a new standard table value.
 - Enter a unique name.
 - Enter a text in the "Description" field.
This specification is displayed in the "Configuration" column in the "Configuration" tab.
 - For technical reasons, enter a counter in the "Value 1" field.
 - Close the standard table.Select the "Refresh (recalculate)" command in the context menu of the configuration list.
2. Check the entry in the "Worksheet" field.

3. Edit the configuration list:
 - Select a cell.
 - Press F2.
 - "Search area"
This option determines the search area for duplicates.
 - "Name space"
This option determines the name space during automatic naming.
 - "Owner"
This option determines the owner of the object to be created.
 - "Packaging"
This option activates the structuring variant "Packaging". See chapter Structuring variant: A10 Packaged structuring (Page 114).
4. Save the object "@40 > A10 @Y Attribute catalog".
The standard table is updated and saved.

Configure settings in the "Configure tab creation" group

1. Check the entry in the "Standard table" field.
This standard table determines which cells are visible in the configuration list.
To enter additional rows in the list, open the standard table.
 - Create a new standard table value.
 - Enter a unique name.
 - Enter a text in the "Description" field.
 - For technical reasons, enter a counter in the "Value 1" field.
 - Close the standard table.Select the "Refresh (recalculate)" command in the context menu of the configuration list.
2. Check the entry in the "Worksheet" field.
3. Edit the configuration list:
 - Select a cell.
 - Press F2.
 - "Name space"
This option determines the name space during automatic naming.
 - "Owner"
This option determines the owner of the object to be created.
 - "Packaging"
This option activates the structuring variant "Packaging". See chapter Structuring variant: A10 Packaged structuring (Page 114).
4. Save the object "@40 > A10 @Y Attribute catalog".
The standard table is updated and saved.

21.7.3 Create individual attribute with "(@40) @Y Attributes and tabs"

Requirement

- You are familiar with the purpose of individual processing.
See chapter Aim of individual editing of the @40 catalog (Page 285).
- Individual processing was configured.
See chapter Configure "(@40) @Y Attributes and tabs" for individual creation (Page 286).

Searching for an attribute

1. Open the properties of the object "(@40) @Y Attributes and tabs".
2. Proceed as described here:
See chapter Finding attributes with "(@40) @Y Attributes and tabs" (Page 18).

Creating an attribute

The "Attribute and creation" tab has already been opened in the preceding step.

1. Enter the description:
 - Mandatory values: "English description", "German description"
 - Optional values: "Chinese description"
2. Select an entry in the "Configuration" list.
The entries available in the list are specified as follows:
"Configuration" tab, "Configuration of individual processing of attributes" group
The meaning of the selected entry is described in chapter Configure "(@40) @Y Attributes and tabs" for individual creation (Page 286).
3. Optional: Activate the option "Duplicate check".
Prevents creation of an attribute when a duplicate is found.
4. Optional: Activate the "Used attributes" option
This option takes account of the use of attributes in the node "@40 @Y > A20 Tab catalog" while checking for duplicates.
5. Optional: Activate the "Active Excel list" option
A worksheet in the Excel workbook is updated.
 - Selection of the updated Excel workbook: "Configuration" tab, "Settings Excel list" group, "Excel list" field
 - Selection of the updated Excel sheet: "Configuration" tab, "Configure attribute creation" group, "Worksheet" fieldSee chapter Configure "(@40) @Y Attributes and tabs" for individual creation (Page 286).
6. Click the "Create" button.

Additional procedure

See chapter Create individual tab with "(@40) @Y Attributes and tabs" (Page 289).

21.7.4 Create individual tab with "(@40) @Y Attributes and tabs"

Requirement

- You are familiar with the purpose of individual processing.
See chapter Aim of individual editing of the @40 catalog (Page 285).
- Individual processing was configured.
See chapter Configure "(@40) @Y Attributes and tabs" for individual creation (Page 286).
- The properties of the object "(@40) @Y Attributes and tabs" are open.
- The "Attributes > Tab creation" tab is open.

I. Create tab and tabs variant

Edit the "Tab creation" group

1. Select an entry in the "Configuration" list.
The entries available in the list are specified as follows:
"Configuration" tab, "Configure tab creation" group
The configuration selected here also determines the owner of the created tab.
2. Optional: Activate the option "Check duplicates".
Prevents creation of a tab when a duplicate is found.
3. Enter the description:
 - Mandatory values: "English description", "German description"
 - Optional values: "Chinese description"
4. Optional: Activate the "Active Excel list" option.
A worksheet in the Excel workbook is updated.
 - Selection of the updated Excel workbook: "Configuration" tab, "Settings Excel list" group, "Excel list" field
 - Selection of the updated Excel sheet: "Configuration" tab, "Configure tab creation" group, "Worksheet" field
5. Optional: Activate the option "Handover to next step"
The entry in the field "Created tab" is transferred to the following field after you have confirmed your entries: "Creation tab variant" group, "Tab" field
6. Click the "Create" button.
An entry appears in the "Created tab" field.

Edit the "Create tab variant" group

1. Optional: Drag an object into the "Tab" field.
The field has a default entry if the information from the previous step was applied.
This object becomes the owner of the created tab variant.
2. Name the tab variant to be created in the "Variant" field.

21.7 Attributes and tabs are generating individually automatically

3. Optional: Enter the description:
 - "English description", "German description", "Chinese description"These fields have a default entry if the information from the previous step was applied.
4. Optional: Activate the "Active Excel list" option.
A worksheet in the Excel workbook is updated.
 - Selection of the updated Excel workbook: "Configuration" tab, "Settings Excel list" group, "Excel list" field
 - Selection of the updated Excel sheet: "Configuration" tab, "Configure tab creation" group, "Worksheet" field
5. Optional: Activate the option "Handover to next step"
The entry in the field "Created variant" is transferred to the following field after you have confirmed your entries: "Place attribute on tab variant" group, "Variant" field
6. Click the "Create" button.
An entry appears in the "Created variant" field.

II. Placing an attribute in a tab variant

Edit the "Place attribute in tab variant" group:

Select tab variant

1. Set a tab carrier object in the "Variant" field.
The field has a default entry if the information from the previous step was applied.
2. Check the entry in the "Tab" field.
This field has a default entry if you have edited the "Variant" field.
3. Check the "Variant structure" field.
This information is derived from the other information and cannot be edited separately.

If the "Tab" field is set correctly, all attributes that have already been placed are listed in the list on the bottom right.

Searching attribute to be placed

1. Edit the "Naming area" field.
The field determines where the search for the attribute to be placed takes place.
2. Select an entry in the "Language" list.
The field determines which languages are searched for the entry of the "Attribute description" field.
3. Edit the "Attribute description" field.
The search is for attributes with this description.
4. Click the "Search" button.
The result list is displayed in the long list.

Placing attributes

1. Select an entry in the result list.
2. Use drag&drop to move the selected row to the "Catalog attribute" field.
The fields "English description", "German description", "Chinese description are filled.

3. Optional: Enter new descriptions:
 - "English description", "German description", "Chinese description"
4. Optional: Activate the "Active Excel list" option.
A worksheet in the Excel workbook is updated.
 - Selection of the updated Excel workbook: "Configuration" tab, "Settings Excel list" group, "Excel list" field
 - Selection of the updated Excel sheet: "Configuration" tab, "Configure tab creation" group, "Worksheet" field
5. Click the "Place attributes" button.

21.7.5 Refreshing the naming area

Requirement

- You are familiar with the purpose of individual processing.
See chapter Aim of individual editing of the @40 catalog (Page 285).

Procedure

If you generate attributes or tabs manually, then you generate a deviation to the specifications in the Excel workbook. For more information on Excel workbooks see chapter Overview of the import of attributes and tabs (Page 258).

Proceed as follows to eliminate the deviation:

1. Open the "Configuration" tab.
2. Click the "Attributes" button or the "Tabs" button.

21.8 Obsolete base data

Obsolete attributes are collected at the following location:

- "@40 > A10 > Y00 > A05 > Y00A99999 Obsolete attributes"
Example: Attributes whose description (DE/EN) was created twice in the pool.

Obsolete standard tables are collected at the following location:

- Standard tables: "Y99 Obsolete standard tables"

The obsolete attributes and standard tables should no longer be used in customizing.

iDB-specific rules for using external files

22.1 Using or editing external files

Definition of the term "external file"

Every file that is saved in the document directory is an external file. Examples:

- .CRp files for report templates
- Other document templates
- .bmp bitmaps
- DLLs
- ico
See chapter Setting the object icon manually (Page 47).

General conditions

- Fields, scripts and report options are stored in report templates.
- Except for .crp files, external files are not taken into account by the working layer management in COMOS. Changes to the external file affect not only the current working layer but also the entire database. These external files are not exported when a working layer is exported.
- The .crp files are working layer-dependent in the iDB. For this reason, every .crp file must reference a COMOS object.

Special characters forbidden in names of external files

File names of external files must not contain language-specific special characters such as é, ö, ß, ø, etc. Umlauts (ä, ö, ü) are, as usual, replaced by the regular character followed by an e and a corresponding character is used for all other special characters.

Examples:

Störung > *Stoerung*

Straße > *Strasse*

Bjørn > Bjorn

École > Ecole

"Revision" file name not permitted

The file name "Revision" is not permitted for pictures that are loaded into reports as follows:

- Report designer > Place picture box
- Move the image file (JPG, BMP) from the Navigator or Windows Explorer to the report using drag-and-drop.

Background:

The "Revision" file name for integrated pictures is reserved exclusively for redlining documents.

See also

Physical storage of the report templates (Page 294)

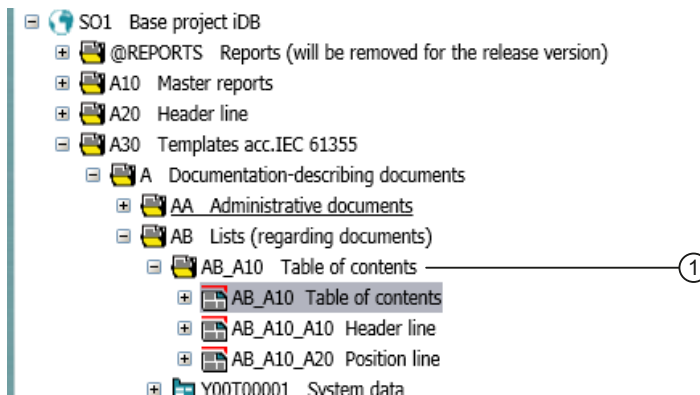
22.2 Physical storage of the report templates

Requirement

- You are familiar with the general information for using external files. See chapter Using or editing external files (Page 293).
- You are familiar with the working layer dependent creation of template files. You can find additional information on this topic in the "COMOS Administration" manual, keyword "Creating a working layer-dependent template file in COMOS".

Overview

There is a structure with document groups on the "Documents" tab:



The external report files are stored project-specifically in a matching directory structure.

Implementation

The following structure exists in the physical document folder for each project folder:

Project folder\...

- Overlays
 - WOXXXX
This folder represents the working layer "MRA".
 - WOYYYY
This folder represents the working layer "UCL".
- Reports
This folder represents the working layer "CRA".

You can find additional information on the terms MRA, UCL and CRA in the "COMOS Administration" manual, keyword "Layer concept of base objects of working layers".

The files are sorted as follows in these folders:

- All report templates are available in the delivery state in the project-specific folder "WOXXXX ". Example:
"SO1\Overlays\WOXXXX".
- The directory structure under SO1 / Reports is empty.
- If a report template is modified in the CRA, this modified report template is stored automatically in the SO1/Reports directory. The original remains in the WOXXXX folder.
- COMOS automatically uses the new modified template even if, for example, the main report is still located in the WOXXXX folder.
- All customer-specific templates continue to be stored in the SO1 / Reports directory.
- No manual interventions are allowed in the WOXXXX directories of MRA and UCL.

Objective

A unit can have various statuses. The statuses can be defined and rated using different design cases. You can only access one value per design case directly in the user interface. You can access all values in scripts.

Procedure

There are two different views of a unit:

- Operator view (Owner/Operator (OO))
The OO talks about run cases.
- Scheduler view (Engineering, Procurement, and Construction (EPC)).
The EPC designs a unit for different design cases.

In each individual case, the attributes of an object can take on different values. Depending on the task, only individual values or the values for each design case of an attribute are to be displayed.

There are two approaches to design cases in COMOS:

- Managing the case variants using simulation case objects
Main application: FEED module
You can find additional information on this topic in the "FEED Operation" manual, keyword "Managing design cases".
Simulation case objects are created under an object with the names of the individual design cases. The attributes that are relevant at this object for a design case exist at all simulation case objects. The software creates a fully dynamic linking between the attribute at the object and the attribute at the active simulation case object.
An individual tab at the FEED objects is responsible for switching between the design cases at the individual object or across areas. This means all values of the design cases are available at the same time with direct access and can be easily reached with object queries.
- Managing case variants using project properties and by switching the attributes
Obsolete. Only continues to be available for compatibility reasons.
Design cases can be defined in the project settings. You can store information on whether each attribute can take different design cases into consideration on the attribute itself. An individual value can be stored for each design case and for each attribute. If you have defined design cases in the table, a list is displayed in the toolbar in which you select the design case that is to be in effect for the entire project. The design cases are switched throughout the entire project for all attributes.

Administration of templates

Requirement

- You are familiar with the general chapter on administering templates.
You can find additional information on this topic in the "COMOS Administration" manual, keyword "Administering templates".

Overview

Templates for iDB are in preparation.

Database update

A database of the iDB type can be updated using an COMOS database update.

You can find additional information on this topic in the "COMOS Administration" manual, keyword "Objective and general conditions of database update".

