OfficeLink

Release 15.5.0.9

CONTACT Software

Sep 25, 2018

Contents

CHAPTER 1

Introduction

CONTACT OfficeLink provides a high level of integration for the Microsoft Office tools Word, Excel, PowerPoint, Outlook, Visio and Project. This integration allows you, for example, to search the system for documents, open the documents in order to edit them and then save the changes in the database. Document status and attributess can also be modified and new documents can be created in the system directly from Microsoft Office.

Table 1.1: List of OfficeLink application options

	Excel	PowerPoint	Word	Project	Visio	Outlook
Document search in CDB	X	X	X	X	X	X(1)
Save changes to CDB	X	X	X	X	X	
Create new in CDB	X	X	X	X	X	X
Create index in CDB	X	X	X	X	X	
Change CDB status	X	X	X	X	X	
Display CDB document properties	X	X	X	X	X	
Synchronization with CDB metadata	X	X	X		X	
Synchronization with CDB projects				X		
PowerReports	X					

Note:

1. For attaching a document from the system to an e-mail.

Important: Not all supported Office applications have to be installed. However, we do not advise using different versions of Office simultaneously.

Note: *CONTACT OfficeLink* can only be used with a Windows client. If you work with *CONTACT OfficeLink*, only one Windows client may be open.

CHAPTER 2

Excel

2.1 Introduction

- *Excel user interface:*

  The installation extends the Excel application by adding the OfficeLink toolbar.

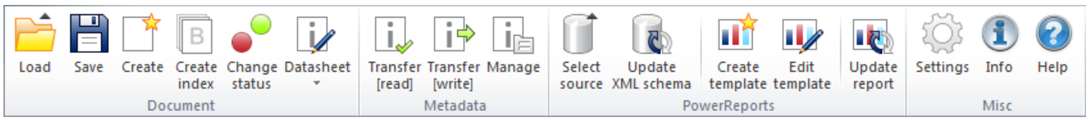


Fig. 2.1: Excel: OfficeLink menu bar

- *Standard operations:*

  The system Microsoft Office integration provides Excel with the option of searching for documents in the system, opening them for editing and then saving changes in the database. Moreover, it is possible to change the document status or properties, or create new documents in the system directly from Excel.

- *Data synchronization:*

  The link gives you the ability to synchronize data with the system. A prerequisite for this is the use of prepared document templates which contain corresponding system variables (see *Create templates for data synchronization* (page 4)). Data synchronization is always carried out for the active document. The active document has to have been opened from the system for editing.

- *PowerReports:*

  PowerReports provide an infrastructure to create reports with Microsoft Excel quickly and easily. The architecture enables the far-reaching use of Microsoft Excel functions in system reports. A great advantage of the PowerReports architecture, for example, is the ability to integrate graphics and diagrams in system reports to visualize information based on the product and project data automatically and appealingly.

- *Settings:*

  The OfficeLink administration manual includes the settings options in the option dialog. Contact your system administrator if you have questions.

2.2 Editing documents

2.2.1 Searching for documents in the system

The *[Load]* button opens the document search mask in the system. The specified search condition automatically limits the search to Excel documents. The result list opens with a preview window after entering additional search criteria and running the search. After selecting one or multiple documents, they are opened for editing.

2.2.2 Save in the system

The *[Save]* button saves the changes to the active document directly in the system.

2.2.3 Create in the system

The *[Create]* button carries out a document creation in the system for the active document. This process initially opens the mask for creating documents. After filling out and confirming the create mask, the document is committed to the system and finally re-opened from the system for editing. The original document is closed after the create is successful. In the case of an error, for example due to deficient authorizations in the system, the document remains open.

Note: Since multiple file formats are supported, the save format is determined there first, before the system document management create mask is opened: *Use as default format:* Enabling this check box defines the selected format as the default save format. The selection dialog is no longer shown when creating documents in the future and the default save format is used instead. The default save format can be changed or removed in the settings dialog at any time. *Excel Open XML file format and macros:* No macros can be saved in the .xlsx Open XML file format of Excel. When attempting to save documents with macros in this format, a corresponding message is displayed.

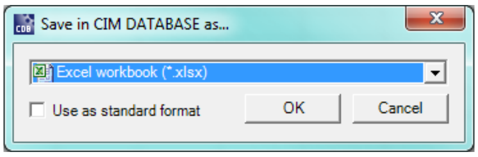


Fig. 2.2: Microsoft Excel: Save format selection dialog

2.2.4 Create index in the system

The *[Create index]* button creates a new index in the system for the active document. This process initially opens the mask for creating indices. After filling out and confirming the index mask, the current changes are backed up in the new document index in the system.

2.2.5 Change document status

The *[Change status]* button carries out a status change operation in the system for the active document.

Note: Since the user potentially loses the rights to re-save the document in the system by changing the status, the link saves the document automatically before calling up the status change operation. If necessary, a corresponding message is displayed here that you have to confirm.

2.2.6 Opening master data change dialog in the system

The *[Master data]* button opens the dialog in the system for changing the master data for the active document. This button is divided into two parts. The bottom part can be used in the system to open an information dialog for the active document instead of the modify dialog.

2.3 Synchronizing data

2.3.1 Metadata takeover from the system

By pressing the *[Transfer [read]]* button, the values of all system variables with read access are updated with the corresponding values of the associated system attributes. This overwrites system variables with read access changed by the user in the document. If an error exists in the expression of a variable, the referenced object does not exist or the relationship is not unique, the value is set to a space character.

2.3.2 Metadata synchronization with the system

By pressing the *[Transfer [write]]* button, the values of all variables with write access are checked for changes. If values have been changed in the document, the affected objects are opened in the system sequentially. The changes are shown in the modify dialogs and have to be confirmed for each object. After completing the data transfer to the system, a metadata acceptance is carried out for system variables with read access.

2.3.3 Create templates for data synchronization

This chapter describes creating document templates with system variables for data synchronization.

- Synchronization of document metadata (e.g. title, document No., etc.)
- Synchronization of 1:1 referenced objects (e.g. project name of the assigned project)
- Transfer 1:N referenced objects (e.g. list with open issues)

Format description for system variables

System variables in Excel documents have the following format: cdb.<Mode>.<Relation>.<Attribute>.<Cardinality>

Mode The mode indicates whether a variable may be read from the system exclusively or updated after changing in the document in the system. Valid values for the mode include:

- *r*

  Only data transfer from the system possible. The transfered value can be changed in the document, but not transferred to the system. The changed value in the document is replaced again by the sytem value when transferring the data again.

- *w*

  Only data transfer to the system possible. This mode can be used to ensure that a document field cannot be changed by a data transfer, though an update is possible in the system (useful for fields calculated by the document). This mode cannot be used for 1:N relationships.

- *rw*

  Data transfer from the system and update in the system possible after changing in the document. This mode cannot be used for 1:N relationships.

- **s*

  If a *s* is added to the above-mentioned mode values (i. e. *rs*, *ws* or *rws*), then the same rules apply as for the respective values without *s*, but the read/write actions run completely on the server side. The *Parameter* field can be used optionally. You might, however, need administrative customizing to use server-side editing of variables, which is explained in the OfficeLink Administration Manual.

Relationship Starting from the document, the relationship indicates the path for the data in the system represented by the variable. The following expressions can be used as a relationship:

- *this*

  The constant expression 'this' is used to specify the attributes of the document itself.

- *Name of a relationship configured in the system*

  Attributes of referenced objects can be transferred and synchronized via a relationship.

- *BY_ZNUM_ZIDX_FROM_<table_name>*

  Use an implicit relationship via the attributes z_nummer and z_index. To do so, the expression BY_ZNUM_ZIDX_FROM_<table_name> is specified. table_name indicates the database table used to reference an object implicitly via z_nummer and z_index. The specified database table must contain the attributes z_nummer and z_index and the combination of z_nummer and z_index must be unique.

Attribute The attribute specifies an attribute of a system object referenced by the relationship.

Subject characteristics as attributes of parts:

If a part relationship is specified as a relationship, the subject characteristics of the part can also be specified as an attribute. This can be carried out in a qualified manner via property identification or unqualified using the sequence number:

- *Qualified subject characteristics:*

  `_SML_<mm_mk>`

Excel: data synchronization - qualified subject characteristics

Subject characteristics with the identification 'L' of the associated part

```
"cdb.r.cdb_doc_to_part._SML_L"
```

- *Unqualified subject characteristics:*

  `_SMLN<nummer> for characteristic identifications`

  `_SMLV<nummer> for characteristic values`

  Property identification and the associated value must have the same number for this.

Excel: data synchronization - unqualified subject characteristics

Identification and value of the first subject characteristic of the associated part

```
"cdb.r.cdb_doc_to_part._SMLN1"
"cdb.r.cdb_doc_to_part._SMLV1"
```

Important: SC attributes cannot be synchronized with write access!

Cardinality Specifies the cardinality of the relationship. If no cardinality has been specified, cardinality 1 is assumed. The cardinality should be specified explicitly when creating new document templates. The following values can be used for the cardinality:

- *1*

  For relationships of the type `this,BY_ZNUM_ZIDX_FROM_<table_name>` and for 1-digit homogeneous relationships.

- *N*

  For n-digit homogeneous relationships. Variables that use n-digit relationships must be positioned by column in the header of a list (see usage of system variables in the document). When synchronizing with the system, the list is adapted to the number of rows required for displaying the result. The first line after the header is used as a style sheet for this. The list is sorted based on the first column.

Excel: data synchronization - system variables

Document number of the document:

```
"cdb.r.this.z_nummer"
```

Transfer and synchronization of the document remark:

```
"cdb.rw.this.z_bemerkung"
```

Transfer of the part number of the associated part:

```
"cdb.r.cdb_doc_to_part.teilenummer"
```

Transfer and synchronization of the part name of the associated part:

```
"cdb.rw.cdb_doc_to_part.benennung"
```

Transfer of the reason of the change history via implicit relation:

```
"cdb.r.BY_ZNUM_ZIDX_FROM_aenderung.begruend"
```

Synchronization of floating points to the system The Excel decimal separator depends on the region and language options configured in the Windows control panel. For attributes defined in the system as *float*, before synchronizing, you have to change the decimal separator from a comma to a dot, where necessary. *.numeric* must be appended to the variable expression of the document variables so that the link recognizes these types of attributes and can correct the decimal separator.

```
cdb.rw.cdbpcs_doc2project.effort_plan.1.numeric
```

Parameter This field can be used to pass additional constant values specific to the respective variable when processing variables on the server side. These can be identifiers, for example, that allow you to link results from 1: N relationships with individual document fields instead of dynamic lists. For example, the parameter *PRIMARY* is added to the following variable, which tells a user exit that you want to have the original source name of the document's primary file.

```
cdb.rs.document2cdb_file.cdbf_original_name.N.string.PRIMARY
```

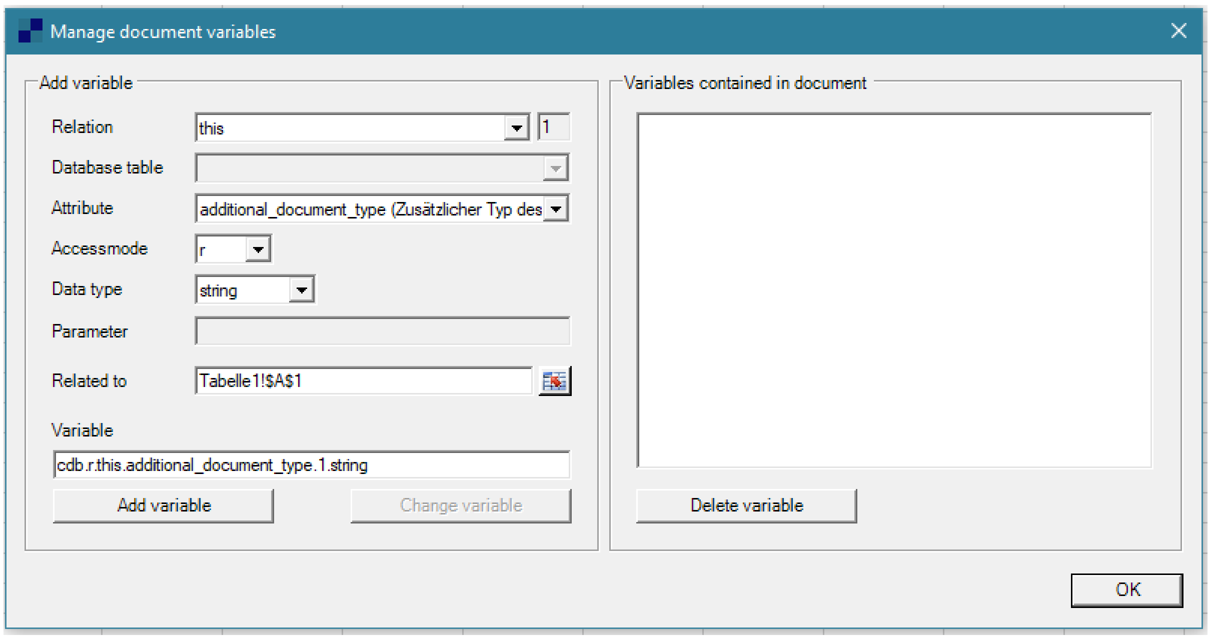
Definition of system variables

System variables are defined using a wizard that calls up the OfficeLink toolbar using the *[Manage]* button. The button has to have been enabled beforehand in the settings dialog so that the button is available. The dialog provides the option of creating new variables in the sytem format on the left side (see *Format description for system variables* (page 4)). On the right side, all system variables contained in the document are displayed. Marking a variable also marks the corresponding cell in the Excel table at the same time. The marked variables are removed from the document by pressing the *[Delete variable]* button.

- *Relationship:*

  Starting from the document, the relationship indicates the path for an attribute of one or more referenced sytem objects. The catalog provides all document relationships configured in the system, for which the document is the start object. For attributes of the document itself, you have to enter `this` in this field.

  To use an implicit relationship that is produced via the key attributes document number and index, you have to select `BY_ZNUM_ZIDX_FROM_` in this field and select a relation as a data source in the database table field.


Manage document variables

Add variable

Relation: this | 1

Database table:

Attribute: additional_document_type (Zusätzlicher Typ des

Accessmode: r

Data type: string

Parameter:

Related to: Tabelle1!A1

Variable: cdb.r.this.additional_document_type.1.string

Fig. 2.3: Excel: data synchronization - dialog for creating system variables

- *Database table:*

  A database table has to be selected only if the special entry `BY_ZNUM_ZIDX_FROM_` for implicit relationships has been selected in the relationship field. In the selected relation, the combination of the attributes z_nummer and z_index has to be unique.

- *Attribute:*

  The attribute specifies an attribute of a system object referenced by the relationship.

- *Synchronization mode:*

  The mode indicates whether a variable may be read from the system exclusively or may be updated after changing in the document in the system.

  r: read only

  w: write only

  rw: read and write access in the system

  rs: server-side read only

  ws: server-side write only

  rws: server-side read and write access in the system

- *Parameter:*

  Optional value used only for server-side editing of variables.

- *Relates to:*

  In this field, you have to specify the cell in which the system variable is to be displayed. The Excel cell should be selected from the catalog. When opening the catalog, the dialog for the variable configuration is hidden and a selection window appears for the Excel cell. The Excel cell can be selected and then committed by clicking the desired cell.

- *System variable:*

  In this field, the expression generated from the other fields is shown for the system variables to be created. The manual modification of this expression is only required for specifying subject characteristics for part relationships.

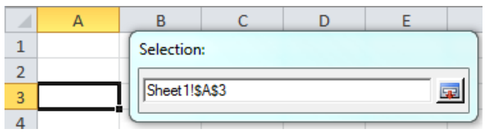


Fig. 2.4: Excel: data synchronization - selecting a cell

When adding the variables by pressing the *[Add variable]* button, the expression shown in the system variable field is reviewed and displays a corresponding error message if invalid. Otherwise, the variables are created as new system variables in the active document and initialized with the `<Attribut>` value as a placeholder. Cells to which a name refers assume the value of the associated system attribute when updating. A name can also be referenced by additional cells. These cells also assume this value.

Using system variables in the document

System variables should be identified for the user in the document to avoid accidental deleting or overwriting. Document variables can be protected in Microsoft Office Excel by locking the corresponding cells. You have to enable sheet protection so that this protection is active. Here, a password must not be assigned since the link temporarily lifts this sheet protection during the data synchronization. If a password has been specified when the sheet protection was enabled, the user is prompted to enter a password during the data synchronization. If the password is not entered or entered incorrectly, the data synchronization fails.

Note: During the data synchronization of variables to the system, attribute changes are also shown if the field cannot be changed by the user interactively according to the mask configuration or would only be able to be changed using a list of options. You have to take this into account when using System-Variables with write access. If necessary, the validity of the values must be ensured through a corresponding check in the user exit.

System variables that use n-digit relationships have to be positioned in Excel by column in the header of a list. When synchronizing with the system, the list is adapted to the number of rows required for displaying the result. The first line after the header is used as a style sheet for this. The list is sorted based on the first column.

Lists/tables in Microsoft Office Excel for n-digit relationships:

1. Insert lists/tables with two rows and the desired number of columns.
2. Insert system variables into the first row using the wizard (see *Definition of system variables* (page 6)). All variables have to use the same relationship for this and be of the cardinality N.
3. Change the placeholder of the system variables in the desired column titles and format as desired.
4. Format the second row. This line is used as a master copy for pasting additional rows for the data synchronization.

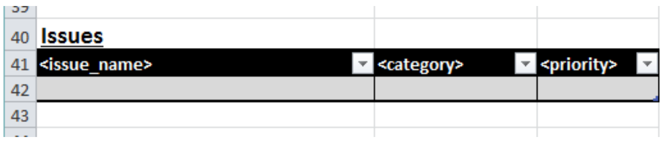
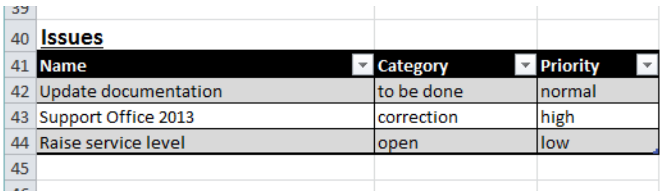


Fig. 2.5: Excel: data synchronization - list/table for n-digit relationships [empty]

When carrying out a data transfer, the table is expanded to the necessary number of rows and filled with the set of results for the relationship used.



Issues

Name	Category	Priority
Update documentation	to be done	normal
Support Office 2013	correction	high
Raise service level	open	low

Fig. 2.6: Excel: data synchronization - list/table for n-digit relationships [filled]

Updating metadata automatically

The metadata of all system variables with read access are committed from CDB automatically when loading a document. The conditions for this are that the document:

1. Has been opened in the client in “Edit” mode (or directly from Office via “Load”)
2. Is a “Drawing” (i.e. has the property “vorlage_kz”)
3. Is not a template (i.e. “vorlage_kz=0”)

2.4 PowerReports

2.4.1 Introduction

The reports retrieve their data from XML sources that can be reused and parameterized, and that are configured in the system. XML sources can use relations of the object model (cdb.objects framework) and configured object rules as so-called data providers. Therefore data sources can also be defined simply by configuring, without additional programming! For example, if a report is to be created for parts that contains the BOM, the associated project and the associated EC, then this report can be created without any programming, since the required relations are already modeled in the object model. For complex queries or computed data, additionally programmed data providers can be linked to an XML source. Thus XML sources can return any heterogeneous data from various data providers.

Cells in a PowerReport can also contain hyperlinks that refer directly to the object in question in the system. With the single-sign-on options, the System and the Office world connect seamlessly with one another.

Reports are created directly in Excel with support from the system Office integration. After selecting the desired XML source, the XML schema required for the data connection is generated automatically from the configured data source and loaded in Excel. Then the data connection can be established by dragging and dropping and the layout can be done.

New reports are saved to the XML source as a report template via Office integration. Thus the report templates are managed as documents in the system and assigned to the underlying data source. With the new infrastructure, newly created reports no longer have to be configured individually in the system for them to be available on the user interface. Instead, it is sufficient to approve the reports by assigning authorized roles or persons.

2.4.2 Architecture

PowerReports retrieves its data from XML sources that are configured in the database. An XML source consists of any number of reusable data providers that provide the data to be represented. A data sources provides two types of information in the form of temporary files that are calculated from the sum of the assigned providers. The first is the XSD schema for establishing the data connection in the report templates (represented in red in the figure below), the second is the data in the form of an XML export file.

The XML export file is generated when calling up a report and committed to the client. At the same time, the Excel template saved in the system vault for the report is instantiated and committed to the client. The client features an active component through the MS Excel integration that compiles the instantiated template and the XML export file into a completed report.

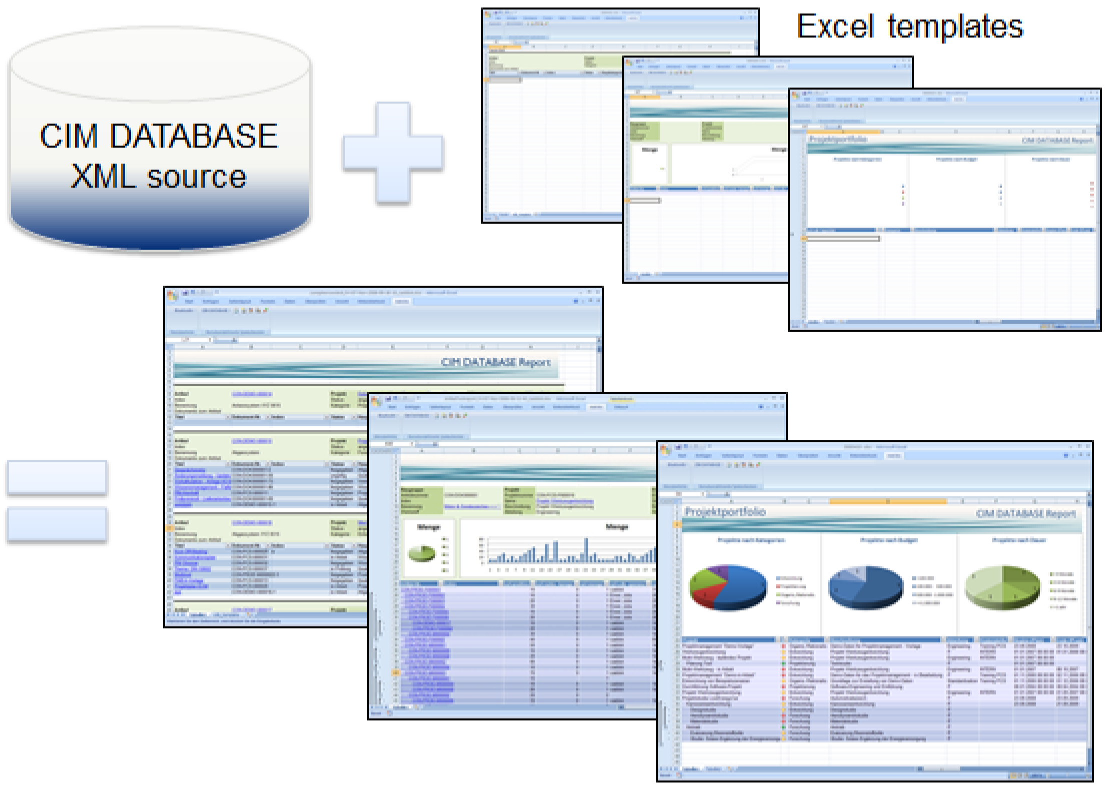


Fig. 2.7: Excel: PowerReports - schema

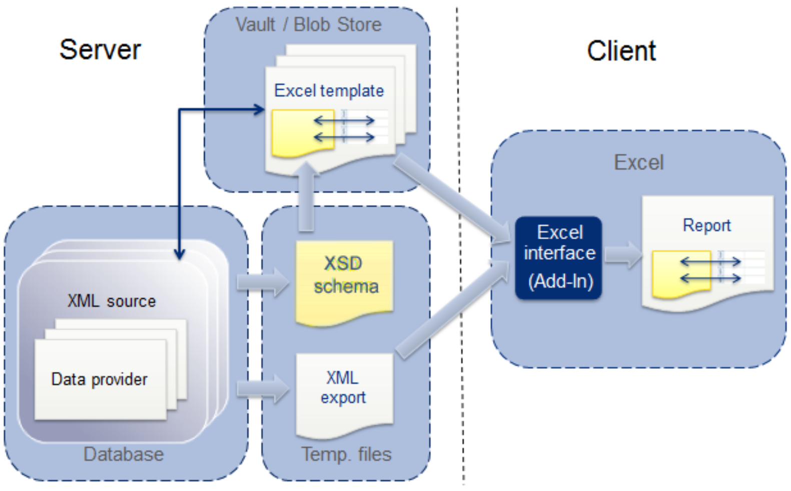


Fig. 2.8: Excel: PowerReports - architecture overview

The XSD schema of a data source is calculated from the sum of the part schemata of the assigned providers. Here, the data providers generate the schema from the underlying database tables in an automated way and add additional attributes if necessary. The schema of a data provider can also be generated completely independent of the database. This is necessary, for example, if a provider only provides calculated data or data from external systems.

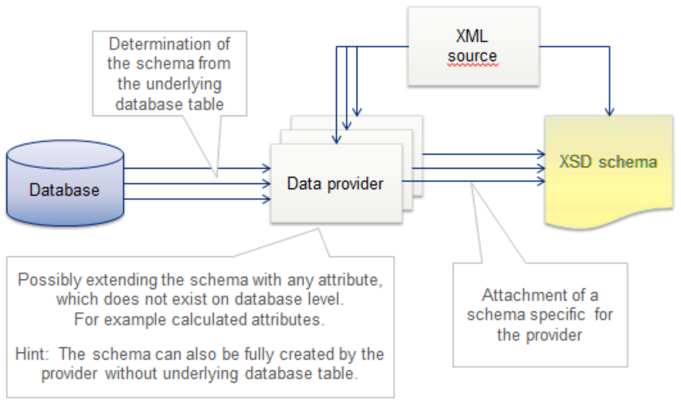

Determination of the schema from the underlying database table
XML source
XSD schema
Database
Data provider
Possibly extending the schema with any attribute, which does not exist on database level. For example calculated attributes.
Hint: The schema can also be fully created by the provider without underlying database table.
Attachment of a schema specific for the provider

Fig. 2.9: Excel: PowerReports - functional principle for schema export

A report can be called up context-free or context-related depending on the data source configuration. A context-free report typically provides data from the total stock of data, for example from all running projects. A context-related report, on the other hand, provides data that relates to one or multiple selected objects, for example information for a project or engineering change. The context results by calling up one or multiple selected objects via the pop-up menu. The selected objects set parameters for the data export.

In addition to setting parameters using the context, a dialog requesting additional parameters from the user can precede this. The selected objects and additional parameters from a preceding dialog are illustrated as 'objects and parameters' in the figure below. The objects and parameters are transferred to the data providers during the XML export. The data providers determine the set of results from the database based on these arguments and design ReportData objects from this, which are then exported to a shared XML file under the consideration of the rights system.

Data providers can be linked to each other through a hierarchical arrangement. In this case, a subordinate provider does not receive the input object of the XML source as arguments, but rather the set of results of the higher-level provider. This makes it possible to set parameters for data providers based on the result of another provider.

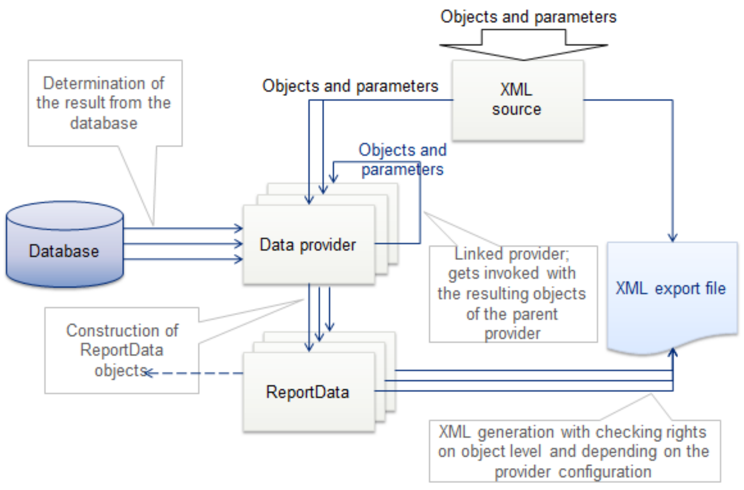

Objects and parameters
Determination of the result from the database
Objects and parameters
XML source
Objects and parameters
Database
Data provider
Linked provider; gets invoked with the resulting objects of the parent provider
XML export file
Construction of ReportData objects
ReportData
XML generation with checking rights on object level and depending on the provider configuration

Fig. 2.10: Excel: PowerReports - functional principle for XML export

2.4.3 Data sources and data providers

PowerReports retrieve their data from XML sources that are configured in the system under *Administration/Configuration* → *Configuration* → *XML data sources*.

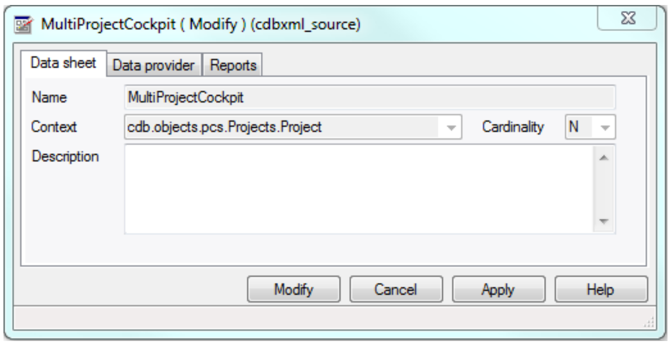
XML sources can deliver context-free or context-related data and, based on these properties, are typically distinguished as two different types:

1. *Context-related data sources*

   Context-related data sources deliver data for a project, EC or a part, for example. Evaluating context-related XML sources always requires at least one object that forms the context. This, for example, includes tasks and open issues for the project. In this example, the project forms the context. Reports based on context-related data sources are always recalled via the pop-up menu of the context-forming object. A report called 'project overview' that contains detail information related to a project is also recalled from the pop-up menu of a project.

2. *Context-free data sources*

   Queries about the entire data stock are an example of context-free data. Evaluating context-free data sources does not require a context-forming object. Reports based on context-free data sources are recalled at a central location from the OfficeLink menu bar under *Settings* → *Additional settings...* → *Reports*.


MultiProjectCockpit (Modify) (cdbxml_source)

Data sheet | Data provider | Reports

Name: MultiProjectCockpit

Context: cdb.objects.pcs.Projects.Project

Cardinality: N

Description:

Fig. 2.11: Excel: PowerReports - Properties of data sources.

Name Unique name of an XML source.

Context Determines whether the data source is context-free or context-related. This field remains empty for context-free data sources. For context-related data sources, the object type within which the context of the data source is to be evaluated is specified in the form of a fully qualified Python name of the associated cdb.objects class. For example, `cdb.objects.pcs.Projects.Project` for a data source that can be evaluated within the context of projects. When creating a new data source all root classes are available via the drop down list. When using Python subclasses the fully qualified Python name must be entered manually.

Cardinality Specifies the cardinality of the context. This field is not evaluated for context-free data sources. For context-related data sources, determines the cardinality and whether the data source can be evaluated based on exactly one context-forming object (single-select) or based on any number of context-forming objects (multi-select). Possible values are 1 or N.

Description Description of the data source.

Data providers

Assigned data providers deliver the data of an XML source. A data source can consist of any number of data providers. You can configure parameters for the data providers, which can be reused.

Data providers are classified into two different groups:

1. *Universal data providers*

Universal data providers are predefined data provider implementations for the most frequent use cases, usable in any data sources without programming, and configurable within their context. The following types of universal data providers are available:

- *Relationship*

  for navigating relationships of the cdb.objects framework
- *Rule*

  for configured, complex queries
- *SimpleQuery*

  for simple, interactive queries
- *GroupBy*

  for grouping the set of results of other providers

2. *Specialized data providers*

   Specialized data providers are customer-specific provider implementations for special use cases. Specialized data providers can also be reused in any data sources. The implementation takes place by deriving a predefined base class and by implementing an interface. Specialized data providers are always of type `CustomCode`.

Context-related data sources automatically contain a special data provider that delivers the context-forming object or objects. As follows, this data provider is referred to as a context provider. The following mapping shows the context provider based on the XML source 'project overview.' The project, from which an underlying report is recalled, is thereby automatically included in the exported XML data and can be presented in the report.

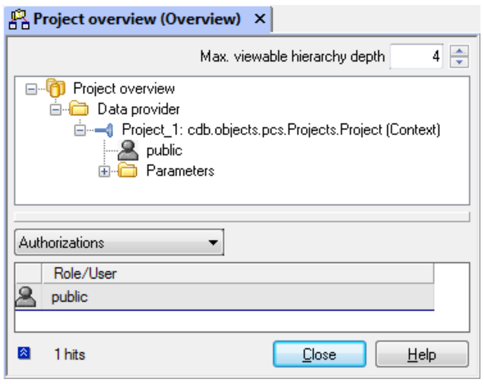

Project overview (Overview)
Max. viewable hierarchy depth 4
Project overview
Data provider
Project_1: cdb.objects.pcs.Projects.Project (Context)
public
Parameters
Authorizations
Role/User
public
1 hits
Close
Help

Fig. 2.12: Excel: PowerReports - Context provider of a project-related report

Data providers can be arranged hierarchically. Within this context, subordinate data providers always work with the set of results of the higher-level data provider. The set of results for the higher-level providers also serves as an entry parameter for the subordinate provider. In this manner, the results of the higher-level provider can, for example, be grouped with a GroupBy provider or navigate relations through the object model.

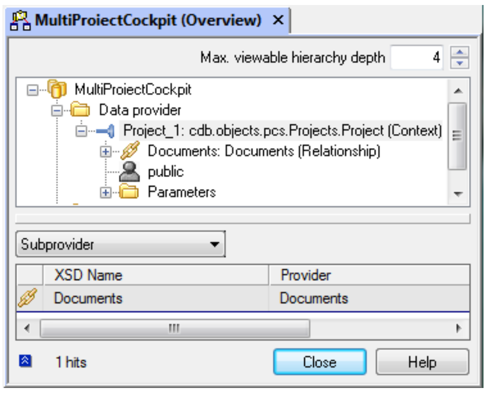
The following illustration again shows the example 'project overview', expanded by adding a relationship provider that delivers the documents for the project. The context provider thus serves as a higher-level provider and, in this manner, provides the start object (the project) for navigating relations.

Properties of data providers

- *XML source*

  Foreign key of the associated XML source. Automatically prepopulated and not changeable.
- *Type*

  Type of data provider (see *Types of providers in detail* (page 14)).



MultiProiectCockpit (Overview)

Max. viewable hierarchy depth 4

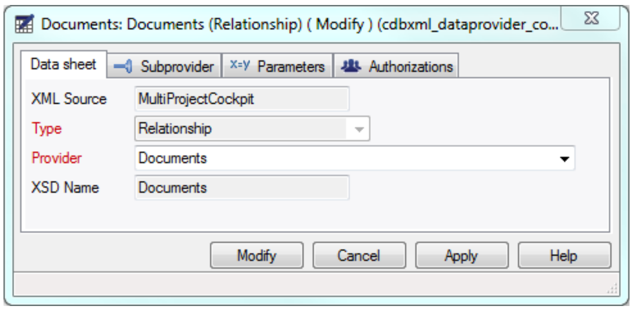
- MultiProiectCockpit
  - Data provider
    - Project_1: cdb.objects.pcs.Projects.Project (Context)
      - Documents: Documents (Relationship)
      - public
      - Parameters

Subprovider

XSD Name	Provider
Documents	Documents

1 hits

Fig. 2.13: Excel: PowerReports - Relationship subprovider for the context provider



Documents: Documents (Relationship) (Modify) (cdbxml_dataprovider_co...

Data sheet | Subprovider | Parameters | Authorizations

XML Source	MultiProjectCockpit
Type	Relationship
Provider	Documents
XSD Name	Documents

Modify | Cancel | Apply | Help

Fig. 2.14: Excel: PowerReports - Properties of data providers

- *Provider*

  Depending on the provider type, this field specifies the data origin (see *Types of providers in detail* (page 14)).

- *XSD name*

  A unique namespace within the context of the data source. Depending on the provider type, it can also be automatically prepopulated. After creating a provider, it cannot be changed.

Types of providers in detail

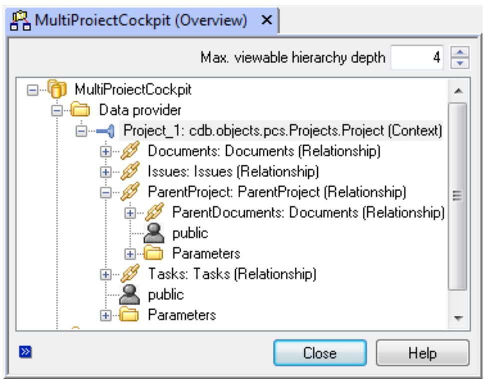
- *Relationship*

  Data providers of the type Relationship provide objects that are determined by navigating a relationship of the cdb.objects framework. As navigating a relationship always requires a start object, relationship providers can exclusively be used as subproviders. Relationship providers can be assigned to all providers that deliver cdb.objects objects as results.

  The name of a relationship that can be navigated starting from the results object of the higher-level providers is entered in the field `Provider`. When creating a relationship provider, the respectively available relations can be chosen from a catalog. When the data source of the parent provider is only Python subclass, then the available relations aren't listet automatically, but must be entered manually.

  By linking relationship providers, data from the entire object model can be determined starting from the start object. The following example demonstrates the use of relationship providers starting from a project. For example, the parent project and all documents that belong to this project are determined.

  The use of a relationship provider on a higher-level provider that returns a n-digit result leads to a multi-export, i.e. the relationship is navigated individually for each result object of the higher-level provider and exported together with the start object. For data sources with multi-export configurations, a special template sheet in the Excel template is required (see *Displaying complex rows for multi-export data sources* (page 39)).


MultiProiectCockpit (Overview)
Max. viewable hierarchy depth 4
Close
Help

```
MultiProiectCockpit
  Data provider
    Project_1: cdb.objects.pcs.Projects.Project (Context)
      Documents: Documents (Relationship)
      Issues: Issues (Relationship)
      ParentProject: ParentProject (Relationship)
        ParentDocuments: Documents (Relationship)
        public
        Parameters
      Tasks: Tasks (Relationship)
      public
      Parameters
```

Fig. 2.15: Excel: PowerReports - relationships as provider

The set of results of a relationship provider, depending on the cardinality of the relationship, is either an `ReportData` object or a `ReportDataList`.

- *Rule*

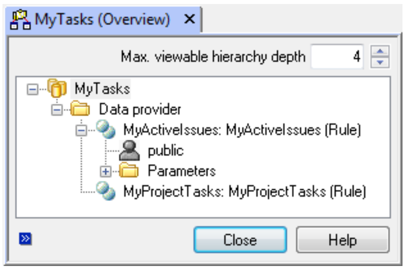
  Data providers of the type Rule provide objects by evaluating a configured rule that describes the objects declaratively. Using configured rules, both simple and more complex queries can be run incorporating properties of referenced objects. For more detailed information about configurable rules, refer to the 'Configurable rules' section of the administrator manual.

  Rule providers can be used without higher-level providers or as a subprovider. In both cases, the rule's parameters can be configured by using variables. The variables can be queried by the user using an upstream dialog. The following is an example of a rule-based provider without a superordinate provider. For example, if a rule named *My Projects: By Category* is to provide projects with a project category defined in a report dialog and the term *category=$(category)* is defined for this purpose, then an attribute named *myprojectsbycategory-category* must be included in the report dialog. Since the variable name *$(category)* in this example corresponds to an actual object attribute name, this variable rule filter can also be linked to and updated in the report itself using an argument attribute that appears automatically in the XML schema. When used as a subprovider, an attempt is made to populate the variables from the attributes of the result object of the higher-level provider unless these are defined by a possible upstream dialog. An example of a term for a rule-based subprovider that is to provide tasks to the superordinate project provider would be *cdb_project_id=$(cdb_project_id)*.

  In the 'Provider' field for rule providers, enter the name of the rule to be used.

  The sorting of the set of results can be determined using the parameter *OrderBy*. The sorting is defined using a comma-separated list of the database attributes according to which the search is to be run. For sorting in descending order, the respective attribute is prefixed with a minus sign.

  The following illustration shows a data source with two rule providers for determining project tasks to be processed and uncompleted open issues for which the logged-in user is responsible:

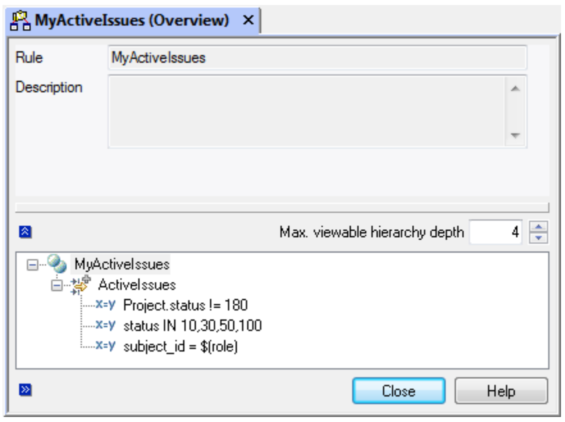


```
MyTasks
  Data provider
    MyActiveIssues: MyActiveIssues (Rule)
      public
      Parameters
    MyProjectTasks: MyProjectTasks (Rule)
```

Fig. 2.16: Excel: PowerReports - provider for rule-based determination of tasks

  The definition of rules for determining the open issues could look like this:

  The set of results of rule providers is always a *ReportDataList*.



MyActiveIssues (Overview)

Rule: MyActiveIssues

Description:

Max. viewable hierarchy depth: 4

```
MyActiveIssues
  ActiveIssues
    x=y Project.status != 180
    x=y status IN 10,30,50,100
    x=y subject_id = $(role)
```

Fig. 2.17: Excel: PowerReports - configured rules for determining a user's open items

- *SimpleQuery*

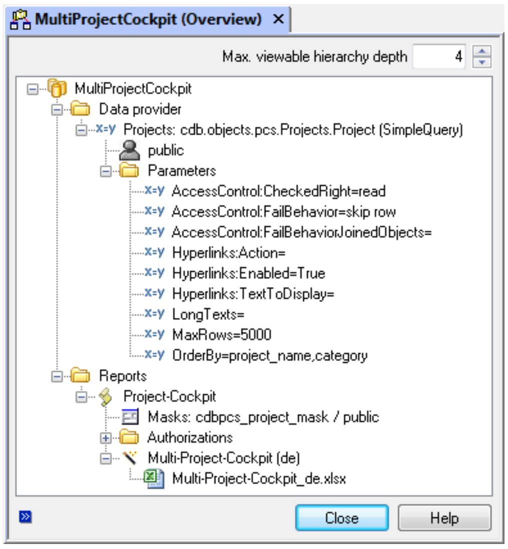
  Data providers of the type SimpleQuery provide objects through queries to a database table, similar to a system search. SimpleQuery providers are used context-free, meaning that they are not used as subproviders, but always at the top level for the data source.

  In the 'Provider' field, enter a cdb.objects class that defines the type of objects to be provided. The cdb.object class thus also defines the database table on which the search is run.

  SimpleQuery providers return the entire data stock of the respective database table without constraints through search conditions. The search conditions to be used are queried by the user for SimpleQuery Providers by an upstream report dialog. For each report based on a data source, this dialog can make different search conditions available for selection or determine them in fixed fashion. Regardless of the search conditions used, the maximum number of results can be restricted using the *MaxRows* parameter. If this number is exceeded, an error message is displayed and the report is not generated.

  The sorting of the set of results can be determined using the parameter *OrderBy*. The sorting is defined using a comma-separated list of the database attributes according to which the search is to be run. For sorting in descending order, the respective attribute is prefixed with a minus sign.

  The following illustration shows a data source with a SimpleQuery provider for projects with a defined sorting and the set of results limited to 5000. The other parameters displayed also apply for all other provider types and are described in detail in *Provider parameters* (page 18).



MultiProjectCockpit (Overview)

Max. viewable hierarchy depth: 4

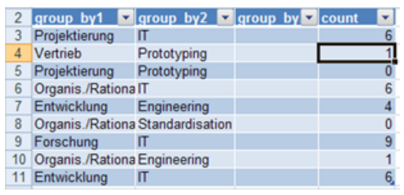
```
MultiProjectCockpit
  Data provider
    x=y Projects: cdb.objects.pcs.Projects.Project (SimpleQuery)
      public
      Parameters
        x=y AccessControl:CheckedRight=read
        x=y AccessControl:FailBehavior=skip row
        x=y AccessControl:FailBehaviorJoinedObjects=
        x=y Hyperlinks:Action=
        x=y Hyperlinks:Enabled=True
        x=y Hyperlinks:TextToDisplay=
        x=y LongTexts=
        x=y MaxRows=5000
        x=y OrderBy=project_name,category
  Reports
    Project-Cockpit
      Masks: cdbpcs_project_mask / public
      Authorizations
      Multi-Project-Cockpit (de)
        Multi-Project-Cockpit_de.xlsx
```

Fig. 2.18: Excel: PowerReports - SimpleQuery provider

SimpleQuery providers always return a *ReportDataList* as the result.

- *GroupBy*

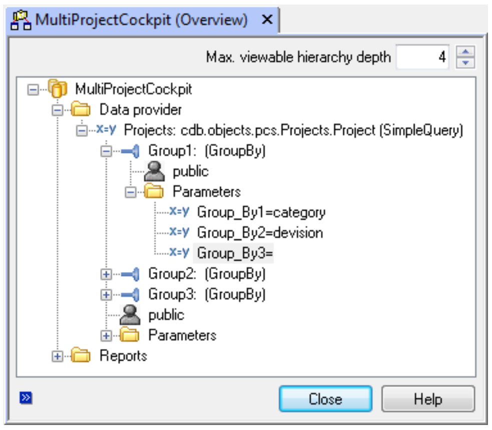
  GroupBy providers group the results of other providers based on configurable attributes or value ranges. They are always used as subprovider for providers who return a *ReportDataList* as the result. In doing so, group functions (sum, average, min, max, count) can be defined for all numerical attributes. This makes the results tables from GroupBy providers particularly well suited for creating diagrams.



2	group_by1	group_by2	group_by	count
3	Projektierung	IT		6
4	Vertrieb	Prototyping		1
5	Projektierung	Prototyping		0
6	Organis./Rationa	IT		6
7	Entwicklung	Engineering		4
8	Organis./Rationa	Standardisation		0
9	Forschung	IT		9
10	Organis./Rationa	Engineering		1
11	Entwicklung	IT		6

Fig. 2.19: Excel: PowerReports - Results table of GroupBy providers

The following figure provides an example of the grouping of projects returned by a higher-level SimpleQuery provider. The criteria for grouping is defined by the parameters Group_By1, Group_By2 and Group_By3. In the simplest case, these parameters are assigned attribute names that are to be used as the basis for grouping. Group functions are defined using optional Function1, Function2, etc. parameters. These are each assigned a value pair consisting of a group function and attribute name separated by a colon, e.g. count:* or average:effort_plan (the count group function is available by default in this case, meaning it does not have to be explicitly defined as FunctionX of the provider). Adding several GroupBy providers lets you group the results from a higher-level data provider by different criteria simultaneously. Each GroupBy provider returns its own results table in the process.



MultiProjectCockpit (Overview)

Max. viewable hierarchy depth 4

```
MultiProjectCockpit
  Data provider
    Projects: cdb.objects.pcs.Projects.Project (SimpleQuery)
      Group1: (GroupBy)
        public
        Parameters
          Group_By1=category
          Group_By2=devision
          Group_By3=
      Group2: (GroupBy)
      Group3: (GroupBy)
      public
      Parameters
  Reports
```

Fig. 2.20: Excel: PowerReports - Grouping projects

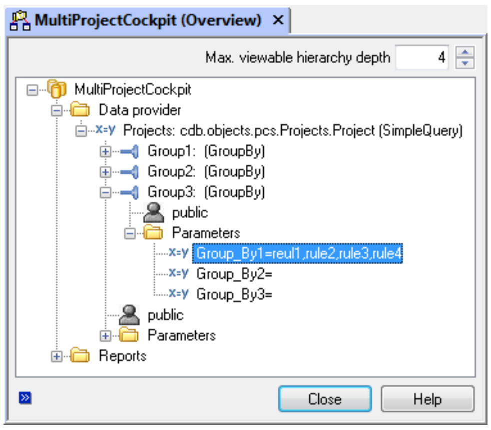
The user can also interactively prompted for the GroupBy_1, GroupBy_2 and Group_By_3 grouping criteria in a preceding report dialog. This allows users themselves to group by any properties. The grouping criteria from the dialog take priority over the criteria defined on the *Parameters* tab.

To group by value ranges, such as by budget for projects, a list of configured rules can be specified instead of an attribute name. In this case, the rules describe the subsets into which the result objects of the higher-level provider are sorted. As shown in the figure below, the rules are specified in parentheses as comma-separated lists.

The grouping rules used in the example specify value ranges for the project budget. A definition of these rules could have the following appearance. You can find more information on configurable rules in the administrator manual in the 'Configurable rules' section.

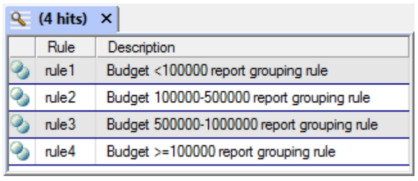
- *CustomCode*

  Data providers of the CustomCode type can determine and return data in any way. For example, CustomCode providers can carry out queries, add computed columns or even return data from external databases.


MultiProjectCockpit (Overview)
Max. viewable hierarchy depth 4
Close
Help

```
MultiProjectCockpit
  Data provider
    Projects: cdb.objects.pcs.Projects.Project (SimpleQuery)
      Group1: (GroupBy)
      Group2: (GroupBy)
      Group3: (GroupBy)
        public
        Parameters
          Group_By1=reul1,rule2,rule3,rule4
          Group_By2=
          Group_By3=
      public
      Parameters
  Reports
```

Fig. 2.21: Excel: PowerReports - Grouping by value ranges with rules


(4 hits)

Rule	Description
rule1	Budget <100000 report grouping rule
rule2	Budget 100000-500000 report grouping rule
rule3	Budget 500000-1000000 report grouping rule
rule4	Budget >=100000 report grouping rule

Fig. 2.22: Excel: PowerReports - Grouping rules using grouping projects by budget as an example

CustomCode providers can construct report data from cdb.objects objects or RecordSets. Furthermore, every aspect of the report data can be freely designed. You can find the implementation of CustomCode providers and a detailed API description in *Implementation of separate providers* (page 20).

- *Context*

  Data providers of the Context type are only created automatically. The system creates them when creating a new context-related data source. A context-related data source always contains exactly one data provider of the Context type. This provider returns the object or objects used to call the report. Subproviders of the Relationship type, for example, can be attached to the context provider to integrate referenced objects into the report.

  Depending on the cardinality of the data source, a context provider returns either a list of objects as *ReportDataList* or a single object as a *ReportData* object.

Provider parameters

Data providers can be configured using parameters. The available parameters depend on the respective provider type and are added automatically with the default values when creating new providers. The parameters described in this section apply to all providers that return data in the form of cdb.objects objects. This also applies to providers of the CustomCode type that you implement yourself.

Rights check at the object level A rights check at the object level can be configured with the following parameters for all data providers that return cdb.objects objects as a set of results.

- *AccessControl:CheckedRight*

  Specifies the right to be checked for each object. The rights check at the object level is disabled if this parameter is not present or is left blank. All of the rights and access right groups in the rights system can be used. The read right is entered by default when creating a universal provider. For large amounts of data, as is possible for providers of the SimpleQuery type, runtime performance deteriorates if the rights check is enabled. If not absolutely necessary, the rights check at the object level should be disabled if there is potential for large amounts of data.

- *AccessControl:FailBehavior*

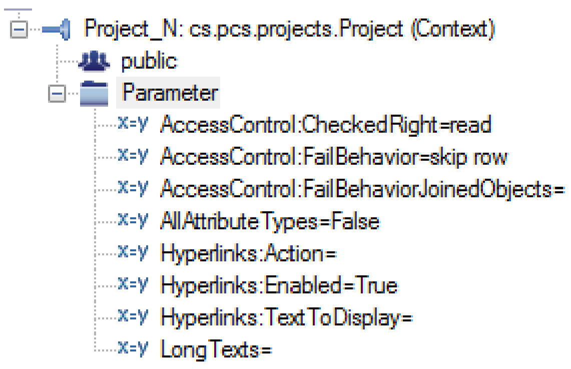

Project_N: cs.pcs.projects.Project (Context)
public
Parameter
x=y AccessControl:CheckedRight=read
x=y AccessControl:FailBehavior=skip row
x=y AccessControl:FailBehaviorJoinedObjects=
x=y AllAttributeTypes=False
x=y Hyperlinks:Action=
x=y Hyperlinks:Enabled=True
x=y Hyperlinks:TextToDisplay=
x=y LongTexts=

Fig. 2.23: Excel: PowerReports - Data provider default parameters

Specifies the behavior in the event of a negative (i.e. failed) rights check. There are two modes available for selection:

- *obfuscate*

  The object attributes are obfuscated before exporting. The key attributes and other added data, such as computed columns, remain legible.

- *skip row*

  The object and, if applicable, any added data are not exported.

Hyperlinks Data providers can export configuration-related hyperlinks that can be used to link to corresponding objects in the system directly from Excel. Hyperlinks are configured using the parameters described below:

- *Hyperlinks:Enabled*

  Enables hyperlinks for the data provider. An additional cdbxml_hyperlink attribute is exported. This can be used in the Excel data connection, automatically linking to the corresponding system object from Excel. Assigning 'true' to this parameter is all that is required to enable hyperlinks. If nothing is assigned to the parameters described below, hyperlinks are generated using the CDB_ShowObject action (displaying the object in an info dialog) labeled with the object description configured in the class directory.

- *Hyperlinks:Action*

  Specifies the system operation to be run when clicking on the hyperlink. All of the operations defined for an object class can be used. This includes system operations such as CDB_Modify and CDB_View, PowerScript operations and GraphView operations such as cdbpcs_project_overview for displaying the project overview. The __default_action__ class variable of the underlying cdb.objects class is evaluated if this field is left blank. If this is also undefined, the CDB_ShowObject operation is used.

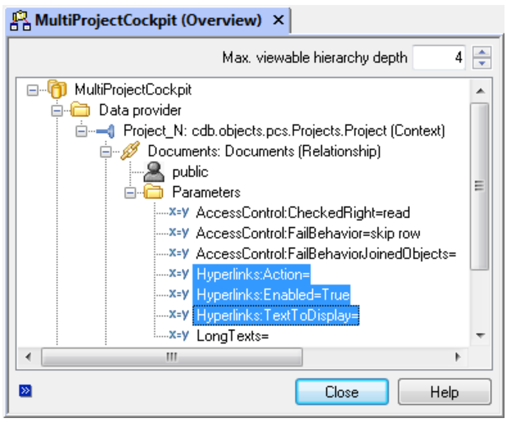
- *Hyperlinks:TextToDisplay*

  Specifies the text to be displayed. You can specify any character string for a constant expression or an attribute name of the underlying cdb.objects class. The configured object description from the system class directory is used if this field is left blank.

LongTexts Data providers can optionally export the longtexts that belong to an object. The longtexts to be exported are specified explicitly by the *LongTexts* parameter for performance reasons. The longtexts to be exported are specified by a comma-separated list of the longtext relations. The entered longtexts are then also available in Excel for establishing the data connection under this name.

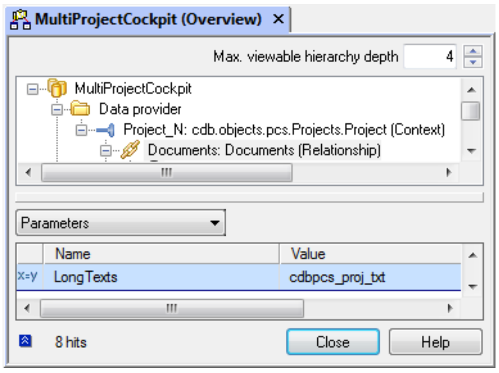
Attribute Types By default data providers always additionally export mapped attributes. When other special attributes like the joined and virtual ones should be exported, then this behavior has to be explicitly activated for every provider. This is done by setting the provider parameter *AllAttributeTypes* to *True*.

Important: For performance reasons the usage of joined and virtual attributes is discouraged for providers


MultiProjectCockpit (Overview)
Max. viewable hierarchy depth 4
Close
Help

```
MultiProjectCockpit
  Data provider
    Project_N: cdb.objects.pcs.Projects.Project (Context)
      Documents: Documents (Relationship)
        public
        Parameters
          AccessControl:CheckedRight=read
          AccessControl:FailBehavior=skip row
          AccessControl:FailBehaviorJoinedObjects=
          Hyperlinks:Action=
          Hyperlinks:Enabled=True
          Hyperlinks:TextToDisplay=
          LongTexts=
```

Fig. 2.24: Excel: PowerReports - Hyperlink configuration and associated schema object


MultiProjectCockpit (Overview)
Max. viewable hierarchy depth 4
Parameters
8 hits
Close
Help

```
MultiProjectCockpit
  Data provider
    Project_N: cdb.objects.pcs.Projects.Project (Context)
      Documents: Documents (Relationship)
```

Name	Value
LongTexts	cdbpcs_proj_txt

Fig. 2.25: Excel: PowerReports - Longtext configuration and associated schema object

that supply many objects.

MaxRowsTruncate The result set of a provider can be limited with this parameter. If a list exceeds the sheet row number 65535 when a list is filled, an XML import error message appears in Excel that no or only some data has been imported.

General access rights at the provider level

Access to the data of a data provider is protected by assigning authorized roles and persons. The ‘public’ role is entered automatically when creating a data provider, meaning there are initially no access limitations at the provider level.

In addition to access limitations at the provider level, providers also carry out a rights check at the object level using the rights system during a data export. The rights check at the object level can be enabled/disabled for each provider. The checked right can be configured.

2.4.4 Implementation of separate providers

Separate data providers can be stored in any Python customer module. In order to be able to assign separate data providers to a data source, the respective Python modules have to be entered into the `default.conf` file, which belongs to the regarding customer module. If the implementations of separate data providers are in the Python module `my_namespace/my_module/PowerReportsCodeSamples.py`, for example, then the my_namespace.my_module.PowerReportsCodeSamples line has to be added to the `my_namespace/my_module/default.conf` file. The entry makes the contained provider classes available during the assignment to the data source.

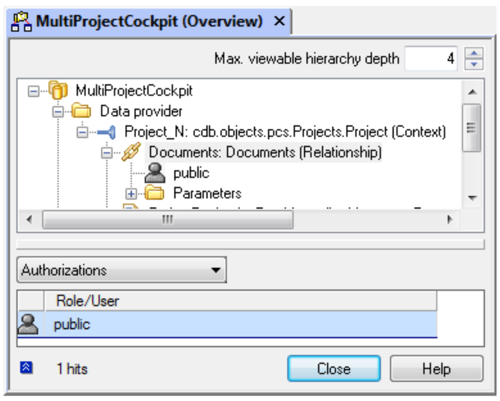
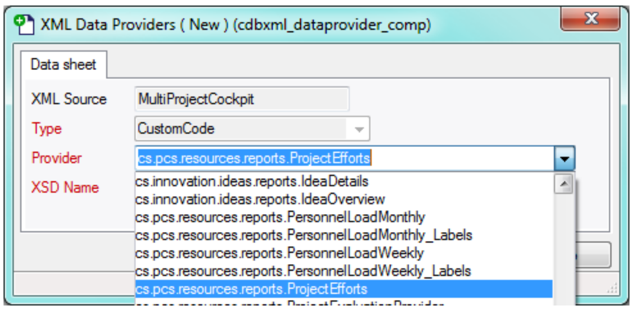

MultiProjectCockpit (Overview)
Max. viewable hierarchy depth 4
MultiProjectCockpit
Data provider
Project_N: cdb.objects.pcs.Projects.Project (Context)
Documents: Documents (Relationship)
public
Parameters
Authorizations
Role/User
public
1 hits
Close
Help

Fig. 2.26: Excel: PowerReports - General access rights at the provider level



XML Source: MultiProjectCockpit
Type: CustomCode
Provider: cs.pcs.resources.reports.ProjectEfforts
XSD Name

cs.innovation.ideas.reports.IdeaDetails
cs.innovation.ideas.reports.IdeaOverview
cs.pcs.resources.reports.PersonnelLoadMonthly
cs.pcs.resources.reports.PersonnelLoadMonthly_Labels
cs.pcs.resources.reports.PersonnelLoadWeekly
cs.pcs.resources.reports.PersonnelLoadWeekly_Labels
cs.pcs.resources.reports.ProjectEfforts

Fig. 2.27: Excel: PowerReports - list of options for separate provider implementations

Separate data providers are implemented by deriving them from the base class `contact.PowerReports.CustomDataProvider`.

```
from cdb.objects.PowerReports import *
class ProjectsProvider(CustomDataProvider):
    ...
```

Two cardinalities have to be specified for a data provider. The cardinality of the set of results and the cardinality for the call. These cardinalities are defined by the two class variables CARD and CALL_CARD.

- *CARD*

  Specifies the result cardinality of a data provider. Valid values are:

  - *CARD_1*

    For data providers that return an individual data object as a set of results.

  - *CARD_N*

    For data providers that return lists as a set of results.

- *CALL_CARD*

  The call cardinality defines, among other things, whether a data provider determines the set of results without context or contextually. Context-related data providers are always assigned to a higher-level provider as subproviders and receive the result of the higher-level provider as the first parameter for determining the separate set of results.

  - *CARD_0*

    For data providers whose set of results is determined without context. For example, a data provider that returns all running projects. Data providers with the call cardinality CARD_0 cannot be assigned as subproviders, but instead are always assigned top level for the data source.

  - *CARD_1*

For data providers that require exactly one data object of the higher-level provider to determine the separate set of results. For example, a data provider that computes the project progress and other KPIs for a project. Data providers with the cardinality CARD_1 are always assigned to a higher-level provider as subproviders. If the higher-level provider returns a list as a result, the subprovider is individually called up multiple times with each list element. This combination of result cardinality of the higher-level provider and call cardinality of the subordinate provider requires the creation of a special template sheet in the Excel template. You can find additional information for creating these kinds of templates in *Creating report templates* (page 33).

– *CARD_N*

  For data providers that expect a results list from the higher-level data provider to determine the set of results. For example, a provider that returns aggregated information via a list of projects that is returned by the higher-level provider.

The following code example shows a data provider that is to return a list of projects and needs no context for this.

```
class ProjectsProvider(CustomDataProvider):
    CARD      = CARD_N
    CALL_CARD = CARD_0
    ...
```

Unlike the previous example, the data provider presented in the following code example can be used only as a subprovider, because a result object of the higher-level provider is needed for determining results. This kind of provider could, for example, return all ongoing projects for a customer. The customer is the result of the higher-level provider.

```
class ProjectsProvider(CustomDataProvider):
    CARD      = CARD_N
    CALL_CARD = CARD_1
    ...
```

The cardinalities yield how the data providers can be linked to each other. The cardinality of the set of results has to fit with the call cardinality of the subprovider.

- *CARD 1 –> CALL_CARD 1*

  The higher-level provider returns, for example, a project that is used by the subprovider as a parameter for computing its own set of results. This could compute, for example, the project progress and other KPIs for the project.

- *CARD N –> CALL_CARD N*

  For example, to compute aggregated data via the results list of the higher-level provider. The subprovider can return no data that refers to individual list elements of the higher-level result, since this relationship is lost during the export and import and cannot be displayed in Excel. In this case, a multi-export configuration has to be used.

- *CARD N –> CALL_CARD 1* (multi-export special case)

  The subprovider is individually called up and exported for every result object of the higher-level provider. With a multi-export configuration, nested lists and complex rows can be displayed in Excel. The Excel data connection for data that is exported via multi-export has to be established during the template creation on a special template sheet. You can find additional information for creating these kinds of templates in *Displaying complex rows for multi-export data sources* (page 39).

The implementation of a data provider essentially consists of two methods. One method for defining the XML schema and one method for determining the set of results.

```
class ProjectsProvider(CustomDataProvider):
    CARD      = CARD_N
    CALL_CARD = CARD_1
    def getSchema(self):
        ...
    def getData(self, parent_result, source_args, **kwargs):
        ...
```

getSchema(self)

Returns the XML schema for establishing the data connection in Excel. The XML schema has to fit with the data returned by the `getData` method. The schema is described by the construction of an object of the type `XSDType`. The cardinality that has to be identical with the result cardinality of the provider is specified as the constructor parameter. With the `add_attr` method, any number of attributes can be subsequently added to the schema object. The data type is specified as the second parameter in the form of a cdb.sqlapi data type. The added attributes do not necessarily have to be in the database; they can also be computed by the provider, but their attribute names must not include any uppercase letters.

```
class ProjectsProvider(CustomDataProvider):
    CARD      = CARD_N
    CALL_CARD = CARD_1
    def getSchema(self):
        schema = XSDType(self.CARD)
        schema.add_attr('cdb_project_id', sqlapi.SQL_CHAR)
        schema.add_attr('project_name', sqlapi.SQL_CHAR)
        schema.add_attr('calculated', sqlapi.SQL_INTEGER)
        ...
        return schema
```

A relation name or cdb.objects class can optionally be specified in the XSDType constructor as a second parameter. Then the schema object automatically contains all attributes contained in the relation. In the case of a cdb.objects class, all mapped attributes are also contained in the schema object.

```
...
def getSchema(self):
    schema = XSDType(self.CARD, 'cdbpcs_project')
    schema.add_attr('calculated', sqlapi.SQL_INTEGER)
    ...
    return schema
```

If the schema is based on only one database relation or cdb.objects class, getSchema can also directly return a relation name or a cdb.objects class. In this case, the XML schema is completely and automatically generated from the database schema. In the case of a cdb.objects class, again all mapped attributes are also contained in the XML schema.

```
...
def getSchema(self):
    return "cdbpcs_project"
```

For data combined by row, the schema can be constructed from the underlying types with the plus operator. For attributes of the same name, a prefix has to be specified as the third parameter in the participating relations that ensures that the attribute names in the XML schema are unique. This prefix is also to be used in the implementation of `getData` when the result objects are being constructed.

The following example shows the construction of a schema for parts, expanded by adding the schema for the associated projects:

```
...
def getSchema(self):
    schema = XSDType(self.CARD,'teile_stamm')
    schema += XSDType(self.CARD, 'cdbpcs_project', 'Project_')
    ...
    return schema
```

getData(self, parent_result, source_args, **kwargs)

Returns the report data when calling up a report. The following arguments are transferred to the provider for determining the set of results:

parent_result Set of results of the higher-level provider. If the higher-level provider is the context provider, `parent_result` contains the objects on which the report was called up. Depending on the result cardinality of the higher-level provider, `parent_result` is either of the type `ReportDataList` or `ReportData`. For top-level providers, this parameter is always `None`.

source_args A dictionary with all arguments from the optionally preceding report dialog. The keys of the dictionaries correspond to the attribute names of the underlying mask configuration, but currently they always consist only of lowercase letters, even if they were defined in the mask with some uppercase letters (in the provider name). For assigning the parameters to the data providers, the attribute names in the mask configuration follow the name schema <Providername>-<Attributname>.

kwargs All arguments from the optionally preceding report dialog that are addressed to the data provider with the naming convention <Providername>- <Attributname>. The address to the provider is to be removed from the argument name, so that the arguments can be used directly via attribute name. As mentioned above, the attribute names at this place always contain only lowercase letters.

The report data is returned depending on the result cardinality (CARD) of the provider. A `ReportData` object has to be returned for the result cardinality CARD_1; a `ReportDataList` has to be returned for the result cardinality CARD_N. The construction of this result object is described in the following.

ReportData

Result object for data providers with the result cardinality CARD_1.

`ReportData` objects are always constructed by specifying the constructing provider as the first parameter. Thus `self` is always transferred as the first parameter. The following example constructs an empty `ReportData` object and returns this as the result.

```
class SampleProvider(CustomDataProvider):
    CARD = CARD_1
    ...
    def getData(self,parent_result,source_args,**kwargs):
        return ReportData(self)
```

The access to the elements of a `ReportData` object corresponds to the usual methods for accessing Python dictionaries. The `keys,values` and `items` methods are available. Individual attributes are set and read by the index operator.

```
def getData(self,parent_result,source_args,**kwargs):
    data = ReportData(self)
    data["cdb_project_id"] = "4711"
    data["project_name"] = "Test project"
    return data
```

`ReportData` objects can be directly constructed from a cdb.objects object or sqlapi.Record. For this purpose, the object or record is transferred to the constructor as the second parameter.

```
def getData(self,parent_result,source_args,**kwargs):
    project = pcs.Project.ByKeys('P000001')
    data = ReportData(self, project)
    return data
```

In addition, any number of fields can be added.

```
def getData(self,parent_result,source_args,**kwargs):
    project = pcs.Project.ByKeys('P000001')
    data = ReportData(self, project)
    data["calculated_attribute"] = 42
    return data
```

ReportData objects that are constructed from cdb.objects objects can also return the long texts belonging to the object as a result. For this purpose, a list of long text relations can be specified as a third parameter.

```
def getData(self,parent_result,source_args,**kwargs):
    project = pcs.Project.ByKeys('P000001')
    data = ReportData(self, project, ['cdbpcs_proj_txt'])
    return data
```

For data combined by row, for example, parts with an associated project, ReportData objects can be linked with the plus operator. For attributes of the same name, a prefix has to be specified as the third parameter in the participating relations that ensures that the attribute names are unique. The same prefix must be used as the one used for creating the schema in the implementation of getSchema. The following example shows the creation of a ReportData object from a part, followed by expansion by adding the associated project.

```
def getData(self,parent_result,source_args,**kwargs):
    item = pdd.Item.ByKeys('4711', '')
    data = ReportData(self, item)
    data += ReportData(self, item.Project, prefix="Project_")
    return data
```

ReportDataList

Result object for data providers with the result cardinality CARD_N.

ReportDataList objects are always constructed by specifying the constructing provider as the first parameter. The following example constructs an empty ReportDataList and returns this as the result.

```
class SampleProvider(CustomDataProvider):
    CARD = CARD_N
    ...
    def getData(self):
        return ReportDataList(self)
```

ReportDataList objects consist of a list of ReportData objects and provide the usual methods for accessing Python lists to access list elements: append, count, extend, index, insert, pop, remove, reverse and sort. Moreover, all list operators can be used. The following example shows how the plus operator can be used to add two list elements, each of which contains a project.

```
def getData(self,parent_result,source_args,**kwargs):
    data = ReportDataList(self)
    data += ReportData(self, pcs.Project.ByKeys('P000001'))
    data += ReportData(self, pcs.Project.ByKeys('P000002'))
    return data
```

A sqlapi.Record or cdb.objects object can also be added directly as an abbreviated form using the plus operator, without first having to construct a ReportData object yourself.

```
def getData(self,parent_result,source_args,**kwargs):
    data = ReportDataList(self)
    data += pcs.Project.ByKeys('P000001')
    data += pcs.Project.ByKeys('P000002')
    return data
```

ReportDataList objects can be constructed directly from a list of cdb.objects objects or a sqlapi.RecordSet2. In doing so, ReportData objects are automatically constructed and added for every list element.

```
def getData(self,parent_result,source_args,**kwargs):
    projects = pcs.Project.Query("status>0")
    data = ReportDataList(self, projects)
    return data
```

A list of long text relations to be included can be specified as a third parameter during the construction from cdb.objects objects.

```
def getData(self,parent_result,source_args,**kwargs):
    projects = pcs.Project.Query("status>0")
    data = ReportDataList(self, projects, ['cdbpcs_proj_txt'])
    return data
```

During the construction of results lists, take into account that the list elements must not be heterogeneous. It is also not possible, for example, to return projects and documents mixed in a ReportDataList.

getClass(self)

Gives the cdb.objects class back to the returned data if the provider data is returned in the form of cdb.objects objects. If the data is not constructed from cdb.objects objects, this method is not implemented. The following example shows the implementation for a provider that returns projects as a result.

```
def getClass():
    return pcs.Projects.Project
```

If a separate provider returns combined data, such as parts with a project, specify the type of the primary object. The primary object is the initial object that was used for the ReportData construction.

```
def getData(self,parent_result,source_args,**kwargs):
    item = pdd.Item.ByKeys('4711', '')
    data = ReportData(self, item) # <- Primary object
    data += ReportData(self, item.Project, prefix="Project_")
    return data
def getClass(self):
    return pdd.Item # <- Primary object
```

The implementation of `getClass` enables the use of relationship providers as subproviders to separate providers. Thus separate providers can be directly integrated into the object model of the cdb.objects framework. For example, if a separate provider returns projects, all relationships proceeding from projects can be appended to the separate provider as subproviders. Therefore open issues, tasks or documents, for example, can be integrated into the data source in this way for the self-determined projects without additional programming.

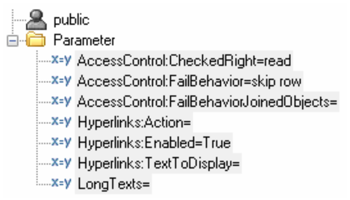
Moreover, by this method the report framework recognizes cdb.objects-based providers, so that if implementation is missing, for example, the parameters described in *Provider parameters* (page 18) are not automatically created.

getDefaultParameters(self)

Returns a dictionary with parameter names and default values. These are automatically created when a provider of this type is created, and they can be subsequently adjusted in the ‘Parameters’ tab of the provider. The parameter names returned through this method can be used in the provider implementation in order to make the provider configurable in the application context. In this manner, you can implement universal and configurable data providers yourself.

```
def getDefaultParameters():
    return {"My Parameter" : "My Value"}
```

cdb.objects-based data providers also always receive the following default parameters automatically, insofar as the `getClass` method was implemented.



```
public
Parameter
  x=y AccessControl:CheckedRight=read
  x=y AccessControl:FailBehavior=skip row
  x=y AccessControl:FailBehaviorJoinedObjects=
  x=y Hyperlinks:Action=
  x=y Hyperlinks:Enabled=True
  x=y Hyperlinks:TextToDisplay=
  x=y LongTexts=
```

Fig. 2.28: Excel: PowerReports - default parameters

Assessment of these parameters takes place automatically; you do not need to implement it yourself for separate providers.

Access to separate parameters of a provider takes place via the `getParameter(name)` method.

```python
def getData(self,parent_result,source_args,**kwargs):
    param = self.getParameter("My Parameter")
    if param == "My Value":
        ...
    else:
        ...
        ...
```

getArgumentDefinitions(self)

The set of results for data providers can be affected by a preceding report dialog (see also *Creating report dialogs* (page 34)). Usually, the user then runs interactive queries with additional restrictive search conditions. So that later, it is possible to trace which data are represented in a report, and thus which query parameters led to the specific result, it is frequently advisable that the query parameters used be included in the report.

The *getArgumentDefinitions* method can be implemented for this purpose. This method returns a dictionary with parameter names and the associated data types. The returned argument names are automatically added to the XML schema with the specified data type under the schema name *Arguments* and are exported along with the data export.

The argument names are automatically expanded to include the provider names so that a unique assignment to the data provider is ensured (similar to the dialog configuration as described in *Creating report dialogs* (page 34)).

Important: The attribute names may not contain any uppercase letters.

Usage in detail, exemplified by a report on projects with a preceding dialog for optional constraint to a specific category (the dialog contains the attribute *projects category*, making it possible to enter a category evaluated by the *projects* data provider):

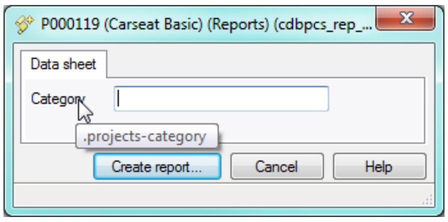


Fig. 2.29: Excel: PowerReports - report dialog using a project category query as an example

The data provider implementation displayed in the following code example evaluates the queried category, thereby restricting the set of results. The *getArgumentDefinitions* method returns the *category* attribute and communicates to the PowerReports Framework that this argument should also be exported.

```
class Projekte(CustomDataProvider):
    def getSchema(self):
        return "cdbpcs_project"
    def getData(self,parent_result,source_args,**kwargs):
        category = kwargs["category"] # Parameter from dialog
        cond = ""
        if category:
            cond = "category='%s'" % category
        return ReportDataList(self, pcs.Projects.Project.Query(cond))
    def getArgumentDefinitions(self):
        return {"category" : sqlapi.SQL_CHAR}
```

Due to the above implementation of *getArgumentDefinitions*, the attribute used in the report dialog is available under *Arguments* and can be used in the data connection for the Excel template.

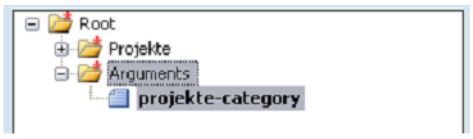
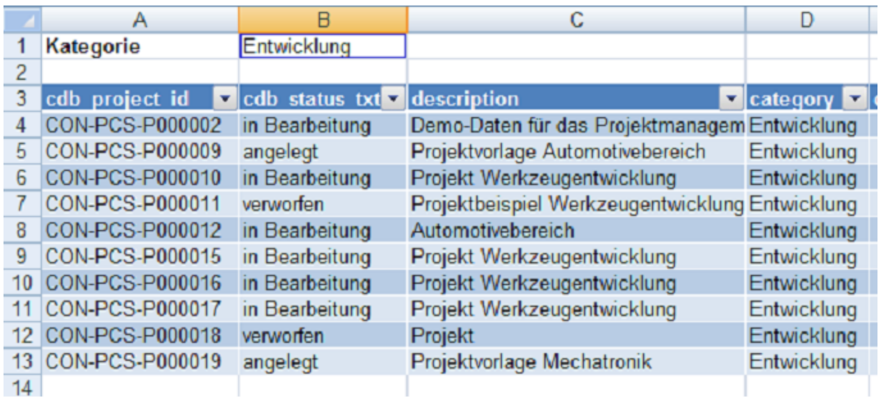


Fig. 2.30: Excel: PowerReports - XML schema for arguments

After the data connection is created, reports contain the search condition used for the project category.

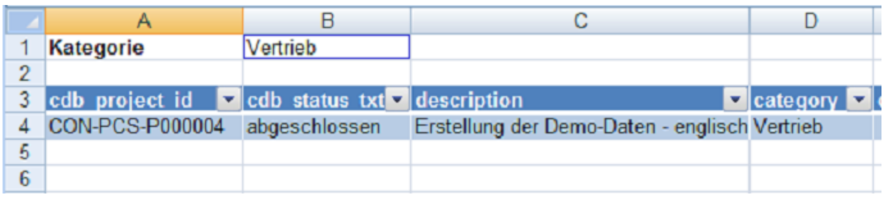


	A	B	C	D
1	Kategorie	Entwicklung		
2				
3	cdb_project_id	cdb_status_txt	description	category
4	CON-PCS-P000002	in Bearbeitung	Demo-Daten für das Projektmanagem	Entwicklung
5	CON-PCS-P000009	angelegt	Projektvorlage Automotivebereich	Entwicklung
6	CON-PCS-P000010	in Bearbeitung	Projekt Werkzeugentwicklung	Entwicklung
7	CON-PCS-P000011	verworfen	Projektbeispiel Werkzeugentwicklung	Entwicklung
8	CON-PCS-P000012	in Bearbeitung	Automotivebereich	Entwicklung
9	CON-PCS-P000015	in Bearbeitung	Projekt Werkzeugentwicklung	Entwicklung
10	CON-PCS-P000016	in Bearbeitung	Projekt Werkzeugentwicklung	Entwicklung
11	CON-PCS-P000017	in Bearbeitung	Projekt Werkzeugentwicklung	Entwicklung
12	CON-PCS-P000018	verworfen	Projekt	Entwicklung
13	CON-PCS-P000019	angelegt	Projektvorlage Mechatronik	Entwicklung
14				

Fig. 2.31: Excel: PowerReports - report with a restricted search condition (1)

The search condition can be modified directly within the report and the *report update* link function can be subsequently called up.

The report is updated with the modified search conditions, without these needing to be called up again from the system.



	A	B	C	D
1	Kategorie	Vertrieb		
2				
3	cdb_project_id	cdb_status_txt	description	category
4	CON-PCS-P000004	abgeschlossen	Erstellung der Demo-Daten - englisch	Vertrieb
5				
6				

Fig. 2.32: Excel: PowerReports - report with a restricted search condition (2)

Adding additional hyperlinks

Hyperlinks are created automatically for the objects returned by a provider, dependent upon the provider configuration. If these objects are to contain further links to referenced objects, these must be explicitly added. Note that the XSD schema must be expanded accordingly. Creation of hyperlinks is carried out with the `MakeReportURL` method. The following example shows how a hyperlink is added for the assigned project of a part.

```
def getSchema(self):
    t = XSDType(1,'teile_stamm')
    t.add_attr("project_hyperlink", sqlapi.SQL_CHAR)
    return t
def getData(self,parent_result,source_args,**kwargs):
    item = pdd.Item.ByKeys('4711', '')
    data = ReportData(self, item)
    data["project_hyperlink"]=MakeReportURL(item.Project)
    return data
```

In the example above, the hyperlink is created with the CDB_ShowObject default action and the configured object description from the class directory.

As the second and third parameters, the `MakeReportURL` method can be passed to any other action and to a label to be displayed. If an attribute name of the object is specified as a label, the value of this attribute is used. Otherwise, the specified character string is invariably used as the label.

```
def getData(self,parent_result,source_args,**kwargs):
    item = pdd.Item.ByKeys('4711', '')
    data = ReportData(self, item)
    data["project_hyperlink"] = MakeReportURL(item.Project, "CDB_Modify", "project_
↪name")
    return data
```

In the case of combined data, the hyperlinks are automatically created for the primary object only. The following example shows report data for a part, expanded by the data of the associated project. No hyperlink is automatically created for the project object in this case. If a hyperlink is to be included in the project, it must be added manually as described previously.

```
def getData(self,parent_result,source_args,**kwargs):
    item = pdd.Item.ByKeys('4711', '')
    data = ReportData(self, item)  # <- Primary object
    data += ReportData(self, item.Project, prefix="Project_")
    data["project_hyperlink"] = MakeReportURL(item.Project)
    return data
```

Adding image files

If a data provider is to return dynamic image contents (see also *Displaying dynamic image contents* (page 38)), then the XSD schema must first be expanded accordingly, as in the previous chapter.

```
def getSchema(self):
    schema = XSDType(CARD_N, ''teile_stamm')
    schema.add_attr('cdbxml_image_preview', sqlapi.SQL_CHAR)
    return schema
```

Image links are added via the *add_image* method.

add_image(self, image=None, attr_name='cdbxml_image', hyperlink='openfile', args)

image Possible values:

- CDB_File object:

  - for example CDB_File.ByKeys(cdb_object_id='...')
  - The file is transferred together with the other report data.
- Object ID of a CDB_File object:
  - For example, some_preview_file_obj.cdb_object_id
  - The file is transferred together with the other report data.
- Simple path:
  - For example, "C:\temp\image.jpg"
  - The file is transferred together with the other report data.
- Absolute URI path:
  - For example, "file:///C:\temp\image.jpg"
  - The file is *not* transferred to the other report data, but rather later imported directly from the link during report generation. Of course, you have to ensure that the respective machine compiling the report has access to this path (e.g. network path).
- HTTP path:
  - For example, "http://local.myserver.com/image.jpg"
  - The file is *not* transferred to the other report data, but rather later imported directly from the link during report generation. Of course, here too, ensure that the respective machine compiling the report has access to this Internet path.

attr_name The attribute previously used to expand the XSD schema at another location (see *add_attr*).

hyperlink By default, every image in the PowerReport is connected with a hyperlink that points directly to the local image file. By clicking on it, one usually sees the image in its (enlarged) original state in Internet Explorer.

The following example shows how to add an image file.

```
def getData(self, parent_result, source_args, **kwargs):
    item = pdd.Item.ByKeys('4711', '')
    data = ReportData(self, item)
    filename = _getPreviewImageFile(item)   # some custom method for retrieving␣
↪regarding preview images
    data.add_image(filename, 'cdbxml_image_preview')
    return data
```

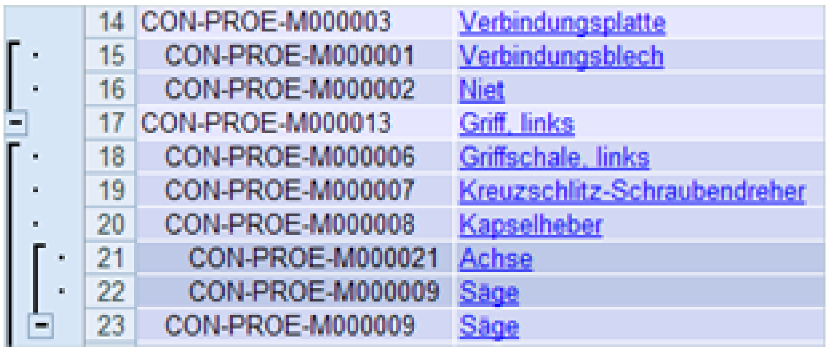
An additional example shows how to install your own additional hyperlinks.

```
def getData(self, parent_result, source_args, **kwargs):
    item = pdd.Item.ByKeys('4711', '')
    data = ReportData(self, item)
    filename = _getPreviewImageFile(item)   # some custom method for retrieving␣
↪regarding preview images
    hyperlink = item.MakeURL(None, plain=2)
    data.add_image(filename, 'cdbxml_image_preview', hyperlink)
    return data
```

Important: When updating a PowerReport, image files are *not* also updated.

Adding hierarchy information for hierarchical data

Hierarchical data, such as product structures or task structures, can be correspondingly represented in PowerReports. Hierarchy levels can be expanded or collapsed in PowerReports and can be color-coded and indented for better overview.



14	CON-PROE-M000003	Verbindungsplatte
15	CON-PROE-M000001	Verbindungsblech
16	CON-PROE-M000002	Niet
17	CON-PROE-M000013	Griff, links
18	CON-PROE-M000006	Griffschale, links
19	CON-PROE-M000007	Kreuzschlitz-Schraubendreher
20	CON-PROE-M000008	Kapselheber
21	CON-PROE-M000021	Achse
22	CON-PROE-M000009	Säge
23	CON-PROE-M000009	Säge

Fig. 2.33: Excel: PowerReports - hierarchical representation using the product structure as an example

To represent hierarchical data, the Excel link requires information regarding the hierarchy depth. This information is imparted to the link through a further attribute with the fixed name `cdbxml_level`.

Using the product structure as an example, the following code shows how `cdbxml_level` is added to the schema and how it can be added during data export. Note that the `getComponents2` method used in the example to determine the product structure is not part of a system standard distribution. Therefore, the code can not be copied for a separate provider without additional adjustments.

```
def getSchema(self):
    t = XSDType(N, "einzelteile")
    t.add_attr("cdbxml_level", sqlapi.SQL_INTEGER)
    return t
def getData(self, parent_result, source_args, **kwargs):
    item = parent_result.getObject()
    result = ReportDataList(self)
    # Custom method getComponents2 provides product structure tuples containing␣
↪item and depth
    for lev, obj in item.getComponents2():
        d = ReportData(self, obj)
        d["cdbxml_level"] = lev
        result.append(d)
    return result
```

You can find the description for creating Excel templates with representation of hierarchical data in *Displaying hierarchical data* (page 35).

2.4.5 Activating PowerReports

Context-related reports have to be activated for each object class used for PowerReports. Besides this, a minimum configuration has to be carried out in the respective pop-up menu for the report selection for each object class to be activated.

Activation occurs in two steps:

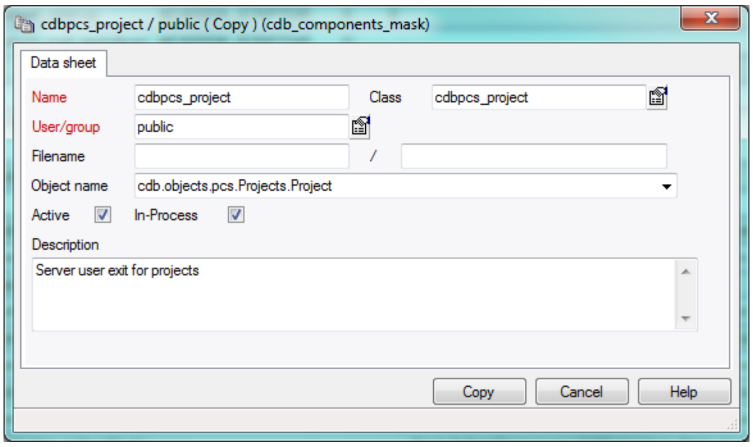
1. Expanding the corresponding cdb.objects class by derivation
2. Activating the ‘Reports’ operation for the pop-up menu

cdb.objects class

Using projects as an example:

```
from cdb.objects.pcs.Projects import Project
from cdb.objects import PowerReports
class MyProject(Project, :guilabel:`PowerReports.WithPowerReports` ):
...
```

If no cdb.objects-based component exists yet for the object class to be activated, you have to create it additionally in the component configuration for the already existing server_<relation> component, if applicable. Using projects as an example:


cdbpcs_project / public (Copy) (cdb_components_mask)

Data sheet

Name	cdbpcs_project	Class	cdbpcs_project
User/group	public		
Filename		/	
Object name	cdb.objects.pcs.Projects.Project		
Active	✓	In-Process	✓

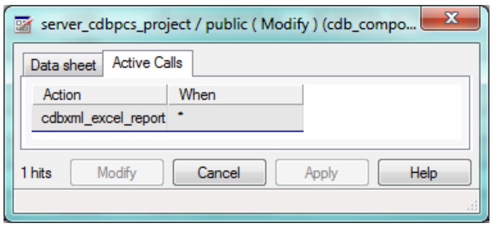
Description

Server user exit for projects

Fig. 2.34: Excel: PowerReports - cdb.objects-based component for projects

You have to create a component for each switch class name if the reports are to be set up for multiple switch classes with different class names but the same type, such as for different project classes. All components have to use the same cdb.objects class in the object name field for this.

In the 'Active Calls' tab, you have to activate the function `cdbxml_excel_report` with the execution time *. *You must never activate all functions with *!* This would cause an unwanted activation of additional user exit codes in existing installations, which would be run in addition to the existing server_<relation> user exit.


server_cdbpcs_project / public (Modify) (cdb_compo...

Data sheet | Active Calls

Action	When
cdbxml_excel_report	*

1 hits

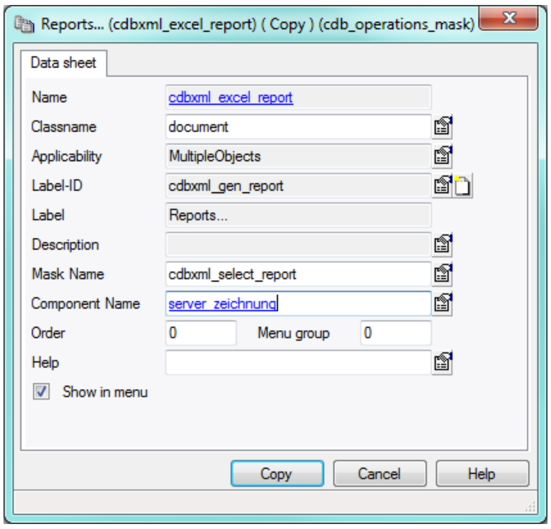
Fig. 2.35: Excel: PowerReports - Activating the cdbxml_excel_report operation

'Reports' operation

The operation is activated in the 'Configuration' tab of the `cdbxml_excel_report` operation by adding an additional entry for the class to be activated. At least one entry has to exist for a project by default, which can be used as a master copy.

This is used to complete the activation of reports using documents as an example. Any number of reports in the context of documents can be called up in the pop-up menu both as single select as well as multi-select.

After selecting the 'Reports...' pop-up menu function, the dialog opens for selecting the report to be run. The report list of options automatically lists all reports available to the user. If you use multi-select, single-select reports are automatically hidden (see *Running PowerReports* (page 44)).


Reports... (cdbxml_excel_report) (Copy) (cdb_operations_mask)

Data sheet

Name	cdbxml_excel_report
Classname	document
Applicability	MultipleObjects
Label-ID	cdbxml_gen_report
Label	Reports...
Description	
Mask Name	cdbxml_select_report
Component Name	server_zeichnung
Order	0
Menu group	0
Help	

☑ Show in menu

Fig. 2.36: Excel: PowerReports - operation configuration using documents as an example

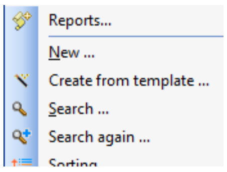


Fig. 2.37: Excel: PowerReports - pop-up menu operation for calling up reports

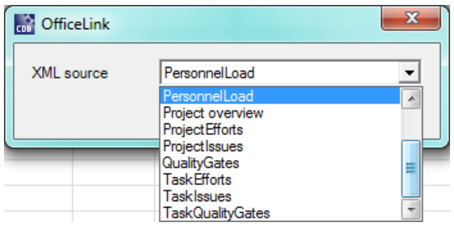
2.4.6 Creating report templates

New reports are created supported by the Excel link in three steps:

1. *Selecting data sources*

   The data source is selected using the link function *[Select data source]*.

   A dialog is displayed for selecting a data source configured in the system.


OfficeLink

XML source: PersonnelLoad

Fig. 2.38: Excel: PowerReports - selection dialog for data sources

2. *Establish data connection and design layout*

   After selecting the data source, the associated XSD schema is loaded and displayed. The XSD schema is used to establish the data connection. The data connection is established by dragging and dropping schema elements to the desired position in the Excel sheet. Here, you can position individual attributes as well as complete objects at once.

3. *Save document as a report template*

   Finally, the newly created report has to be saved as a report template for the data source. You can do so using the link function *[Create template]*.

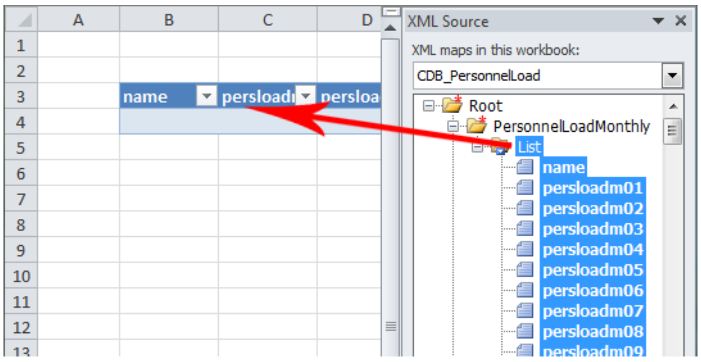


Fig. 2.39: Excel: PowerReports - Establishing data connection using drag and drop

Report templates are attached in the system to the respective report directly as files. Here, you can create multiple language-specific templates for one and the same report (see *Creating language-specific reports* (page 42)).

Note: Since report templates are not created as separate documents but are rather directly attached to the reports, report templates cannot be opened later via the *[Load]* function of the link for re-editing. You can open for re-editing either in the client (e.g. in the project overview) or after running a report via the *[Edit template]* function.

After successfully creating the file, the report template is assigned to the data source automatically. The report can then be called up immediately. Depending on the data source, this can be done context-free or in the pop-up menu of a corresponding object for the report creator.

All reports created for a data source are listed in the reports folder of a data source. There, you can adjust the authorizations, the title of the report and, if applicable, a preceding dialog. As an example, figure 32 shows the project portfolio report for the data source of the same name. Any number of reports with varying characteristics and in any of the different language versions can be created for a data source.

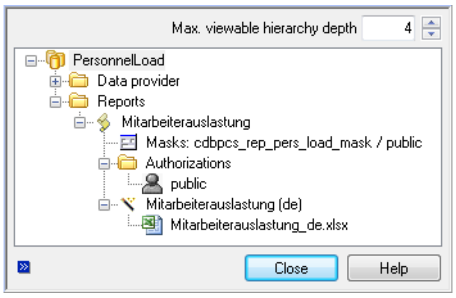

Max. viewable hierarchy depth
4
PersonnelLoad
Data provider
Reports
Mitarbeiterauslastung
Masks: cdbpcs_rep_pers_load_mask / public
Authorizations
public
Mitarbeiterauslastung (de)
Mitarbeiterauslastung_de.xlsx
Close
Help

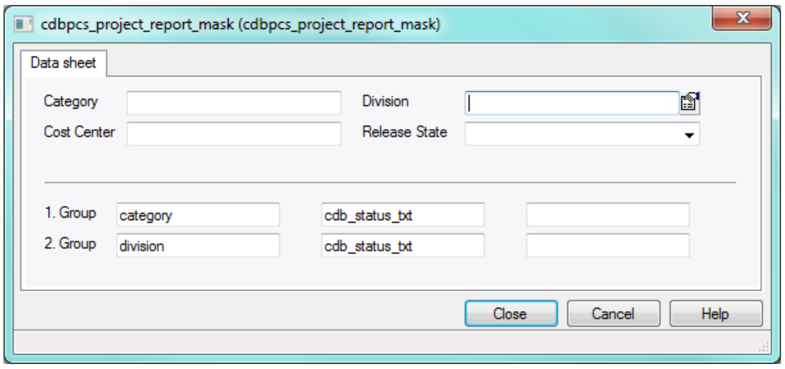
Fig. 2.40: Excel: PowerReports - Report for data source

The report title and a preceding dialog, if applicable, can be configured in the data sheet for assigning a template to the data source. The `Authorizations` tab controls the visibility of reports in the report selection dialog. The report creator is authorized automatically by a corresponding assignment upon creation. Other users can call up the report only after they have also become authorized by a role assignment or by direct personal assignment.

Furthermore, various additional settings for the type of the report generation and its layout can be configured in the data sheet (see *Selection configuration* (page 45)).

Creating report dialogs

A report can be optionally preceded by a dialog that asks the user to set parameters that can be forwarded to the data providers when calling up a report and influence the data retrieval there. This always useful if users are to be able to influence the report result by making their own entries. For example, by entering a time period to be considered or other filtering condition in combination with SimpleQuery data providers.



cdbpcs_project_report_mask (cdbpcs_project_report_mask)

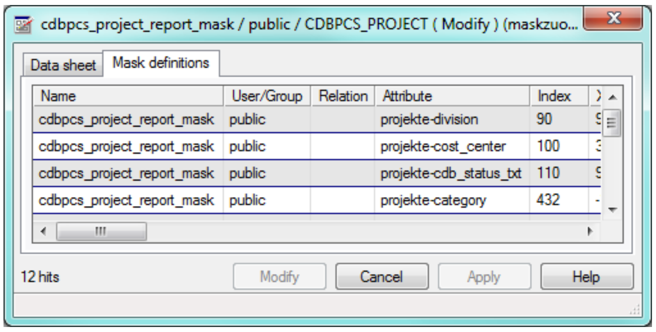
Data sheet

Category		Division	
Cost Center		Release State	

1. Group	category	cdb_status_txt	
2. Group	division	cdb_status_txt	

Fig. 2.41: Excel: PowerReports - report dialog for querying projects

The parameters are addressed to data providers by naming conventions for the attribute names of the mask fields: <Providername>-<Attributname>, where the attribute names must not contain any capital letters. The relation field of the mask field configuration remains empty.



cdbpcs_project_report_mask / public / CDBPCS_PROJECT (Modify) (maskzuo...

Name	User/Group	Relation	Attribute	Index
cdbpcs_project_report_mask	public		projekte-division	90
cdbpcs_project_report_mask	public		projekte-cost_center	100
cdbpcs_project_report_mask	public		projekte-cdb_status_txt	110
cdbpcs_project_report_mask	public		projekte-category	432

12 hits

Fig. 2.42: Excel: PowerReports - addressing the dialog fields to data provider as parameters

The options for setting parameters of the universal standard providers using dialogs can be found in *Types of providers in detail* (page 14) for the respective provider type.

Note: If fields are changed, deleted or added in the report dialogs, note that the XML schema has to be updated in the report templates that use the respective dialog.

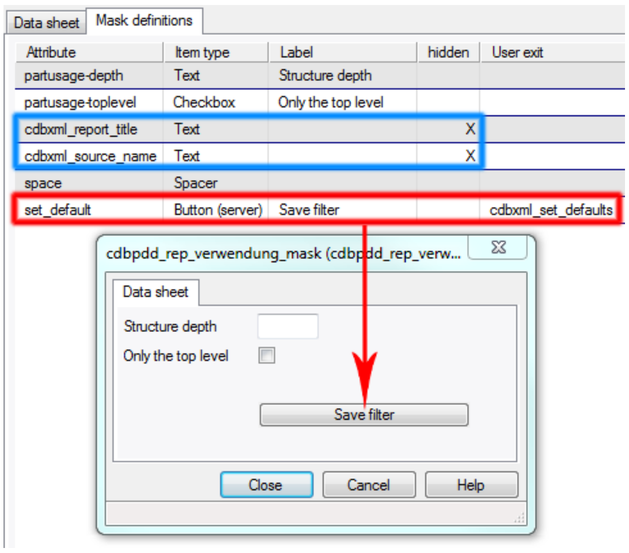
Moreover, each dialog can be supplemented by the option to save all presettings. This can be used to potentially prevent repeated entries when re-running reports. To achieve this, you must add three elements to the dialog in question: the 2 (hidden, but which identify the respective report) text fields *cdbxml_source_name* and *cdbxml_report_title*; 1 button which triggers the component *cdbxml_set_defaults* when pushed (see *Excel: PowerReports - Saving the entry values of report dialogs* (page 36)).

Displaying hierarchical data

In PowerReports, data organized in a hierarchical manner can be displayed accordingly. Typical examples of hierarchical data are product structures or task structures.

A prerequisite for the hierarchical representation is that the data provider provides the hierarchy information in the attribute `cdbxml_level`. If this prerequisite is met, the hierarchy levels can be expanded or collapsed in the PowerReport and color-coded and indented for a better overview.

To create a list organized in a hierarchical manner, the attribute `cdbxml_level` must be contained in the list. Ideally, this attribute is appended to the end of the header and set as invisible after completing the layout work



Data sheet | Mask definitions

Attribute	Item type	Label	hidden	User exit
partusage-depth	Text	Structure depth		
partusage-toplevel	Checkbox	Only the top level		
cdbxml_report_title	Text		X	
cdbxml_source_name	Text		X	
space	Spacer			
set_default	Button (server)	Save filter		cdbxml_set_defaults

cdbpdd_rep_verwendung_mask (cdbpdd_rep_verw...

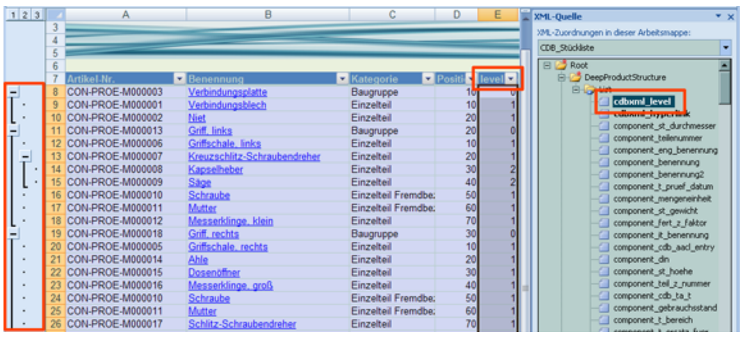
Data sheet

Structure depth

Only the top level ☐

Fig. 2.43: Excel: PowerReports - Saving the entry values of report dialogs

by setting the column width to 0. If the attribute `cdbxml_level` is contained in the list, the results table is structured automatically using Excel functions, as illustrated in the figure below in the left edge.



	Artikel-Nr.	Benennung	Kategorie	Positi	level
8	CON-PROE-M000003	Verbindungsplatte	Baugruppe	10	0
9	CON-PROE-M000001	Verbindungsblech	Einzelteil	10	1
10	CON-PROE-M000002	Niet	Einzelteil	20	1
11	CON-PROE-M000013	Griff, links	Baugruppe	20	0
12	CON-PROE-M000006	Griffschale, links	Einzelteil	10	1
13	CON-PROE-M000007	Kreuzschlitz-Schraubendreher	Einzelteil	20	1
14	CON-PROE-M000008	Kapselheber	Einzelteil	30	2
15	CON-PROE-M000009	Säge	Einzelteil	40	2
16	CON-PROE-M000010	Schraube	Einzelteil Fremdbe:	50	1
17	CON-PROE-M000011	Mutter	Einzelteil Fremdbe:	60	1
18	CON-PROE-M000012	Messerklinge, klein	Einzelteil	70	1
19	CON-PROE-M000018	Griff, rechts	Baugruppe	30	0
20	CON-PROE-M000005	Griffschale, rechts	Einzelteil	10	1
21	CON-PROE-M000014	Ahle	Einzelteil	20	1
22	CON-PROE-M000015	Dosenöffner	Einzelteil	30	1
23	CON-PROE-M000016	Messerklinge, groß	Einzelteil	40	1
24	CON-PROE-M000010	Schraube	Einzelteil Fremdbe:	50	1
25	CON-PROE-M000011	Mutter	Einzelteil Fremdbe:	60	1
26	CON-PROE-M000017	Schlitz-Schraubendreher	Einzelteil	70	1

XML-Quelle

XML-Zuordnungen in dieser Arbeitsmappe:

CDB_Stückliste

```
Root
  DeepProductStructure
    List
      cdbxml_level
      cdbxml_hyperlink
      component_st_durchmesser
      component_teilenummer
      component_eng_benennung
      component_benennung
      component_benennung2
      component_t_pruef_datum
      component_mengeneinheit
      component_st_gewicht
      component_fert_z_faktor
      component_it_benennung
      component_cdb_aacl_entry
      component_din
      component_st_hoehe
      component_teil_z_nummer
      component_cdb_ta_t
      component_gebrauchsstand
      component_t_bereich
```

Fig. 2.44: Excel: PowerReports - hierarchical representation using the product structure as an example

This organization can be additionally highlighted by indenting any column and by color-coding. For this purpose, you have to add a specific expression in each of the column labels of the `cdbxml_level` attribute. The expressions have to be separated from each other by a colon. Indentation of a column:

```
indent(column)
```

Color coding:

```
colorize(<rgb_start_color>),(<rgb_offset>)
```

`column` specifies the column index in which the indentations are to be carried out.

`rgb_start_color` specifies the start color as a RGB color code for the hierarchy level 0. For the levels below that, the row color is determined by adding the `rgb_offsets`.

Excel: PowerReports - hierarchical representation with indentation of the first column and color coding

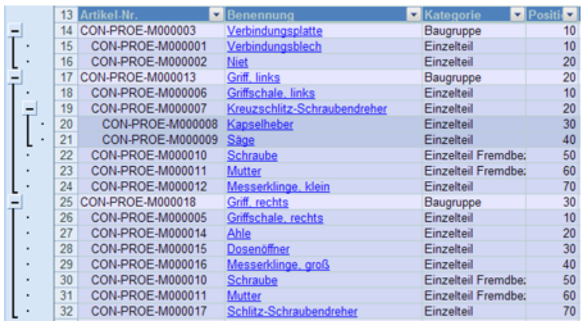
Only indentation of the 1st column:

```
level:indent(1)
```

Indentation of the 1st column + blue color coding:

```
level:colorize(230,230,255),(20,15,12):indent(1)
```

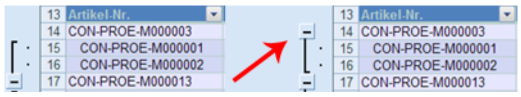
The expression level:colorize(230,230,255),(20,15,12):indent(1), for example, results in the following representation.



13	Artikel-Nr.	Benennung	Kategorie	Positi
14	CON-PROE-M000003	Verbindungsplatte	Baugruppe	10
15	CON-PROE-M000001	Verbindungsblech	Einzelteil	10
16	CON-PROE-M000002	Niet	Einzelteil	20
17	CON-PROE-M000013	Griff, links	Baugruppe	20
18	CON-PROE-M000006	Griffschale, links	Einzelteil	10
19	CON-PROE-M000007	Kreuzschlitz-Schraubendreher	Einzelteil	20
20	CON-PROE-M000008	Kapselheber	Einzelteil	30
21	CON-PROE-M000009	Säge	Einzelteil	40
22	CON-PROE-M000010	Schraube	Einzelteil Fremdbe:	50
23	CON-PROE-M000011	Mutter	Einzelteil Fremdbe:	60
24	CON-PROE-M000012	Messerklinge, klein	Einzelteil	70
25	CON-PROE-M000018	Griff, rechts	Baugruppe	30
26	CON-PROE-M000005	Griffschale, rechts	Einzelteil	10
27	CON-PROE-M000014	Ahle	Einzelteil	20
28	CON-PROE-M000015	Dosenöffner	Einzelteil	30
29	CON-PROE-M000016	Messerklinge, groß	Einzelteil	40
30	CON-PROE-M000010	Schraube	Einzelteil Fremdbe:	50
31	CON-PROE-M000011	Mutter	Einzelteil Fremdbe:	60
32	CON-PROE-M000017	Schlitz-Schraubendreher	Einzelteil	70

Fig. 2.45: Excel: PowerReports - hierarchical representation with indentation of the first column and color coding

Note: Groupings with the setting 'Main rows below detail data' are displayed in Excel by default. However, the user usually prefers them above the detail data, as shown in the example above. Therefore, make sure that this setting is disabled when creating the PowerReport template (*Data* → *Organization* → *Settings*).



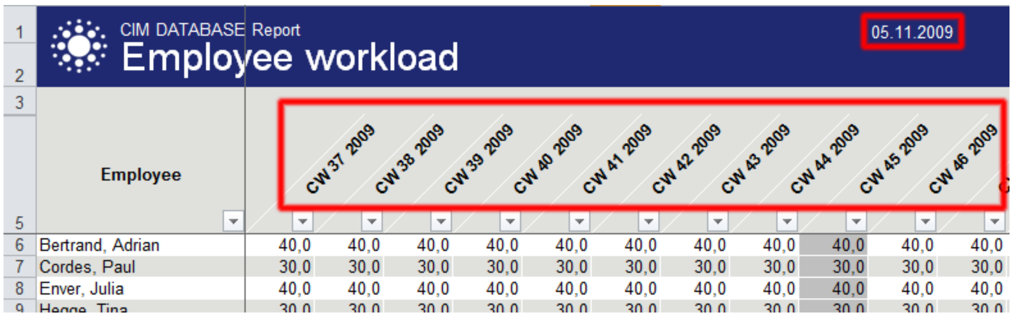
13	Artikel-Nr.
14	CON-PROE-M000003
15	CON-PROE-M000001
16	CON-PROE-M000002
17	CON-PROE-M000013

13	Artikel-Nr.
14	CON-PROE-M000003
15	CON-PROE-M000001
16	CON-PROE-M000002
17	CON-PROE-M000013

Fig. 2.46: Excel: PowerReports - hierarchical representation - grouping settings

Displaying dynamic column titles in lists

In PowerReports, you have the option to dynamically assign values to the column titles of lists that are filled by providers. The PowerReport is used as an example:

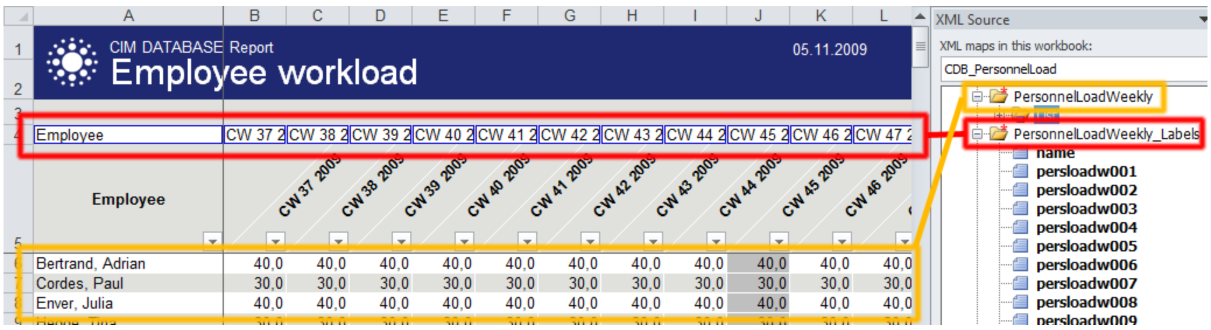


CIM DATABASE Report
Employee workload
05.11.2009

	Employee	CW 37 2009	CW 38 2009	CW 39 2009	CW 40 2009	CW 41 2009	CW 42 2009	CW 43 2009	CW 44 2009	CW 45 2009	CW 46 2009
6	Bertrand, Adrian	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0
7	Cordes, Paul	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0
8	Enver, Julia	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0
9	Hegge, Tina	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0

Fig. 2.47: Excel: PowerReports - displaying dynamic column titles

This PowerReport for displaying the employee workload shows a time period specified by the user before generating in a list. Afterwards, the time period can be changed in the field highlighted in red and the PowerReport can be updated. This makes the column titles dependent on the time period since they display the respective calendar weeks. This is implemented by adding an additional provider ('PersonnelLoadWeekly_Label'), which provides the current column titles each time a PowerReport is created/updated. You have to place the values in the row directly above the list so that the Excel link can find, export and commit them to the list. Upon completion, this row must be hidden in the template.



CIM DATABASE Report
Employee workload
05.11.2009

A	B	C	D	E	F	G	H	I	J	K	L
Employee	CW 37 2	CW 38 2	CW 39 2	CW 40 2	CW 41 2	CW 42 2	CW 43 2	CW 44 2	CW 45 2	CW 46 2	CW 47 2
Employee	CW 37 2009	CW 38 2009	CW 39 2009	CW 40 2009	CW 41 2009	CW 42 2009	CW 43 2009	CW 44 2009	CW 45 2009	CW 46 2009	
Bertrand, Adrian	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0
Cordes, Paul	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0	30,0
Enver, Julia	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0	40,0

XML Source
XML maps in this workbook:
CDB_PersonnelLoad

- PersonnelLoadWeekly
- PersonnelLoadWeekly_Labels
  - name
  - persloadw001
  - persloadw002
  - persloadw003
  - persloadw004
  - persloadw005
  - persloadw006
  - persloadw007
  - persloadw008
  - persloadw009

Fig. 2.48: Excel: PowerReports - displaying dynamic column titles (Layout)

Displaying dynamic image contents

In PowerReports, you have the option to fill both simple fields as well as list fields with image files (in all formats supported by Excel). Section *Adding image files* (page 29) describes which images you can provide and how you can provide them. A corresponding sample image must be inserted in the report template so that the image files dynamically loaded when generating a report are correctly inserted resp. receive a desired uniform size or position within the cell. Either the image name has to start with "cdb_image" or the attribute name assigned to the associated field by the data provider has to start with "cdbxml_image" so that this image can be recognized by the link as a sample. Furthermore, certain formatting rules in the alternative text box can be defined in the properties dialog of the respective image.

Important: The upper left corner of the sample image has to be definitely placed into the regarding cell, else Excel links the image to another cell and the dynamic image content possibly won't get inserted at all.

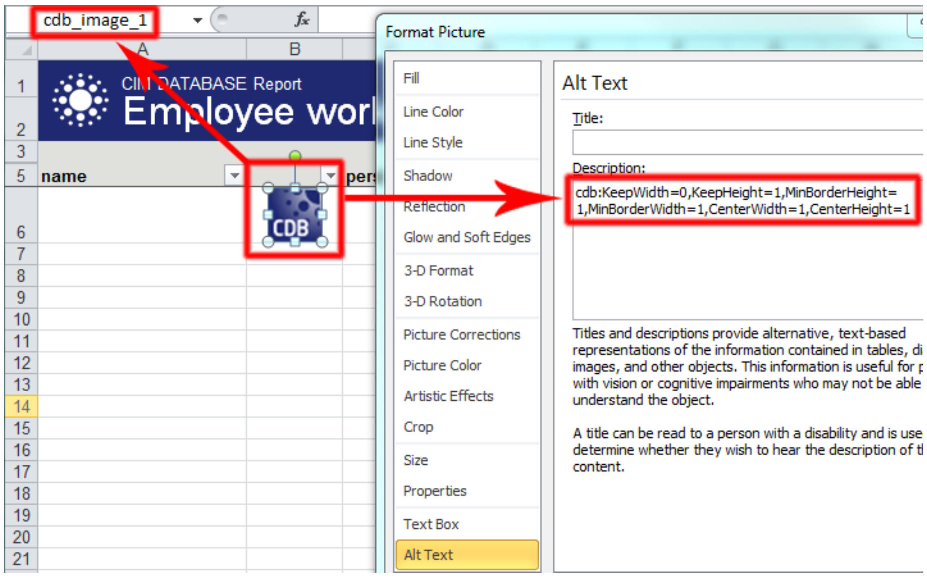
In some cases there are unexpected results in lists with dynamic image content (e.g. distorted or invisible content). Currently this mainly effects Excel 2010 and usually can be avoided by arranging the sample image inside of the cell, so that it (resp. the dynamic image content later on) does not exceed the cell borders. If a PowerReport is intentionally designed in a way so that the user might wish to filter or sort a list which contains dynamic image content, then the regarding sample images have to be formatted with the property "Move and size with cells", else list row movements caused by filtering or sorting might result in wrong image positions.

Important: Dynamic image contents are not currently supported in complex reports.

Formatting options:

- *KeepWidth*: The width of the sample image is transfered to the dynamic image content. This variable is activated internally by default since keeping the column width is usually desired.
- *KeepHeight*: The height of the sample image is transfered to the dynamic image content.
- *MinBorderHeight*: The minimum distance between image height and cell height. For example, a value of "1" guarantees that the respective row height is enlarged when filling images that are too tall. This prevents overlapping image contents. Higher values such as "20" or "30" provide a visual border between the image and the cell.
- *MinBorderWidth*: The minimum distance between image width and cell width.
- *CenterWidth*: The dynamic image content is centered horizontally within the cell.
- *CenterHeight*: The dynamic image content is centered vertically within the cell.

Important: The entire formatting text must begin with the prefix *"cdb:"*.



Alt Text

Title:

Description:

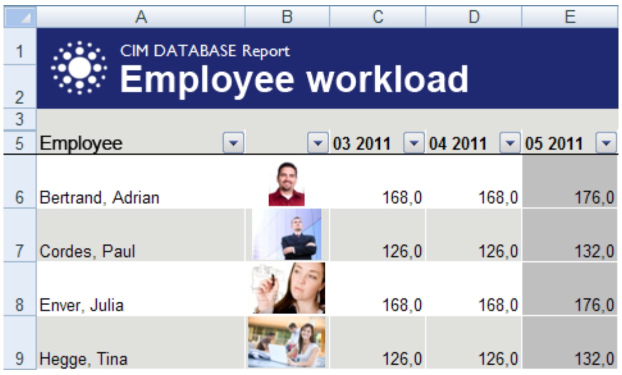
cdb:KeepWidth=0,KeepHeight=1,MinBorderHeight=1,MinBorderWidth=1,CenterWidth=1,CenterHeight=1

Titles and descriptions provide alternative, text-based representations of the information contained in tables, di images, and other objects. This information is useful for p with vision or cognitive impairments who may not be able understand the object.

A title can be read to a person with a disability and is use determine whether they wish to hear the description of tl content.

Fig. 2.49: Excel: PowerReports - dynamic image contents (sample image)

- Formatting example 1: *"cdb:KeepHeight=1,CenterWidth=1"*



	A	B	C	D	E
5	Employee		03 2011	04 2011	05 2011
6	Bertrand, Adrian		168,0	168,0	176,0
7	Cordes, Paul		126,0	126,0	132,0
8	Enver, Julia		168,0	168,0	176,0
9	Hegge, Tina		126,0	126,0	132,0

Fig. 2.50: Excel: PowerReports - dynamic image contents (example 1)

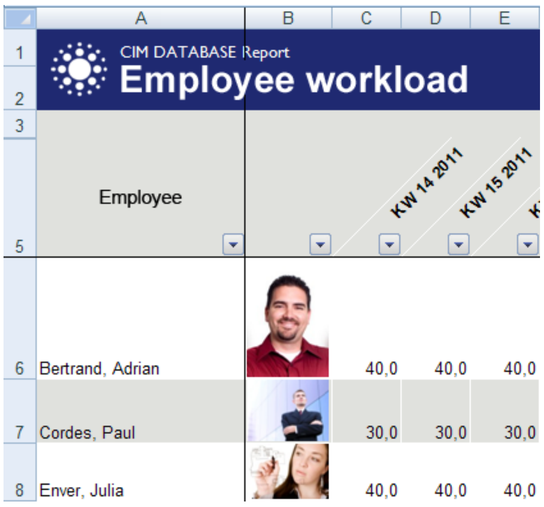
- Formatting example 2: *"cdb:MinBorderHeight=1,KeepWidth=1"*

Displaying complex rows for multi-export data sources

Microsoft Excel does not allow lists consisting of more complex rows to be created. A list row always consists of only one row of Excel cells. However, displaying more complex data in list form is often required. The following display of a part list, each with a list of the assigned documents, cannot be implemented with Excel lists alone, for example. Instead, you have to create these types of lists by joining a layout that repeats itself. This type of complex list element is indicated in red in the figure below. This element repeats as frequently as desired. This is done to form a complex list.

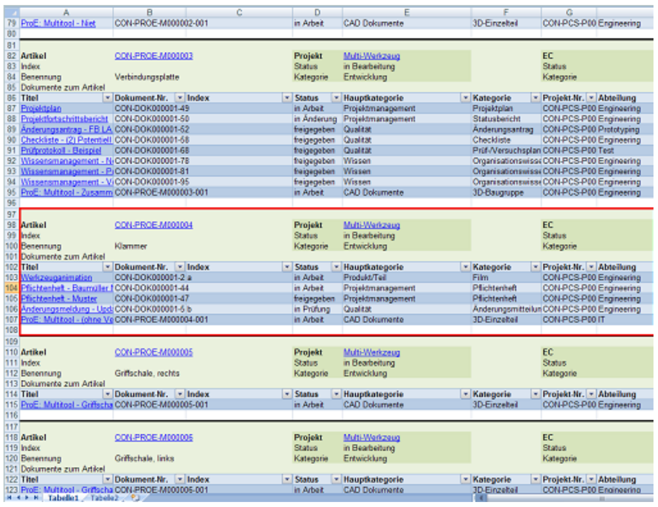
The layout and data connection for a complex list element has to occur on a special template sheet with the fixed name `cdb_template`. Only one complex row is implemented on the template sheet.

When running reports, the complex rows of a data source are exported in individual XML files. They are then imported into the template sheet individually using the Excel link. Between the import processes, the result from the template sheet is copied to the target sheet.



	A	B	C	D	E
1	CIM DATABASE Report				
2	Employee workload				
3					
5	Employee		KW 14 2011	KW 15 2011	K
6	Bertrand, Adrian		40,0	40,0	40,0
7	Cordes, Paul		30,0	30,0	30,0
8	Enver, Julia		40,0	40,0	40,0

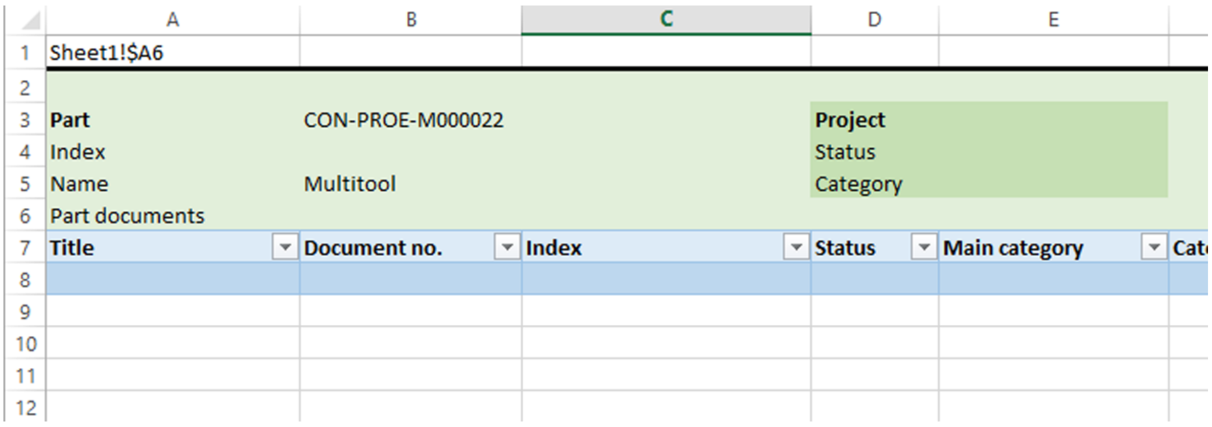
Fig. 2.51: Excel: PowerReports - dynamic image contents (example 2)



	A	B	C	D	E	F	G	
79	ProE: Multitool - Niet	CON-PROE-M000002-001		in Arbeit	CAD Dokumente	3D-Einzelteil	CON-PCS-P00	Engineering
80								
81								
82	Artikel	CON-PROE-M000003		Projekt	Multi-Werkzeug		EC	
83	Index			Status	in Bearbeitung		Status	
84	Benennung	Verbindungsplatte		Kategorie	Entwicklung		Kategorie	
85	Dokumente zum Artikel							
86	Titel	Dokument-Nr.	Index	Status	Hauptkategorie	Kategorie	Projekt-Nr.	Abteilung
87	Projektplan	CON-DOK000001-49		in Arbeit	Projektmanagement	Projektplan	CON-PCS-P00	Engineering
88	Projektfortschrittsbericht	CON-DOK000001-50		in Änderung	Projektmanagement	Statusbericht	CON-PCS-P00	Engineering
89	Änderungsantrag - FB LA	CON-DOK000001-52		freigegeben	Qualität	Änderungsantrag	CON-PCS-P00	Prototyping
90	Checkliste - (2) Potentiell	CON-DOK000001-58		freigegeben	Qualität	Checkliste	CON-PCS-P00	Engineering
91	Prüfprotokoll - Beispiel	CON-DOK000001-68		freigegeben	Qualität	Prüf-/Versuchsplan	CON-PCS-P00	Test
92	Wissensmanagement - N	CON-DOK000001-78		freigegeben	Wissen	Organisationswisse	CON-PCS-P00	Engineering
93	Wissensmanagement - P	CON-DOK000001-81		freigegeben	Wissen	Organisationswisse	CON-PCS-P00	Engineering
94	Wissensmanagement - V	CON-DOK000001-95		freigegeben	Wissen	Organisationswisse	CON-PCS-P00	Engineering
95	ProE: Multitool - Zusamm	CON-PROE-M000003-001		in Arbeit	CAD Dokumente	3D-Baugruppe	CON-PCS-P00	Engineering
96								
97								
98	Artikel	CON-PROE-M000004		Projekt	Multi-Werkzeug		EC	
99	Index			Status	in Bearbeitung		Status	
100	Benennung	Klammer		Kategorie	Entwicklung		Kategorie	
101	Dokumente zum Artikel							
102	Titel	Dokument-Nr.	Index	Status	Hauptkategorie	Kategorie	Projekt-Nr.	Abteilung
103	Werkzeuganimation	CON-DOK000001-2	a	in Arbeit	Produkt/Teil	Film	CON-PCS-P00	Engineering
104	Pflichtenheft - Baumüller I	CON-DOK000001-44		in Arbeit	Projektmanagement	Pflichtenheft	CON-PCS-P00	Engineering
105	Pflichtenheft - Muster	CON-DOK000001-47		freigegeben	Projektmanagement	Pflichtenheft	CON-PCS-P00	Engineering
106	Änderungsmeldung - Upd	CON-DOK000001-5	b	in Prüfung	Qualität	Änderungsmitteilun	CON-PCS-P00	Engineering
107	ProE: Multitool - (ohne Ve	CON-PROE-M000004-001		in Arbeit	CAD Dokumente	3D-Einzelteil	CON-PCS-P00	IT
108								
109								
110	Artikel	CON-PROE-M000005		Projekt	Multi-Werkzeug		EC	
111	Index			Status	in Bearbeitung		Status	
112	Benennung	Griffschale, rechts		Kategorie	Entwicklung		Kategorie	
113	Dokumente zum Artikel							
114	Titel	Dokument-Nr.	Index	Status	Hauptkategorie	Kategorie	Projekt-Nr.	Abteilung
115	ProE: Multitool - Griffscha	CON-PROE-M000005-001		in Arbeit	CAD Dokumente	3D-Einzelteil	CON-PCS-P00	Engineering
116								
117								
118	Artikel	CON-PROE-M000006		Projekt	Multi-Werkzeug		EC	
119	Index			Status	in Bearbeitung		Status	
120	Benennung	Griffschale, links		Kategorie	Entwicklung		Kategorie	
121	Dokumente zum Artikel							
122	Titel	Dokument-Nr.	Index	Status	Hauptkategorie	Kategorie	Projekt-Nr.	Abteilung
123	ProE: Multitool - Griffscha	CON-PROE-M000006-001		in Arbeit	CAD Dokumente	3D-Einzelteil	CON-PCS-P00	Engineering

Tabelle1 Tabelle2

Fig. 2.52: Excel: PowerReports - Complex rows using a part list as an example



	A	B	C	D	E	
1	Sheet1!$A6					
2						
3	Part	CON-PROE-M000022		Project		
4	Index			Status		
5	Name	Multitool		Category		
6	Part documents					
7	Title	Document no.	Index	Status	Main category	Cat
8						
9						
10						
11						
12						

Fig. 2.53: Excel: PowerReports - Definition of a complex row

The target sheet and a target cell can be specified in the top left cell of the template sheet. This target address describes the target for the top left cell of the first complex cell to be copied. All other rows are attached to each other starting there. If no target address is specified, the top left cell of the first sheet contained in the workbook is used as a starting cell.

The PowerReports Framework automatically identifies data that has to be imported and exported as complex rows based on the result and call cardinalities of linked data providers.

A multi-export configuration exists whenever the higher-level provider provides a list as a result (result cardinality N) and if an assigned subprovider has the call cardinality 1. Thus, the subprovider provides data that relates to individual list elements of the higher-level provider.

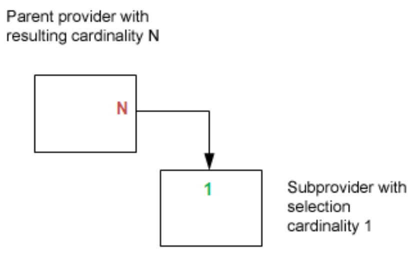

Parent provider with resulting cardinality N
N
1
Subprovider with selection cardinality 1

Fig. 2.54: Excel: PowerReports - schema for multi-export

For example, if a SimpleQuery provider provides a list of parts and a relationship provider provides the assigned documents as an assigned subprovider, a configuration exists that can only be displayed as a complex list in Excel.

An assigned GroupBy provider, on the other hand, would not represent a multi-export configuration since GroupBy providers are called up with the entire result of the higher-level provider, i.e. have the call cardinality N. GroupBy providers do not provide results that relate to individual list elements of the parent provider, but rather aggregated information that relates exclusively to the entire list.

Displaying longtexts

For rows where cells are only or sometimes filled with longtexts, the property *line break* " of the *complete row* has to be set so that the link recognizes this and can adjust its row height automatically.

If longtexts are to be shown in cells consisting of multiple composite cells, the correct display can only be achieved via a detour:

1. The actual link with the longtext attribute of the data source must not occur with the desired target cell, but rather with a *hidden cell* at the end located in the *same row*.
2. So that the longtexts are not displayed as cut off with empty rows at the beginning, this helper cell must have the *same width*.
3. The "*line break* " property has to be set for the entire row.

Re-editing finished PowerReports using VBA

Sometimes, you might want to carry out certain programming tasks using VBA and the Excel object model after a report has been completed by the link (e.g. to format filled diagrams correctly or put the focus on something). For this purpose, the link searches for a VBA function with a certain name in the Excel workbook. If it is found, it is called up each time after completing/updating a report.

Adding a VBA function for automated re-editing of finished PowerReports

1. Open the VBA editor in Excel.
2. Insert a module in the PowerReport template.

3. Integrate a function with the name 'CDBReportDone':

```
Public Sub CDBReportDone()
    ' ...
End Sub
```

Note: If the security settings in Office are set to automatically disable macros which are not digitally signed, then the report templates must be signed with a certificate before they are stored in the system. The standard PowerReports supplied with the system, which contain VBA code for post-processing, are already signed with a Contact GmbH certificate and under usage of a timestamp server. In order to sign custom report templates or modified standard report templates with an own certificate and under usage of a timestamp server, following registry entries must be manually set, since it is not possible to configure a certificate timestamp server via the Office interface:

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\VBA\Security]
"TimeStampURL"="http://timestamp.some-ca.com/scripts/timestamp.dll"
"TimeStampRetryCount"=dword:00000005
"TimeStampRetryDelay"=dword:00000005
```

If the digital signing is proceeded without a timestamp, then the validity of the signature expires after the expiry of the certificate. By using a timestamp the signature is valid without any time limitation.

Creating personal reports

Users have the option to modify the layout and data connection of a called up report directly and save it as a personal report. Thus, a personal report can be created spontaneously if the user is not pleased with report result or if it is not suitable for his/her purposes.

The modified report is saved as a personal report with the Excel link function Create *Reports* → *Create report template*. The modified report is created by calling up this function as a new report template for the underlying data source and authorized for the user implementing this. Other users do not have access to this report, but could become authorized later by the administrator via role and personnel assignments.

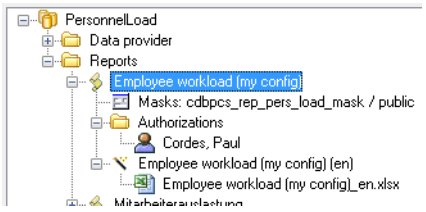

PersonnelLoad
Data provider
Reports
Employee workload (my config)
Masks: cdbpcs_rep_pers_load_mask / public
Authorizations
Cordes, Paul
Employee workload (my config) (en)
Employee workload (my config)_en.xlsx

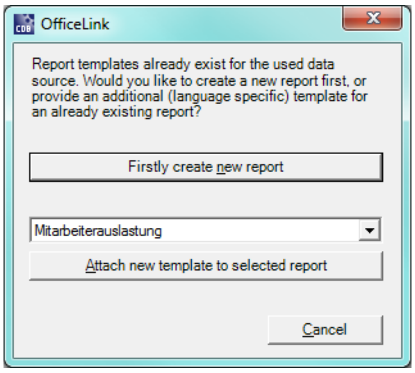
Fig. 2.55: Excel: PowerReports - Personal report using employee workload as an example

When creating a personal report, the preceding report dialog, if present, is taken over by the original report. The change of the report dialog is not provided by the user.

Creating language-specific reports

The designer of a PowerReport has the option to adapt an existing report template for a certain language and then provide it for the same higher-level report so that the users running the report later can choose between the different languages. If you open an already created report template for editing and run *[Create template]* later, the link, in an intermediate dialog, offers to create a language-specific report template for already existing reports.

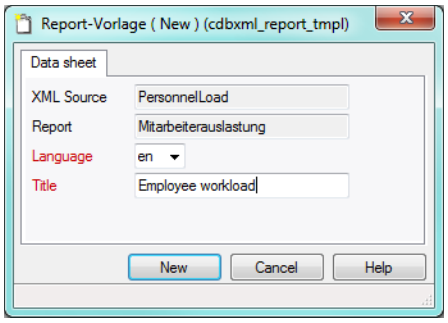
In the dialog following this, you only have to select the language and a unique title.


OfficeLink

Report templates already exist for the used data source. Would you like to create a new report first, or provide an additional (language specific) template for an already existing report?

Firstly create new report
Mitarbeiterauslastung
Attach new template to selected report
Cancel

Fig. 2.56: Excel: PowerReports - Report for data source


Report-Vorlage (New) (cdbxml_report_tmpl)

Data sheet

XML Source	PersonnelLoad
Report	Mitarbeiterauslastung
Language	en
Title	Employee workload

Fig. 2.57: Excel: PowerReports - Report for data source

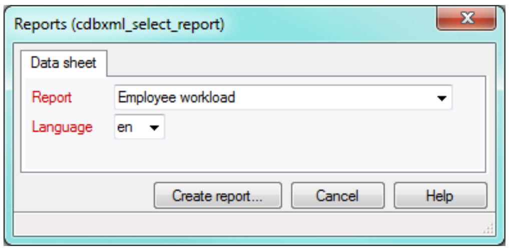
Special notes

- When linking between list attributes of a provider scheme and the Excel cells, i.e. if table columns develop due to the linking in the sheet, you have to take particular care that the respective table is set up as a whole in a logical sequence from left to right. If columns are deleted or inserted from this table later, this can cause the report generation to have errors or even be aborted in some cases, since the columns potentially no longer form an entire table in this case, but rather multiple tables next to each other. You can tell whether a table consists of multiple tables by the small triangles on the bottom right edge of each table. Especially since Excel 2010, this has been giving users problems that can only be solved by removing tables and setting up new ones.
- Filling data in composite cells can cause unexpected malfunctions. In these cases, you can use the option “Text alignment > Horizontal > Center via selection” instead of the “Combine cells” option.
- For cells preconfigured with date formats or user-defined number formats, the option “Line break” must not be set at the same time since this can lead to a loss of the formatting.
- When linking between a schema attribute and a cell, Excel formats the cell automatically with the number category that fits the respective attribute value type. Since now, cells that are to be filled with floating points later are assigned the number category ‘standard’ by Excel automatically, potential decimal places are only displayed if they do not equal zero. If a fixed decimal place format (0.00) is desired, you have to change the number category according to the link in ‘Number’.
- A lower table must never be wider than the table(s) above it if multiple dynamically filled tables are used on top of each other in any sheet. Thus, it may only be as wide or, of course, narrower since internal problems will be caused otherwise, which would cancel the report generation as a result. If lower tables have to be wider based on their design, the higher tables have to be broadened, for example, by using invisible dummy columns.

2.4.7 Running PowerReports

Selection dialog

After selecting a report pop-up menu function, the dialog opens for selecting the report to be run. The list of options for reports lists all reports available for the respective user automatically. If you use multi-select, single-select reports are automatically hidden.



Report: Employee workload
Language: en

Fig. 2.58: Excel: PowerReports - Report selection dialog

The combo boxes for selecting the report and the language are preset with the respective registration language automatically. If no reports are available in the log-on language, the (configurable) standard languages are used instead. If reports are also not available for these languages, the dialog is preassigned with the next available language. Chapter *Creating language-specific reports* (page 42) explains how to create templates in additional languages for a report.

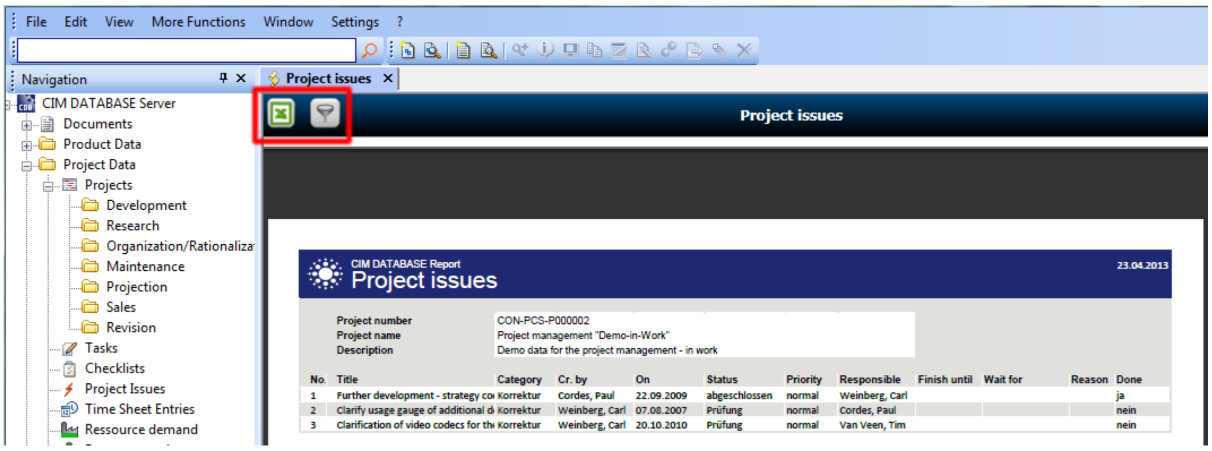
Types and formats for running

Report execution

- Client
  - Possible report formats: *Excel*
  - Condition: The link must be installed on the workstation computer.
- Server (asynchronous)
  - Possible report formats: *Excel, PDF, Excel & PDF*
  - The link does not have to be installed on the workstation computer.
  - The completed PowerReports are sent to the user as via e-mail by default.
- Server (synchronous)
  - Possible report formats: *Excel, PDF, E-Link*
  - The link does not have to be installed on the workstation computer.
  - The completed PowerReports are opened on the workstation computer for viewing directly after they are ready.

Note: The generation on the server side is only possible if its CDB administrator installed and activated the respective services.

In CDB versions 9.8 *SP1* and later, all completed PowerReports can be opened in a separate tab within the PC client (as eLink-Panel). A button with an Excel symbol is located within the header, which opens the displayed report in Excel, if needed. If the report has an additional dialog, a second button can be seen with a filter symbol that re-runs the respective report after showing the intermediate dialog.



CIM DATABASE Report
Project issues
23.04.2013

Project number	CON-PCS-P000002
Project name	Project management "Demo-in-Work"
Description	Demo data for the project management - in work

No.	Title	Category	Cr. by	On	Status	Priority	Responsible	Finish until	Wait for	Reason	Done
1	Further development - strategy co	Korrektur	Cordes, Paul	22.09.2009	abgeschlossen	normal	Weinberg, Carl				ja
2	Clarify usage gauge of additional d	Korrektur	Weinberg, Carl	07.08.2007	Prüfung	normal	Cordes, Paul				nein
3	Clarification of video codecs for th	Korrektur	Weinberg, Carl	20.10.2010	Prüfung	normal	Van Veen, Tim				nein

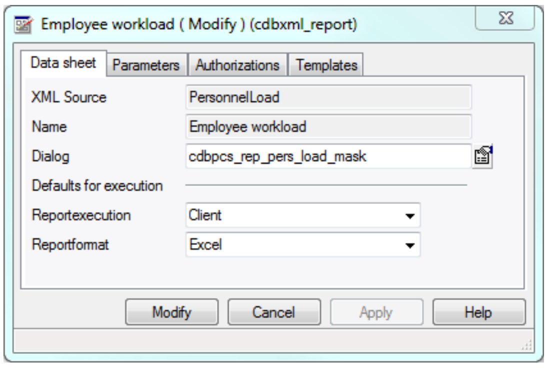
Fig. 2.59: Excel: PowerReports - Report selection as eLink-Panel

Note: Adobe Acrobat must be installed so that the report display works as a eLink-Panel on the workstation computer.

Note: The CIM Database/WEB client can only be used to generate PowerReports with the run type *Server (asynchronous)*.

Selection configuration

In CDB versions 9.8 *SP1* and later, you can configure for each report separately whether a PowerReport, for example, is to always be displayed directly in the PC client as a eLink-Panel panel or opened in Excel. You can also configure whether it is to be sent as an e-mail only after completion in Excel and PDF formats for performance reasons.



XML Source	PersonnelLoad
Name	Employee workload
Dialog	cdbpcs_rep_pers_load_mask
Defaults for execution	
Reportexecution	Client
Reportformat	Excel

Fig. 2.60: Excel: PowerReports - Report selection configuration

Note: *Report execution* and *report format* can then also be changed in the report selection dialog, even though these fields are hidden there by default.

Outlook

3.1 Introduction

- *Outlook interface:*

  The installation extends the Outlook application by adding the OfficeLink menu and toolbar.

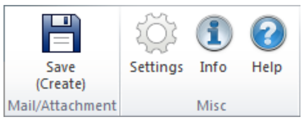
  

  Fig. 3.1: Outlook: OfficeLink toolbar

- *Create e-mail/attachment:*

  The system Microsoft Office integration provides Outlook with the option of archiving individual messages directly from Outlook in the form of a `MSG` file as a document in the system. In Outlook, `MSG` files represent a complete message object including all attachments. The extension also provides the user with the option of separately archiving any received attachments to the respective (selected) message in the system and/or the message itself. The user receives the option—in the form of a dialog—of selecting which elements are to be archived separately. In addition, a decision can be made with each complete archiving procedure as to whether separately archived attachments are to be extracted from the message (`MSG` file) before these themselves are archived in the system. Furthermore, as part of the archiving procedure the user is offered the option of whether successful archiving is to be followed by automatic deletion of the message file in Outlook.

  For each individual file to be archived (both message and attachments), the create mask is displayed in CIM Database/WIN. The create mask for the message provides the option of prepopulating individual fields with information (such as *Subject* or *Sender*) of the underlying message by configuring the system.

  Note: If attachments are not created as documents, but appended to the e-mail documents as files in CIM Database/WIN, the create mask for these files is automatically filled-in and skipped.

  Note: The assignment configuration of the metadata of a message and the respective system attributes of a create mask is taken care of by the system administrator.

- *Append document:*

  The add-in also provides the option of appending documents from the system directly to the new message as an attachment when composing a new message. This process is initiated via a toolbar in the message window.

- *Settings:*

  The OfficeLink administration manual includes the settings options in the option dialog. Contact your system administrator if you have questions.

3.2 E-mails and attachments

3.2.1 Create messages in the system

In general, a message from Outlook can be created in the system with the following steps:

1. Select the messages to be archived by marking them.

   Note: The *[Create e-mail/attachment]* button is enabled if at least one message is marked. Thus it is also possible for multiple elements to be marked and created simultaneously.

2. Click the enabled *[Create e-mail/attachment]* button.

   A search is carried out to see if an CIM Database/WIN instance is already running. If no CIM Database/WIN has been started, there is an automatic attempt to open a new program instance.

   Note: It is generally recommended that an instance of CIM Database/WIN be started before beginning the archiving procedures.

3. Pressing the *[Create e-mail/attachment]* button causes a *Selection dialog* to appear. This dialog provides options that can be used to archive the selected messages. Sender, Subject, Date and Time are specified again in the e-mail node. If the message to be archived contains attachments, then these are listed subordinately in the table of the dialog.

   Important: Messages that contain no attachments are not listed in the selection dialog. They are handled in accordance with the *Default action for e-mails* from the system integration (see settings dialog). If none of the selected messages contains attachments, then the selection dialog is skipped.

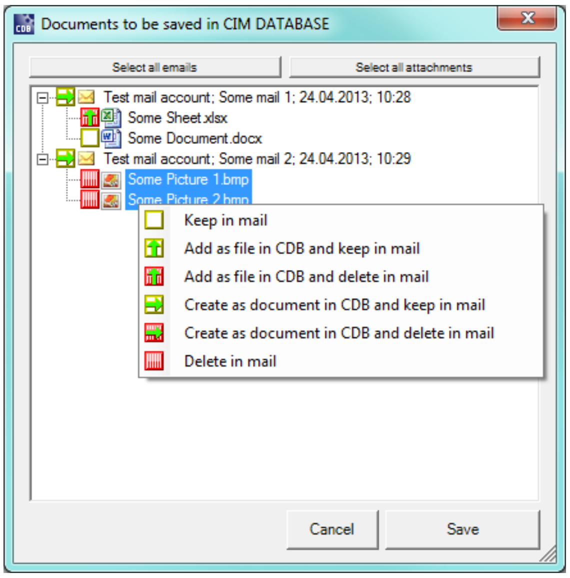

Documents to be saved in CIM DATABASE
Select all emails
Select all attachments
Test mail account; Some mail 1; 24.04.2013; 10:28
Some Sheet.xlsx
Some Document.docx
Test mail account; Some mail 2; 24.04.2013; 10:29
Some Picture 1.bmp
Some Picture 2.bmp
Keep in mail
Add as file in CDB and keep in mail
Add as file in CDB and delete in mail
Create as document in CDB and keep in mail
Create as document in CDB and delete in mail
Delete in mail
Cancel
Save

Fig. 3.2: Outlook: Messages with attachments in the archiving selection dialog

4. This dialog provides the option of deciding whether the attachments to a message are to be extracted from the message, that is, created separately in the system.

   The user has the option of selecting the elements in the dialog table. It is possible to select one or more elements. Likewise, the *[Select all e-mails]* and *[Select all attachments]* control elements at the upper edge of the dialog can also be used if the user would like to select/change an action for all e-mails or attachments. The actions are changed either by simply clicking the action symbol, or by using the pop-up menu, which appears as soon as you right-click a selection marker.

   Note: The extracting/deleting of attachments has an effect only on the message copy to be archived in the system. The original message in Outlook remains unchanged, unless the user selects the option for the respective message to have it automatically deleted in Outlook after the archiving procedure.

   Note: If the same action is always executed for e-mails or attachments, this can be preconfigured for faster work in the settings dialog. There, the selection dialog can also be entirely deactivated (see OfficeLink administration manual). Default action for pictures in HTML-Emails is always "Keep in mail".

   Note: A grayed out element represents an embedded object within an e-mail. The only valid action for this objects is "Keep in mail". The action cannot be changed. Example: pictures embedded in e-mails in Rich Text Format (RTF).

5. After making the selection, confirm the dialog using the *[Save]* button. Then the mask for creating documents appears in the system for every file (message/attachment) to be archived. The individual fields of the create mask are prepopulated corresponding to the configuration. The original default mask settings are overwritten with the configured values in the process. Then the user has to assign attributes and confirm each mask with the respective necessary parameters added. If attachments are not created as documents, but appended to the e-mail documents as files in CIM Database/WIN, the create mask for these files is automatically filled-in and skipped.

   Note: If necessary, depending on the settings configured, the attachments are extracted from the message to be archived and/or the original message in Outlook is automatically deleted in the end.

3.2.2 Append system documents as message attachment

In general, system documents can be appended to a message to be composed from Outlook with the following steps:

1. Create a new e-mail message via the Outlook main menu or the corresponding button in the default toolbar.

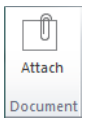


Fig. 3.3: Outlook: 'Append' control element

2. Pressing this button initiates a system document search and the document search mask appears. Here you can configure parameters for the search mask as usual with corresponding search criteria. After confirming the mask you get a dialog with the resulting document result list.

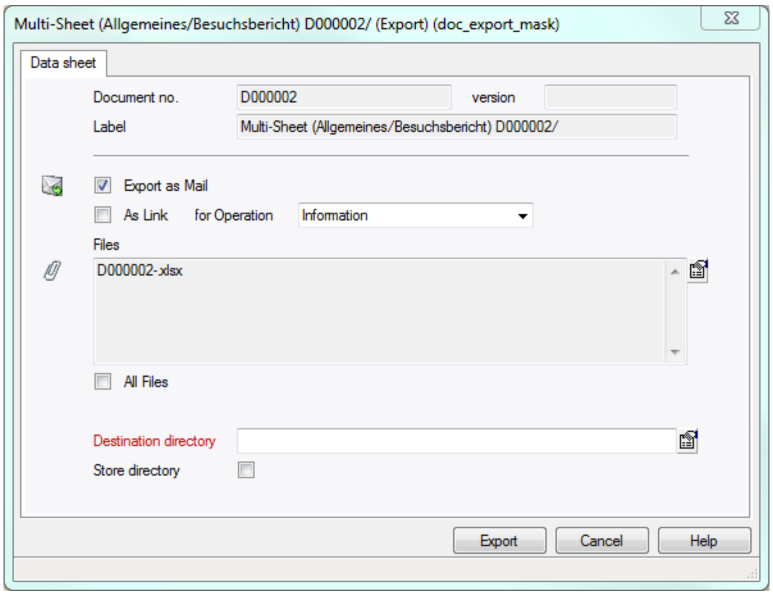
   Select one or more documents from the result list you want to send.

Since documents can be assigned multiple files, it is possible that you will get another selection dialog. The selection process for documents with additionally assigned files can be further automated in the settings dialog (see OfficeLink administration manual).

Note: The export right is checked for the selected system documents.

3.2.3 Messages from the document export function of CIM Database/WIN

With the system operation ‘Export documents as e-mail’, a cdbinfo file, which is given to the message to be sent as an additional attachment, is also exported in the XML format in addition to the selected documents. Messages that were originally sent this way can likewise be archived. Based on the cdbinfo file, the link identifies which attachments come from the system and does not offer these again for archiving. After the outgoing message has been archived, the references between the message and the attachments already in the system are written.


Multi-Sheet (Allgemeines/Besuchsbericht) D000002/ (Export) (doc_export_mask)

Data sheet

Document no.	D000002	version	
Label	Multi-Sheet (Allgemeines/Besuchsbericht) D000002/		

[✓] Export as Mail
[ ] As Link for Operation: Information

Files
D000002-xlsx

[ ] All Files

Destination directory:
Store directory [ ]

Fig. 3.4: Outlook: ‘Export as e-mail’ in CIM Database/WIN

CHAPTER 4

PowerPoint

4.1 Introduction

- *PowerPoint interface:*

  The installation extends the PowerPoint application by adding the OfficeLink toolbar.

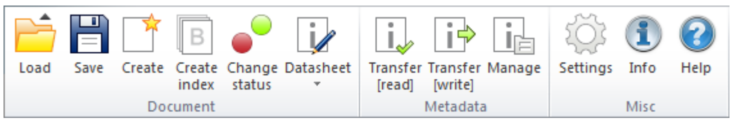


Fig. 4.1: PowerPoint: OfficeLink menu bar

- *Standard operations:*

  The system Microsoft Office integration provides PowerPoint with the option of searching for documents in the system, opening them for editing and then saving changes in the database. Moreover, it is possible to change the document status or properties, or create new documents in the system directly from PowerPoint.

- *Data synchronization:*

  The link gives you the ability to synchronize data with the system. A prerequisite for this is the use of prepared document templates which contain corresponding system variables (see *Create templates for data synchronization* (page 52)). Data synchronization is always carried out for the active document. The active document has to have been opened from the system for editing.

- *Settings:*

  The OfficeLink administration manual includes the settings options in the option dialog. Contact your system administrator if you have questions.

4.2 Editing documents

4.2.1 Searching for documents in the system

The *[Load]* button opens the document search mask in the system. The default search condition automatically limits the search to PowerPoint documents. The result list opens with a preview window after entering additional search criteria and running the search. After selecting one or multiple documents, they are opened for editing.

4.2.2 Save in the system

The *[Save]* button saves the changes to the active document directly in the system.

4.2.3 Create in the system

The *[Create]* button carries out a document creation in the system for the active document. This process initially opens the mask for creating documents. After filling out and confirming the create mask, the document is committed to the system and finally re-opened from the system for editing. The original document is closed after the create is successful. In the case of an error, for example due to deficient authorizations in the system, the document remains open.

Note: Since multiple file formats are supported, the save format is determined there first, before the system document management create mask is opened: *Use as default format:* Enabling this check box defines the selected format as the default save format. The selection dialog is no longer shown when creating documents in the future and the default save format is used instead. The default save format can be changed or removed in the settings dialog at any time. *PowerPoint Open XML file format and macros:* No macros can be saved in the .pptx Open XML file format of PowerPoint. When attempting to save documents with macros in this format, a corresponding message is displayed.

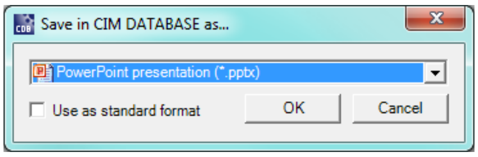


Fig. 4.2: Microsoft PowerPoint: Save format selection dialog

4.2.4 Create index in the system

The *[Create index]* button creates a new index in the system for the active document. This process initially opens the mask for creating indices. After filling out and confirming the index mask, the current changes are backed up in the new document index in the system.

4.2.5 Change document status

The *[Change status]* button carries out a status change operation in the system for the active document.

Note: Since the user potentially loses the rights to re-save the document in the system by changing the status, the link saves the document automatically before calling up the status change operation. If necessary, a corresponding message is displayed here that you have to confirm.

4.2.6 Opening master data change dialog in the system

The *[Master data]* button opens the dialog in the system for changing the master data for the active document. This button is divided into two parts. The bottom part can be used in the system to open an information dialog for the active document instead of the modify dialog.

4.3 Synchronizing data

4.3.1 Metadata takeover from the system

By pressing the *[Transfer [read]]* button, the values of all system variables with read access are updated with the corresponding values of the associated system attributes. This overwrites system variables with read access changed by the user in the document. If an error exists in the expression of a variable, the referenced object does not exist or the relationship is not unique, the value is set to a space character.

4.3.2 Metadata synchronization with the system

By pressing the *[Transfer [write]]* button, the values of all variables with write access are checked for changes. If values have been changed in the document, the affected objects are opened in the system sequentially. The changes are shown in the modify dialogs and have to be confirmed for each object. After completing the data transfer to the system, a metadata acceptance is carried out for system variables with read access.

4.3.3 Create templates for data synchronization

This chapter describes creating document templates with system variables for data synchronization.

- Synchronization of document metadata (e.g. title, document No., etc.)
- Synchronization of 1:1 referenced objects (e.g. project name of the assigned project)

Format description for system variables

System variables in PowerPoint documents have the following format: cdb.<Mode>.<Relation>.<Attribute>.<Cardinality>

Mode The mode indicates whether a variable may be read from the system exclusively or updated after changing in the document in the system. Valid values for the mode include:

- *r*

  Only data transfer from the system possible. The transfered value can be changed in the document, but not transferred to the system. The changed value in the document is replaced again by the system value when transferring the data again.

- *w*

  Only data transfer to the system possible. This mode can be used to ensure that a document field cannot be changed by a data transfer, though an update is possible in the system (useful for field calculated by the document). This mode cannot be used for 1:N relationships.

- *rw*

  Data transfer from the system and update in the system possible after changing in the document. This mode cannot be used for 1:N relationships.

- **s*

  If a *s* is added to the above-mentioned mode values (i. e. *rs*, *ws* or *rws*), then the same rules apply as for the respective values without *s*, but the read/write actions run completely on the server side. The *Parameter* field can be used optionally. You might, however, you need administrative customizing to use server-side editing of variables, which is explained in the OfficeLink Administration Manual.

Relationship Starting from the document, the relationship indicates the path for the data in the system represented by the variable. The following expressions can be used as a relationship:

- *this*

  The constant expression ‘this’ is used to specify the attributes of the document itself.

- *Name of a relationship configured in the system*

  Attributes of referenced objects can be transferred and synchronized via a relationship.

- *BY_ZNUM_ZIDX_FROM_<table_name>*

  Use an implicit relationship via the attributes z_nummer and z_index. To do so, the expression BY_ZNUM_ZIDX_FROM_<table_name> is specified. table_name indicates the database table used to reference an object implicitly via z_nummer and z_index. The specified database table must contain the attributes z_nummer and z_index and the combination of z_nummer and z_index must be unique.

Attribute The attribute specifies an attribute of a system object referenced by the relationship.

Subject characteristics as attributes of parts:

If a part relationship is specified as a relationship, the subject characteristics of the part can also be specified as an attribute. This can be carried out in a qualified manner via property identification or unqualified using the sequence number:

- *Qualified subject characteristics:*

  `_SML_<mm_mk>`

PowerPoint: data synchronization - subject characteristics qualified

Subject characteristics with the identification 'L' of the associated part

```
``cdb.r.cdb_doc_to_part._SML_L``
```

- *Unqualified subject characteristics:*

  `_SMLN<nummer> for characteristic identifications`

  `_SMLV<nummer> for characteristic values`

  Property identification and the associated value must have the same number for this.

PowerPoint: data synchronization - subject characteristics unqualified

Identification and value of the first subject characteristic of the associated part

```
``cdb.r.cdb_doc_to_part._SMLN1``
``cdb.r.cdb_doc_to_part._SMLV1``
```

Important: SC attributes cannot be synchronized with write access!

Cardinality Specifies the cardinality of the relationship. If no cardinality has been specified, cardinality 1 is assumed. The cardinality should be specified explicitly when creating new document templates. The following values can be used for the cardinality:

- *1*

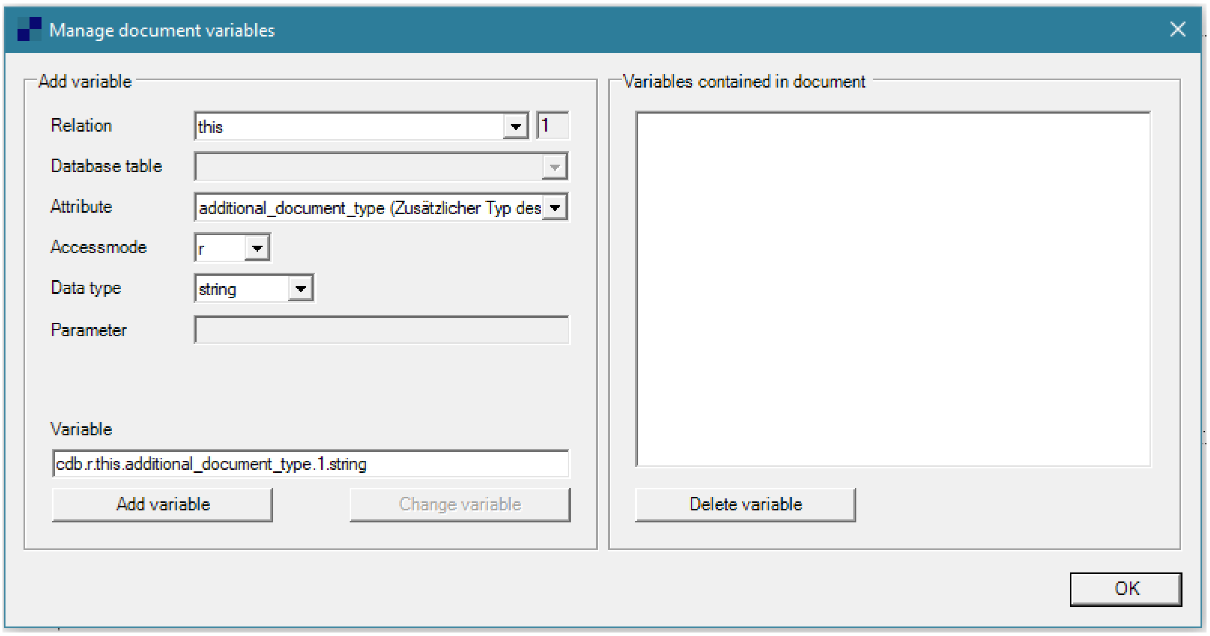
  For relationships of the type `this,BY_ZNUM_ZIDX_FROM_<table_name>` and for 1-digit homogeneous relationships.

Parameter This field can be used to pass additional constant values specific to the respective variable when processing variables on the server side. These can be identifiers, for example, that allow you to link results from 1: N relationships with individual document fields instead of dynamic lists. For example, the parameter *PRIMARY* is added to the following variable, which tells a user exit that you want to have the original source name of the document's primary file.

```
cdb.rs.document2cdb_file.cdbf_original_name.N.string.PRIMARY
```

Definition of system variables

System variables are defined using a wizard that calls up the OfficeLink toolbar using the *[Manage]* button. The button has to have been enabled beforehand in the settings dialog so that the button is available. The dialog provides the option of creating new variables in the system format on the left side (see *Format description for* *system variables* (page 52)). On the right side, all system variables contained in the document are displayed. By marking a variable, the corresponding text field is marked in the PowerPoint document at the same time. The marked variables are removed from the document by pressing the *[Delete variable]* button.


Manage document variables

Add variable

Relation	this 1
Database table	
Attribute	additional_document_type (Zusätzlicher Typ des
Accessmode	r
Data type	string
Parameter	

Variable

cdb.r.this.additional_document_type.1.string

Fig. 4.3: PowerPoint: data synchronization - dialog for creating system variables

- *Relationship:*

  Starting from the document, the relationship indicates the path for an attribute of one or more referenced system objects. The catalog provides all document relationships configured in the system, for which the document is the start object. For attributes of the document itself, you have to enter `this` in this field.

  To use an implicit relationship that is produced via the key attributes document number and index, you have to select `BY_ZNUM_ZIDX_FROM_` in this field and select a relation as a data source in the database table field.

- *Database table:*

  A database table has to be selected only if the special entry `BY_ZNUM_ZIDX_FROM_` for implicit relationships has been selected in the relationship field. In the selected relation, the combination of the attributes z_nummer and z_index has to be unique.

- *Attribute:*

  The attribute specifies an attribute of a system object referenced by the relationship.

- *Synchronization mode:*

  The mode indicates whether a variable may be read from the system exclusively or may be updated after changing in the document in the system.

  r: read only

  w: write only

  rw: read and write access in the system

  rs: server-side read only

  ws: server-side write only

  rws: server-side read and write access in the system

- *Parameter:*

Optional value used only for server-side editing of variables.

- *System variable:*

  In this field, the expression generated from the other fields is shown for the system variables to be created. The manual modification of this expression is only required for specifying subject characteristics for part relationships.

When adding the variables by pressing the *[Add variable]* button, the expression shown in the system variable field is reviewed and displays a corresponding error message if invalid. Otherwise, the variables are created as new system variables in the active document and initialized with the `<Attribut>` value as a placeholder.

Important: The system variables can be positioned only in simple PowerPoint 'forms'. Furthermore, only one system variable can be defined per PowerPoint form. An error message appears if there is an attempt to define a new variable without the simple form being enabled, or if the currently enabled form already contains a variable.

Modifying system variables

The text fields containing system variables can be modified by clicking in the field to be modified. Then the cursor is in the field and enables free editing.

Using system variables in the document

System variables should be identified for the user in the document to avoid accidental deleting or overwriting. The protection of document variables is possible only on condition. One option is to create forms.

Note: During the data synchronization of variables to the system, attribute changes are also shown if the field cannot be changed by the user interactively according to the mask configuration or would only be able to be changed using a list of options. You have to take this into account when using system variables with write access. If necessary, the validity of the values must be ensured through a corresponding check in the user exit.

Updating metadata automatically

The metadata of all system variables with read access are committed from CDB automatically when loading a document. The conditions for this are that the document:

1. Has been opened in the client in "Edit" mode (or directly from Office via "Load")
2. Is a "Drawing" (i.e. has the property "vorlage_kz")
3. Is not a template (i.e. "vorlage_kz=0")

Project

5.1 Introduction

- *Project interface:*

  The installation extends the Project application by adding the OfficeLink toolbar.

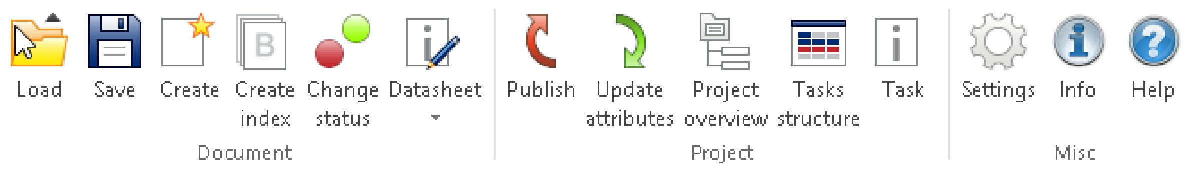


Fig. 5.1: Project: OfficeLink menu bar

- *Standard operations:*

  The system Microsoft Office integration provides Project with the option of searching for documents in the system, opening them for editing and then saving changes in the database. Moreover, it is possible to change the document status or properties, or create new documents in the system directly from Project.

- *Project synchronization:*

  The system integration for Microsoft Office enables projects that are managed by the system PCS to be edited from Microsoft Project. Project data is synchronized partly bidirectionally in the process.

- *Settings:*

  The OfficeLink administration manual includes the settings options in the option dialog. Contact your system administrator if you have questions.

5.2 Editing documents

5.2.1 Searching for documents in the system

The *[Load]* button opens the document search mask in the system. The default search condition automatically limits the search to Project documents. The result list opens with a preview window after entering additional search criteria and running the search. After selecting one or multiple documents, they are opened for editing.

5.2.2 Save in the system

The *[Save]* button saves the changes to the active document directly in the system.

5.2.3 Create in the system

The *[Create]* button carries out a document creation in the system for the active document. This process initially opens the mask for creating documents. After filling out and confirming the create mask, the document is committed to the system and finally re-opened from the system for editing. The original document is closed after the create is successful. In the case of an error, for example due to deficient authorizations in the system, the document remains open.

5.2.4 Create index in the system

The *[Create index]* button creates a new index in the system for the active document. This process initially opens the mask for creating indices. After filling out and confirming the index mask, the current changes are backed up in the new document index in the system.

5.2.5 Change document status

The *[Change status]* button carries out a status change operation in the system for the active document.

Note: Since the user potentially loses the rights to re-save the document in the system by changing the status, the link saves the document automatically before calling up the status change operation. If necessary, a corresponding message is displayed here that you have to confirm.

5.2.6 Opening master data change dialog in the system

The *[Master data]* button opens the dialog in the system for changing the master data for the active document. This button is divided into two parts. The bottom part can be used in the system to open an information dialog for the active document instead of the modify dialog.

5.3 Project synchronization

5.3.1 Function

The Microsoft Project integration enables project plans managed by the system PCS to be edited in Microsoft Project. To do so, Microsoft Project is expanded by adding the following functions:

- Publish the project structure in the system starting from the project plan
- Update specific task attributes with current information from the system
- Display project and task overviews from the system for loaded project plans in Microsoft Project
- Directly display an information sheet in the system of a task selected in Microsoft Project

When publishing project plans, a partly bidirectional synchronization of all project data and tasks occurs. This includes project name, project scheduling, task structure, task relations and milestones, so that the shared subset of all data fields between the system and Microsoft Project can be completely covered.

Important: It is also necessary to look up further information in the Projects User Manual.

5.3.2 Publishing Microsoft Project plans to the PCS system

By "Publish" the project structure stored in the PCS system can be updated by the Microsoft Project plan. The changes to the project plan are transferred to the configured system relations, new tasks are created accordingly and deleted tasks are removed from the project structure. After successfully importing resp. updating in the PCS system, both project plans have the same status.

During this data reconciliation, the changed Microsoft Project plan is stored in the system and is also exported in Microsoft Project XML format and stored as an XML file next to the project plan in the system. Afterwards, an import process is automatically executed in the system using the XML file just created. For more information about the preview dialog that now opens, refer to the Projects User Manual.

5.3.3 Updating of task attributes from the system PCS

During "Update attributes", the project plan is updated with specific task attributes currently saved in the system. There is no data transfer to the system. By default, only a few attributes are transferred from the system to Microsoft Project:

Table 5.1: Updated task attributes

MSP Task field	PCS task field
Number10	Status
Text12	Reference id
Text14	Responsible person
Text15	Status (text)

To customize the list of these updated task attributes, see the OfficeLink Administration Manual.

Important: Consistent data reconciliation on both sides cannot be guaranteed if the list is extended by task attributes that indirectly change the project structure again, such as task start, duration, end, predecessor, successor or similar.

5.3.4 General tips

Note: If you edit a project plan the first time using the Microsoft Project integration, synchronization information is saved in the Microsoft Project file. Even without manual changes, therefore, Microsoft Project asks whether changes to the file should be saved upon quitting.

Note: Relationships between tasks of different projects are not synchronized. The synchronization of projects with such relationships is possible, but they are not visible in MS Project and are ignored during synchronization. If a task is deleted in MS Project, all relationships will be deleted in the system during synchronization. No separate message is displayed.

Visio

6.1 Introduction

- *Visio interface:*

  The installation expands the Visio application by adding the OfficeLink toolbar.

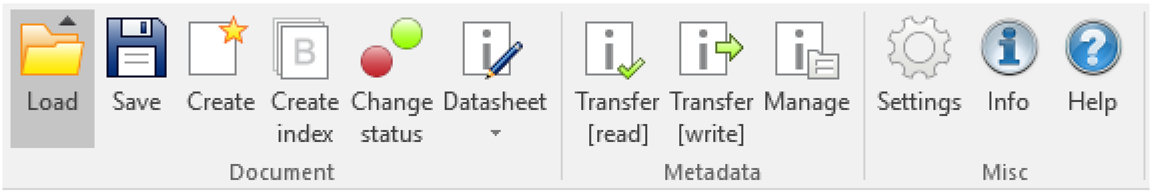


Fig. 6.1: Visio: OfficeLink menu bar

- *Standard operations:*

  The system Microsoft Office integration provides Visio with the option of searching for documents in the system, opening them for editing and then saving changes in the database. Moreover, it is possible to change the document status or properties, or create new documents in the system directly from Visio.

- *Data synchronization:*

  The link gives you the ability to synchronize data with the system. A prerequisite for this is the use of prepared document templates which contain corresponding system variables (see *Create templates for data synchronization* (page 61)). Data synchronization is always carried out for the active document. The active document has to have been opened from the system for editing.

- *Settings:*

  The OfficeLink administration manual includes the settings options in the option dialog. Contact your system administrator if you have questions.

6.2 Editing documents

6.2.1 Searching for documents in the system

The *[Load]* button opens the document search mask in the system. The specified search condition automatically limits the search to Visio documents. The result list opens with a preview window after entering additional search criteria and running the search. After selecting one or multiple documents, they are opened for editing.

6.2.2 Save in the system

The *[Save]* button saves the changes to the active document directly in the system.

6.2.3 Create in the system

The *[Create]* button carries out a document creation in the system for the active document. This process initially opens the mask for creating documents. After filling out and confirming the create mask, the document is committed to the system and finally re-opened from the system for editing. The original document is closed after the create is successful. In the case of an error, for example due to deficient authorizations in the system, the document remains open.

Note: Since multiple file formats from the system are supported with *Visio 2013*, the save format is determined there first, before the system document management create mask is opened: *Use as default format:* Enabling this check box defines the selected format as the default save format. The selection dialog is no longer shown when creating documents in the future and the default save format is used instead. The default save format can be changed or removed in the settings dialog at any time. *Visio Open XML file format and macros:* No macros can be saved in the .vsdx Open XML file format of Visio. When attempting to save documents with macros in this format, a corresponding message is displayed.

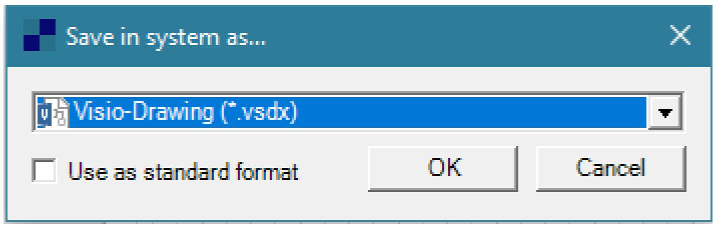


Fig. 6.2: Visio: Save format selection dialog

6.2.4 Create index in the system

The *[Create index]* button creates a new index in the system for the active document. This process initially opens the mask for creating indices. After filling out and confirming the index mask, the current changes are backed up in the new document index in the system.

6.2.5 Change document status

The *[Change status]* button carries out a status change operation in the system for the active document.

Note: Since the user potentially loses the rights to re-save the document in the system by changing the status, the link saves the document automatically before calling up the status change operation. If necessary, a corresponding message is displayed here that you have to confirm.

6.2.6 Opening master data change dialog in the system

The *[Master data]* button opens the dialog in the system for changing the master data for the active document. This button is divided into two parts. The bottom part can be used in the system to open an information dialog for the active document instead of the modify dialog.

6.3 Synchronizing data

6.3.1 Metadata takeover from the system

By pressing the *[Transfer [read]]* button, the values of all system variables with read access are updated with the corresponding values of the associated system attributes. This overwrites system variables with read access changed by the user in the document. If an error exists in the expression of a variable, the referenced object does not exist or the relationship is not unique, the value is set to a space character.

6.3.2 Metadata synchronization with the system

By pressing the *[Transfer [write]]* button, the values of all variables with write access are checked for changes. If values have been changed in the document, the affected objects are opened in the system sequentially. The changes are shown in the modify dialogs and have to be confirmed for each object. After completing the data transfer to the system, a metadata acceptance is carried out for system variables with read access.

6.3.3 Create templates for data synchronization

This chapter describes creating document templates with system variables for data synchronization.

- Synchronization of document metadata (e.g. title, document No., etc.)
- Synchronization of 1:1 referenced objects (e.g. project name of the assigned project)

Format description for system variables

System variables in Visio documents have the following format: cdb.<Mode>.<Relation>.<Attribute>.<Cardinality>

Mode The mode indicates whether a variable may be read from the system exclusively or updated after changing in the document in the system. Valid values for the mode include:

- *r*

  Only data transfer from the system possible. The transfered value can be changed in the document, but not transferred to the system. The changed value in the document is replaced again by the system value when transferring the data again.

- *w*

  Only data transfer to the system possible. This mode can be used to ensure that a document field cannot be changed by a data transfer, though an update is possible in the system (useful for field calculated by the document). This mode cannot be used for 1:N relationships.

- *rw*

  Data transfer from the system and update in the system possible after changing in the document. This mode cannot be used for 1:N relationships.

Relationship Starting from the document, the relationship indicates the path for the data in the system represented by the variable. The following expressions can be used as a relationship:

- *this*

  The constant expression 'this' is used to specify the attributes of the document itself.

- *Name of a relationship configured in CIM DATABASE*

  Attributes of referenced objects can be transferred and synchronized via a relationship.

- *BY_ZNUM_ZIDX_FROM_<table_name>*

  Use an implicit relationship via the attributes z_nummer and z_index. To do so, the expression BY_ZNUM_ZIDX_FROM_<table_name> is specified. table_name indicates the database table used to reference an object implicitly via z_nummer and z_index. The specified database table must contain the attributes z_nummer and z_index and the combination of z_nummer and z_index must be unique.

Attribute The attribute specifies an attribute of a system object referenced by the relationship.

Subject characteristics as attributes of parts:

If a part relationship is specified as a relationship, the subject characteristics of the part can also be specified as an attribute. This can be carried out in a qualified manner via property identification or unqualified using the sequence number:

- *Qualified subject characteristics:*

  `_SML_<mm_mk>`

Visio: data synchronization - subject characteristics qualified

Subject characteristics with the identification 'L' of the associated part

```
``cdb.r.cdb_doc_to_part._SML_L``
```

- *Unqualified subject characteristics:*

  `_SMLN<nummer> for characteristic identifications`

  `_SMLV<nummer> for characteristic values`

  Property identification and the associated value must have the same number for this.

Visio: data synchronization - subject characteristics unqualified

Identification and value of the first subject characteristic of the associated part

```
``cdb.r.cdb_doc_to_part._SMLN1``
``cdb.r.cdb_doc_to_part._SMLV1``
```

Important: SC attributes cannot be synchronized with write access!

Cardinality Specifies the cardinality of the relationship. If no cardinality has been specified, cardinality 1 is assumed. The cardinality should be specified explicitly when creating new document templates. The following values can be used for the cardinality:

- *1*

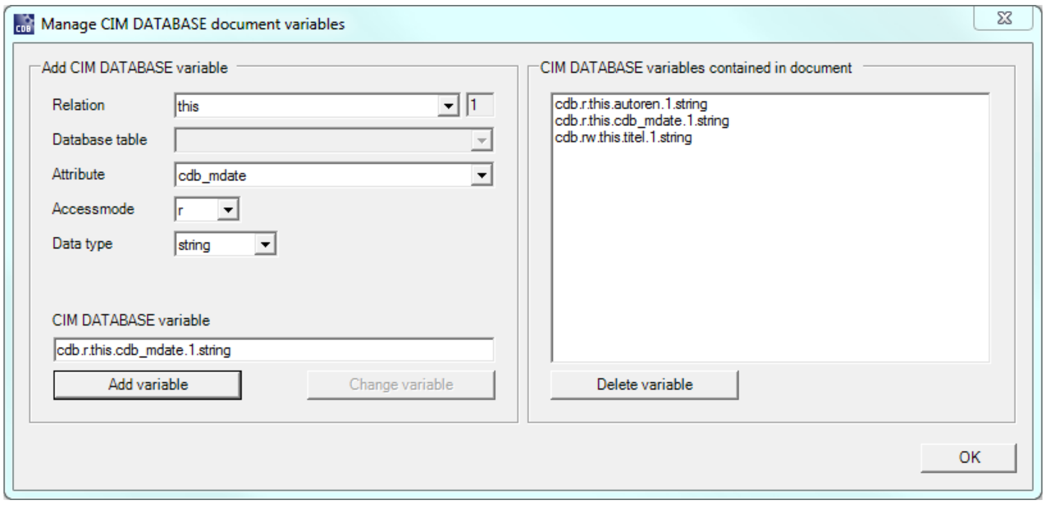
  For relationships of the type `this,BY_ZNUM_ZIDX_FROM_<table_name>` and for 1-digit homogeneous relationships.

Definition of system variables

System variables are defined using a wizard that calls up the OfficeLink toolbar using the *[Manage]* button. The button has to have been enabled beforehand in the settings dialog so that the button is available. The dialog provides the option of creating new variables in the system format on the left side (see *Format description for system variables* (page 61)). On the right side, all system variables contained in the document are displayed. By marking a variable, the corresponding text field is marked in the Visio document at the same time. The marked variables are removed from the document by pressing the *[Delete variable]* button.

- *Relationship:*

  Starting from the document, the relationship indicates the path for an attribute of one or more referenced system objects. The catalog provides all document relationships configured in the system, for which the document is the start object. For attributes of the document itself, you have to enter `this` in this field.


Manage CIM DATABASE document variables
Add CIM DATABASE variable
Relation
Database table
Attribute
Accessmode
Data type
Add variable
Change variable
CIM DATABASE variables contained in document
Delete variable
OK

Relation: this, 1
Attribute: cdb_mdate
Accessmode: r
Data type: string
CIM DATABASE variable: cdb.r.this.cdb_mdate.1.string

CIM DATABASE variables contained in document:

```
cdb.r.this.autoren.1.string
cdb.r.this.cdb_mdate.1.string
cdb.rw.this.titel.1.string
```

Fig. 6.3: Visio: data synchronization - dialog for creating system variables

To use an implicit relationship that is produced via the key attributes document number and index, you have to select `BY_ZNUM_ZIDX_FROM_` in this field and select a relation as a data source in the database table field.

- *Database table:*

  A database table has to be selected only if the special entry `BY_ZNUM_ZIDX_FROM_` for implicit relationships has been selected in the relationship field. In the selected relation, the combination of the attributes z_nummer and z_index has to be unique.

- *Attribute:*

  The attribute specifies an attribute of a system object referenced by the relationship.

- *Synchronization mode:*

  The mode indicates whether a variable may be read from the system exclusively or may be updated after changing in the document in the system.

  r: read only

  w: write only

  rw: read and write access in CIM DATABASE

- *CIM DATABASE variable:*

  In this field, the expression generated from the other fields is shown for the system variables to be created. The manual modification of this expression is only required for specifying subject characteristics for part relationships.

When adding the variables by pressing the *[Add variable]* button, the expression shown in the system variable field is reviewed and displays a corresponding error message if invalid. Otherwise, the variables are created as new system variables in the active document and initialized with the `<Attribut>` value as a placeholder.

Important: The system variables can be positioned only in simple Visio 'forms'. Furthermore, only one system variable can be defined per Visio form. An error message appears if there is an attempt to define a new variable without the simple form being enabled, or if the currently enabled form already contains a variable.

Modifying system variables

The text fields containing system variables can be modified by clicking in the field to be modified. Then the cursor is in the field and enables free editing.

Using system variables in the document

System variables should be identified for the user in the document to avoid accidental deleting or overwriting. The protection of document variables is possible only on condition. One option is to create forms.

Note: During the data synchronization of variables to the system, attribute changes are also shown if the field cannot be changed by the user interactively according to the mask configuration or would only be able to be changed using a list of options. You have to take this into account when using system variables with write access. If necessary, the validity of the values must be ensured through a corresponding check in the user exit.

Updating metadata automatically

The metadata of all system variables with read access are committed from CDB automatically when loading a document. The conditions for this are that the document:

1. Has been opened in the client in “Edit” mode (or directly from Office via “Load”)
2. Is a “Drawing” (i.e. has the property “vorlage_kz”)
3. Is not a template (i.e. “vorlage_kz=0”)

CHAPTER 7

Word

7.1 Introduction

- *Word interface:*

  The installation expands the Word application by adding the OfficeLink toolbar.

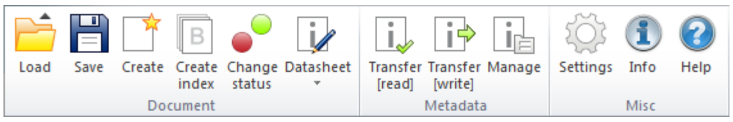


Fig. 7.1: Word: OfficeLink menu bar

- *Standard operations:*

  The system Microsoft Office integration provides Word with the option of searching for documents in the system, opening them for editing and then saving changes in the database. Moreover, it is possible to change the document status or properties, or create new documents in the system directly from Word.

- *Data synchronization:*

  The link gives you the ability to synchronize data with the system. A prerequisite for this is the use of prepared document templates which contain corresponding system variables (see *Create templates for data synchronization* (page 67)). Data synchronization is always carried out for the active document. The active document has to have been opened from the system for editing.

- *Settings:*

  The OfficeLink administration manual includes the settings options in the option dialog. Contact your system administrator if you have questions.

7.2 Editing documents

7.2.1 Searching for documents in the system

The *[Load]* button opens the document search mask in the system. The specified search condition automatically limits the search to Word documents. The result list opens with a preview window after entering additional search criteria and running the search. After selecting one or multiple documents, they are opened for editing.

7.2.2 Save in the system

The *[Save]* button saves the changes to the active document directly in the system.

7.2.3 Create in the system

The *[Create]* button carries out a document creation in the system for the active document. This process initially opens the mask for creating documents. After filling out and confirming the create mask, the document is committed to the system and finally re-opened from the system for editing. The original document is closed after the create is successful. In the case of an error, for example due to deficient authorizations in the system, the document remains open.

Note: Since multiple file formats from the system are supported with Microsoft Word, the save format is determined there first, before the the system document management create mask is opened: *Use as default format:* Enabling this check box defines the selected format as the default save format. The selection dialog is no longer shown when creating documents in the future and the default save format is used instead. The default save format can be changed or removed in the settings dialog at any time. *Word Open XML file format and macros:* No macros can be saved in the .docx Open XML file format of Word. When attempting to save documents with macros in this format, a corresponding message is displayed.

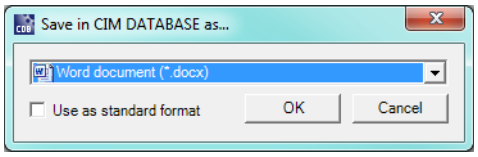


Fig. 7.2: Microsoft Word: Save format selection dialog

7.2.4 Create index in the system

The *[Create index]* button creates a new index in the system for the active document. This process initially opens the mask for creating indices. After filling out and confirming the index mask, the current changes are backed up in the new document index in the system.

7.2.5 Change document status

The *[Change status]* button carries out a status change operation in the system for the active document.

Note: Since the user potentially loses the rights to re-save the document in the system by changing the status, the link saves the document automatically before calling up the status change operation. If necessary, a corresponding message is displayed here that you have to confirm.

7.2.6 Opening master data change dialog in the system

The *[Master data]* button opens the dialog in the system for changing the master data for the active document. This button is divided into two parts. The bottom part can be used in the system to open an information dialog for the active document instead of the modify dialog.

7.3 Synchronizing data

7.3.1 Metadata takeover from the system

By pressing the *[Transfer [read]]* button, the values of all system variables with read access are updated with the corresponding values of the associated system attributes. This overwrites system variables with read access changed by the user in the document. If an error exists in the expression of a variable, the referenced object does not exist or the relationship is not unique, the value is set to a space character.

7.3.2 Metadata synchronization with the system

By pressing the *[Transfer [write]]* button, the values of all variables with write access are checked for changes. If values have been changed in the document, the affected objects are opened in the system sequentially. The changes are shown in the modify dialogs and have to be confirmed for each object. After completing the data transfer to the system, a metadata acceptance is carried out for system variables with read access.

7.3.3 Create templates for data synchronization

This chapter describes creating document templates with system variables for data synchronization.

- Synchronization of document metadata (e.g. title, document No., etc.)
- Synchronization of 1:1 referenced objects (e.g. project name of the assigned project)
- Transfer 1:N referenced objects (e.g. list with open issues)

Format description for system variables

System variables in Word documents have the following format: cdb.<Mode>.<Relation>.<Attribute>.<Cardinality>

Mode The mode indicates whether a variable may be read from the system exclusively or updated after changing in the document in the system. Valid values for the mode include:

- *r*

  Only data transfer from the system possible. The transfered value can be changed in the document, but not transferred to the system. The changed value in the document is replaced again by the system value when transferring the data again.

- *w*

  Only data transfer to the system possible. This mode can be used to ensure that a document field cannot be changed by a data transfer, though an update is possible in the system (useful for field calculated by the document). This mode cannot be used for 1:N relationships.

- *rw*

  Data transfer from the system and update in the system possible after changing in the document. This mode cannot be used for 1:N relationships.

- **s*

  If a *s* is added to the above-mentioned mode values (i. e. *rs*, *ws* or *rws*), then the same rules apply as for the respective values without *s*, but the read/write actions run completely on the server side. The *Parameter* field can be used optionally. You might, however, you need administrative customizing to use server-side editing of variables, which is explained in the OfficeLink Administration Manual.

Relationship Starting from the document, the relationship indicates the path for the data in the system represented by the variable. The following expressions can be used as a relationship:

- *this*

  The constant expression ‘this’ is used to specify the attributes of the document itself.

- *Name of a relationship configured in the system*

  Attributes of referenced objects can be transferred and synchronized via a relationship.

- *BY_ZNUM_ZIDX_FROM_<table_name>*

  Use an implicit relationship via the attributes z_nummer and z_index. To do so, the expression BY_ZNUM_ZIDX_FROM_<table_name> is specified. table_name indicates the database table used to reference an object implicitly via z_nummer and z_index. The specified database table must contain the attributes z_nummer and z_index and the combination of z_nummer and z_index must be unique.

Attribute The attribute specifies an attribute of a system object referenced by the relationship.

Subject characteristics as attributes of parts:

If a part relationship is specified as a relationship, the subject characteristics of the part can also be specified as an attribute. This can be carried out in a qualified manner via property identification or unqualified using the sequence number:

- *Qualified subject characteristics:*

  `_SML_<mm_mk>`

Word: data synchronization - subject characteristics qualified

Subject characteristics with the identification 'L' of the associated part

```
``cdb.r.cdb_doc_to_part._SML_L``
```

- *Unqualified subject characteristics:*

  `_SMLN<nummer> for characteristic identifications`

  `_SMLV<nummer> for characteristic values`

  Property identification and the associated value must have the same number for this.

Word: data synchronization - subject characteristics unqualified

Identification and value of the first subject characteristic of the associated part

```
``cdb.r.cdb_doc_to_part._SMLN1``
``cdb.r.cdb_doc_to_part._SMLV1``
```

Important: SC attributes cannot be synchronized with write access!

Cardinality Specifies the cardinality of the relationship. If no cardinality has been specified, cardinality 1 is assumed. The cardinality should be specified explicitly when creating new document templates. The following values can be used for the cardinality:

- *1*

  For relationships of the type `this,BY_ZNUM_ZIDX_FROM_<table_name>` and for 1-digit homogeneous relationships.

- *N*

  For n-digit homogeneous relationships. Variables that use n-digit relationships must be positioned by column in the header of a list (see usage of system variables in the document). When synchronizing with the system, the list is adapted to the number of rows required for displaying the result. The first line after the header is used as a style sheet for this. The list is sorted based on the first column.

Word: data synchronization - system variables

Document number of the document:

```
"cdb.r.this.z_nummer"
```

Transfer and synchronization of the document remark:

```
"cdb.rw.this.z_bemerkung"
```

Transfer of the part number of the associated part:

```
"cdb.r.cdb_doc_to_part.teilenummer"
```

Transfer and synchronization of the part name of the associated part:

```
"cdb.rw.cdb_doc_to_part.benennung"
```

Transfer of the reason of the change history via implicit relation:

```
"cdb.r.BY_ZNUM_ZIDX_FROM_aenderung.begruend"
```

Parameter This field can be used to pass additional constant values specific to the respective variable when processing variables on the server side. These can be identifiers, for example, that allow you to link results from 1: N relationships with individual document fields instead of dynamic lists. For example, the parameter *PRIMARY* is added to the following variable, which tells a user exit that you want to have the original source name of the document's primary file.

```
cdb.rs.document2cdb_file.cdbf_original_name.N.string.PRIMARY
```

Definition of system variables

System variables are defined using a wizard that calls up the OfficeLink toolbar using the *[Manage]* button. The button has to have been enabled beforehand in the settings dialog so that the button is available. The dialog provides the option of creating new variables in the system format on the left side (see *Format description for system variables* (page 67)). On the right side, all system variables contained in the document are displayed. By marking a variable, the corresponding text field is marked in the Word document at the same time. The marked variables are removed from the document by pressing the *[Delete variable]* button.

- *Relationship:*

  Starting from the document, the relationship indicates the path for an attribute of one or more referenced system objects. The catalog provides all document relationships configured in the system, for which the document is the start object. For attributes of the document itself, you have to enter `this` in this field.

  To use an implicit relationship that is produced via the key attributes document number and index, you have to select `BY_ZNUM_ZIDX_FROM_` in this field and select a relation as a data source in the database table field.

- *Database table:*

  A database table has to be selected only if the special entry `BY_ZNUM_ZIDX_FROM_` for implicit relationships has been selected in the relationship field. In the selected relation, the combination of the attributes z_nummer and z_index has to be unique.

- *Attribute:*

  The attribute specifies an attribute of a system object referenced by the relationship.

- *Synchronization mode:*

  The mode indicates whether a variable may be read from the system exclusively or may be updated after changing in the document in the system.

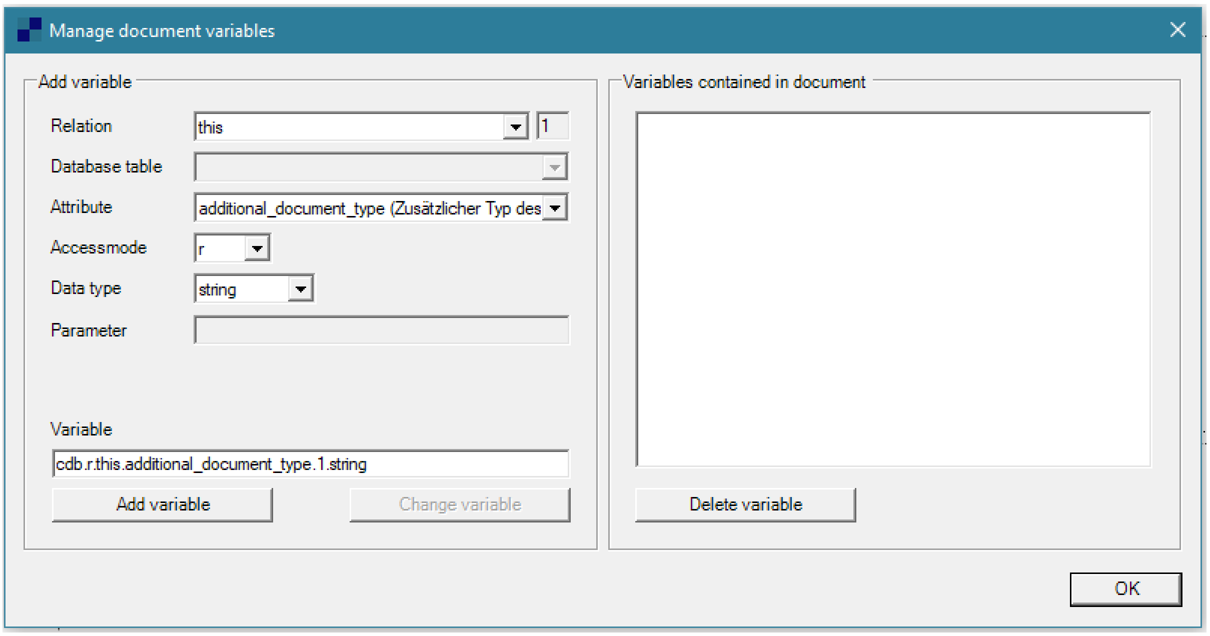

Manage document variables
Add variable
Relation
this
1
Database table
Attribute
additional_document_type (Zusätzlicher Typ des
Accessmode
r
Data type
string
Parameter
Variable
cdb.r.this.additional_document_type.1.string
Add variable
Change variable
Variables contained in document
Delete variable
OK

Fig. 7.3: Word: data synchronization - dialog for creating system variables

r: read only

w: write only

rw: read and write access in the system

rs: server-side read only

ws: server-side write only

rws: server-side read and write access in the system

- *Parameter:*

  Optional value used only for server-side editing of variables.

- *System variable:*

  In this field, the expression generated from the other fields is shown for the system variables to be created. The manual modification of this expression is only required for specifying subject characteristics for part relationships.

When adding the variables by pressing the *[Add variable]* button, the expression shown in the system variable field is reviewed and displays a corresponding error message if invalid. Otherwise, the variables are created as new system variables in the active document and initialized with the `<Attribut>` value as a placeholder. The system variables are represented by fields of the DocVariable type and can be freely positioned in the document after being inserted.

Modifying system variables

The text fields containing system variables can be modified by clicking in the field to be modified. Then the cursor is in the field and enables free editing. This mechanism only works if the field consists of at least two characters. If the field consists of only one character, a mouse click into the field and direct editing is no longer possible. For this purpose, the OfficeLink toolbar in Word contains the additional button submenu *[Change document variable value]*, which opens a special dialog for changing the value.

Using system variables in the document

System variables should be identified for the user in the document to avoid accidental deleting or overwriting. The protection of document variables is possible only on condition. One option is to create forms.

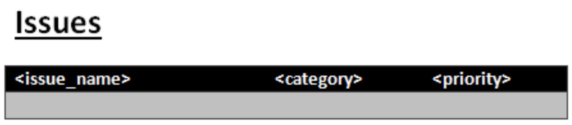
Note: During the data synchronization of variables to the system, attribute changes are also shown if the field cannot be changed by the user interactively according to the mask configuration or would only be able to be changed using a list of options. You have to take this into account when using system variables with write access. If necessary, the validity of the values must be ensured through a corresponding check in the user exit.

System variables that use n-digit relationships have to be positioned in Word by column in the header of a table. When synchronizing with the system, the table is adapted to the number of rows required for displaying the result. The first line after the header is used as a style sheet for this.

Important: If n-digit relationships are built into the header or footer of a document, and then the headers are changed to meaningful texts (for example, from "<z_nummer>" to "Document number"), these table headers must be locked using CTRL+F11. If these fields are not locked, the speaking text will be changed to its original placeholder name (e.g. back from "Document number" to "<z_nummer>") due to an idiosyncrasy in Word during a later PDF conversion. If necessary, the lock can be released using CTRL+SHIFT+F11.

Tables in Microsoft Office Word for n-digit relationships:

1. Insert tables with two rows and the desired number of columns
2. Enter column title into the first row and format it as desired
3. Insert the system variables as the column title using the wizard (see *Definition of system variables* (page 69)). All variables have to use the same relationship for this and be of the cardinality N.



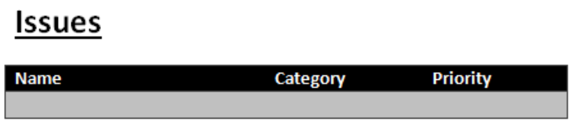
Issues

<issue_name>	<category>	<priority>

Fig. 7.4: Word: data synchronization - table for n-digit relationships [empty] (1)

4. Rename system variables so that their attributes can no longer be seen.

   Important: When changing the variable then it mustn't be fully selected, since it gets deleted that way. See the section "Modifying system variables".



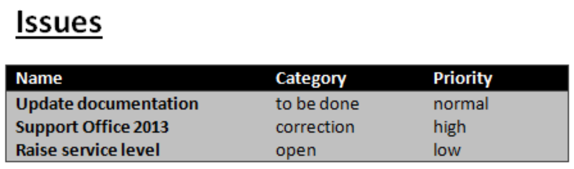
Issues

Name	Category	Priority

Fig. 7.5: Word: data synchronization - table for n-digit relationships [empty] (2)

5. Format the second row. This line is used as a master copy for pasting additional rows for the data synchronization.
6. Define a unique list title in the list property dialog. This is required when placing variables multiple times into different lists or when the list might exceed the page size.

When carrying out a data transfer, the table is expanded to the necessary number of rows and filled with the set of results for the relationship used.



Issues

Name	Category	Priority
Update documentation	to be done	normal
Support Office 2013	correction	high
Raise service level	open	low

Fig. 7.6: Word: data synchronization - table for n-digit relationships [filled]

Formatting fields

Especially with date values it is sometimes preferred, that the values of updated fields are displayed in a special format. This can be reached as described below:

1. Press ALT+F9 in order to display field codes instead of their values
2. Replace the string * *MERGEFORMAT* with the desired formatting code (e.g. @ *"dd.MM.yyyy"*)
3. Press ALT+F9 again in order to display values instead of field codes

Updating metadata automatically

The metadata of all system variables with read access are committed from CDB automatically when loading a document. The conditions for this are that the document:

1. Has been opened in the client in "Edit" mode (or directly from Office via "Load")
2. Is a "Drawing" (i.e. has the property "vorlage_kz")
3. Is not a template (i.e. "vorlage_kz=0")

List of Figures

List of Tables