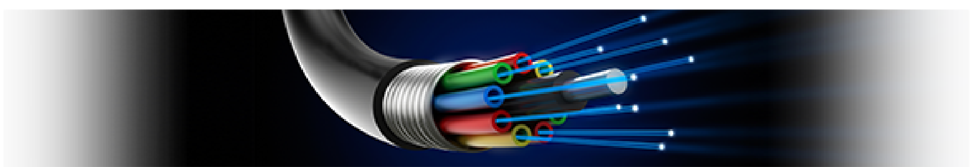
# New Features 2025-25.00



## Disclaimer

**Technical Inquiries**
Please contact our Support Department!
E-Mail: e3-support@zuken.com

**Note**
Zuken is not responsible for any errors, which may appear in this documentation. Liability, due to direct and indirect losses resulting from the delivery or use of this documentation, is excluded to the extent permitted by law. This documentation contains copyrighted information.
All rights, especially those pertaining to the duplication and distribution as well as the translation, are reserved. This documentation, whether wholly or in part, may not be reproduced in any form (photocopy, microfilm, etc.), or processed, duplicated or distributed using an electronic system without Zuken’s prior written consent.

**Contact**
Zuken E3 GmbH
Lämmerweg 55
D-89079 Ulm/Einsingen

Tel: 07305/9309-0
Fax: 07305/9309-99
Web: http://www.zuken.com
E-Mail: e3-info@Zuken.com

© Copyright 2025, Zuken E3 GmbH

# Table of Contents

# E³.series Release Notes

Date: 5/15/2024

This manual describes the following functions of version:

**2025 Build 25.00**

**!! Please Note when Switching to New Version !!**
Projects and databases are converted when opening them with this new ***E³.*series** Release. Please keep in mind that these projects can no longer be edited with an older version. That's why we recommend backing up projects beforehand.

**Note:**

- The latest version of the license server must be installed when using a FlexNet license server.
- Starting with Microsoft Office Version 2016, the TrueType font ***‚Arial Unicode MS'*** is no longer included in the standard delivery of Microsoft Office due to financial reasons.

  This font has been and is still in use for ***E³.*series**, ***E³*** template projects and ***E³*** example projects. You have most likely used this font to documents for ***E³*** projects. Presently, Zuken is not aware of any comparable functionality that is free-of-charge for ***‚Arial Unicode MS'***.

  To ensure that your new and older projects and the documents generated from them are displayed correctly on devices without older MS Office versions, we recommend you purchase and install the original font ***‚Arial Unicode MS Regular'*** on your devices. Furthermore, we recommend you change your templates to another font style so that new projects no longer use this font.

- **New Features**

## Update Information of Version 2025

### Windows Support

The following Windows systems are supported with ***E³.*****series** 2025:

**Client Operating System:**

- Windows 10 64-Bit (Version 22H2)
- Windows 11 64-Bit (Version 22H2)

**Server Operating System:**

- Windows Server 2019
- Windows Server 2022 21H2

For more information, please refer to the installation description.

### Installation

Before installing, we recommend you make a backup of your current installation (at least the data and databases).

The installation of ***E³.*****series** 2025 is started using

**`SETUP.EXE`**

which can be found on the DVD's main directory.

The Version can be installed in addition to an existing ***E³.*****series** Version.

**Note:** If internal IT policies prohibit the installation of Microsoft Visual C++ 2008 Redistributable, the following converters must **not** be installed in the installation process:

- Converter 2001-2003
- Converter 2004-2008
- Converter 2009-2011

The converters can be selected in the installation dialog along with the installation parameters.
For more information about the installation parameters, refer to the installation help of ***E³.*****series**.

The default directory is: `\Program Files\Zuken\E3.series_<version>`

Therefore, this version can also be used alongside existing versions of ***E³.*series**. However, note that you cannot use your existing database(s), or the existing projects, in both versions, as they are updated when opened in ***E³.*series** 2025, so copies should be used for***E³.*series** 2025.

## Databases

With this new version, a new default database is installed. Existing databases are not overwritten.

The new database contains new components and symbols.

## Projects

Existing projects are automatically converted upon opening with the new version. So long as the file has not been saved in the new version, it can still be opened with an older version. After saving a project with ***E³.*series** Version 2025, a project can no longer be opened with an earlier version.

## Multi-User Installation

Existing multi-user projects from Version 2023 can directly be opened with Version 2025. They are automatically converted upon opening. However, this cannot be undone, and the projects can no longer be opened with Version 2023.

That’s why we recommend installing Version 2025 multi-user projects in parallel to the existing version, so that it’s also possible to still access these projects with ***E³.*series** Version 2023.

To transfer the projects to Version 2025, these projects must be saved in Version 2023 as Single-User projects. They can then be stored in the multi-user environment of Version 2025.

In order to run the Multi-User databases in parallel

- a new SQL Server process must be installed for Microsoft SQL Server or another server must be installed.
- a new database user account for ORACLE must be used.

The E3 Server process must then access the new database or the new database user account.

## ORACLE

When setting up the ***E³***-MU-Server, a new name must be specified for the Multi-User Administrator.

Thus, a new database user account is created and both ***E³.*series** versions can work simultaneously in the multi-user environment.

## Microsoft SQL Server

First, a new instance of the SQL Server processes must be installed.

Enter a new instance name, which must be used with the set-up installation of the ***E³*** MU-Server.

The rest of the installation continues as described in the separate **MuSetup** installation description.

## Using an Existing Work Environment

Due to the new software component for the user interface, adjustments to the work environment (arrangement of windows, structure of toolbars, own programs added, hidden commands, assignment of keys to commands, ...) cannot be adopted by versions before 2020.

These assignments must be recreated and saved in the working environment.

## New Features in Version 2025 Build 25.00

The following functionality is new in Version 2025 Build 25.00:

# Overview - New Features in COM-Interface

The following new features are available in the COM Interface and described in the individual chapters:

# COM-Interface - Enhancements and Changed Behavior of Interfaces and Functions

This chapter provides an overview of which interfaces or functions have been enhanced in ***E³*.series** 2025 or had their behavior changed.

All enhancements that could be assigned to a customer request, which was documented by the ***E³*.series** Support in the form of a work item, are listed separately at the beginning of the chapter.

For detailed documentation, refer to the Help integrated in ***E³*.series**:

## Customer Request: Enhancement to *e3Dimension* - *Create*

The behavior of the following function has been enhanced in the ***e3Dimension*** interface:

**return = e3Dimension.Create ( shtid, x1, y1, x2, y2, flags, distance, text, tx, ty )**

- The parameter **flags** supports the following new value for creating angular dimensions:
    - **0x0080** (Decimal value **128**): If the parameter has this value, an angular dimension is created between the two objects whose coordinates are defined with the parameters **x1|y1** and **x2|y2**.
      The angular dimension can only be created if the objects are not parallel to each other.

**References:** `Designer-16325` and `Designer-34952`

## Customer Request: Enhancement to *e3Symbol* - *GetDisplayLength* and *SetDisplayLength*

The behavior of the following functions has been enhanced in the ***e3Symbol*** interface:

**return = e3Symbol.GetDisplayLength ( )**

**return = e3Symbol.SetDisplayLength ( length )**

- The functions are now also compatible with symbols for **shields**, **twisted pairs** and **bundles**.

**Reference:** `Designer-28605`

---

### Customer Request: Enhancement to *e3Symbol - PlacePins*

The behavior of the following function has been enhanced in the ***e3Symbol*** interface:

**return = e3Symbol.PlacePins ( pinarray, symname, vers, shti, x, y, rot, scale )**

- The function now also supports feed-through connectors and pins of feed-through connectors.
  To place a **feed-through connector**, transfer the device ID to **pinarray**.
  To place **pins of a feed-through connector**, transfer the device pin ID to **pinarray**.

Note the following restrictions when placing feed-through connectors with **e3Symbol.PlacePins ()**:

- The symbol name is not evaluated with **symname**.
- The version is not evaluated with **vers**.
- The characteristics of feed-through connectors can be exchanged with **e3Symbol.SetCharacteristic ()**.

**Reference:** `Designer-42167`

---

### Enhancement to *e3Application*, *e3Clone* and *e3DbeApplication - Include*

The behavior of the following functions has been enhanced in the ***e3Application***, **e3Clone** and **e3DbeApplication** interfaces:

**return = e3Application.Include ( text )**

**return = e3Clone.Include ( text )**

**return = e3DbeApplication.Include ( text )**

- The function has the following new return value:
  - **return -5**: Failed calling the function - the function is used recursively.

---

### Enhancement to *e3Component* and *e3Connection - AddAttributeValue*

The behavior of the following functions has been enhanced in the ***e3Component*** and ***e3Connection*** interfaces:

**return = e3Component.AddAttributeValue ( name, value )**

**return = e3Connection.AddAttributeValue ( name, value )**

- If the attribute **.ISOLATE** is successfully added with the function, the return value of the function outputs the ID of the **.ISOLATE** attribute at pin 1.

---

## Enhancement to *e3DbeModel* and *e3DbeSymbol* - *ImportDXF*

The behavior of the following functions has been enhanced in the ***e3DbeModel*** and ***e3DbeSymbol*** interfaces:

**return = e3DbeModel.ImportDXF ( filename, scale, x, y, rot, font, flags )**

**return = e3DbeSymbol.ImportDXF ( filename, scale, x, y, rot, font, flags )**

- If the functions are used to place drawings from DXF files that are larger than the sheet format actually allows, the sheet format is automatically adjusted so that the content can be placed.
- The function has the following new return value:
  - **return -7**: Failed calling the function - the DXF file could not be read because, for example, the file format is invalid.

---

The behavior of the following function has been enhanced in the **e3DbeModel** interface:

**return = e3DbeModel.ImportDXF ( filename, scale, x, y, rot, font, flags )**

- The parameter **flags** supports the following new values:
  - **0x0020** (Decimal value **32**): If the parameter has this value, information for the production of drill holes, threaded holes and cutouts is imported from the DXF file.
    This production information includes the drill hole definition and the contour type.
    Polygons, lines, circles and arcs that have either the **ACI color value 6** or the color value defined in the `DXFImport.cfg` file are recognized as drill holes, threaded holes and cutouts.
    If cutouts have already been defined for models before the DXF file is imported, no production information is imported.
  - **0x0040** (Decimal value **64**): If the parameter is also used with the value **0x0020**, only the production information for drill holes, threaded holes and cutouts is imported.
    If the parameter is also used with the value **0x0080**, only the production information is imported from contours.
    If the parameter is used without **0x0020** or **0x0080**, the parameter has no influence on the import.
  - **0x0080** (Decimal value **128**): If the parameter has this value, information for production regarding space requirements is imported from the DXF file.
    This production information includes, among other things, the dimensions of objects.
    Polygons and lines that have either the **ACI color value 4** or the color value defined in the `DXFImport.cfg` file are detected for determining the space requirement.
    If the import file contains different information on space requirements, no production information is imported.
  - The function has the following new return value:
    - **return -6**: Failed calling the function - production information could not be imported in its entirety.

---

## Enhancement to *e3DbeModel*, *e3DbeSymbol* and *e3Sheet* - *GetDXFSize*

The behavior of the following functions has been enhanced in the ***e3DbeModel***, ***e3DbeSymbol*** and ***e3Sheet*** interfaces:

**return = e3DbeModel.GetDXFSize ( filename, font, dx, dy, scale )**

**return = e3DbeSymbol.GetDXFSize ( filename, font, dx, dy, scale )**

**return = e3Sheet.GetDXFSize ( filename, font, dx, dy, scale )**

- The function can now also be used to query the size of the contents of DGN files.
- The function has the following new return value:
  - **return -4**: Failed calling the function - the DXF file could not be read because, for example the file format is invalid.

---

## Enhancement to *e3Device - AddAttibuteValue*

The behavior of the following function has been enhanced in the ***e3Device*** interface:

**return = e3Device.AddAttibuteValue ( name, value )**

- If the attribute **.ISOLATE** is successfully added with the function, the return value of the function outputs the ID of the **.ISOLATE** attribute at pin 1.

---

## Enhancement to *e3Job - ExportPDF* and *ExportPDFEx*

The behavior of the following function has been enhanced in the ***e3Job*** interface:

**return = e3Job.ExportPDF ( file, shtids, options, password )**

- The function only returns the following values:
  - **return 1**: Successfully called the function - the selected sheets were exported to PDF format.
  - **return 0**: Failed calling the function - the export failed.
- For more extensive functionality, use the function **e3Job.ExportPDFEx**.

---

The behavior of the following functions has been enhanced in the ***e3Job*** interface:

**return = e3Job.ExportPDF ( file, shtids, options, password )**

**return = e3Job.ExportPDFEx ( file, shtids, options, itemListType, items, alternativeColour, imageBrightness, password )**

- The parameter **options** supports the following new value for exporting sheets as 2D projections:
  - **0x80000** (Decimal value **524288**): If the parameter has this value, 2D projections of the desired sheets are exported.
    No matter the display mode set, the contents of the sheets are then exported as ***Projection on 2D sheet***.

---

The behavior of the following function has been enhanced in the ***e3Job*** interface:

**return = e3Job.ExportPDFEx ( file, shtids, options, itemListType, items, alternativeColour, imageBrightness, password )**

- The parameter **itemListType** supports the following new values for specifying which objects are exported:

    - **2**: All objects **not** specified in **items** are output.
    - **3**: All objects specified in **items** are output.
- The function has the following new return values:
    - **return -13**: Failed calling the function - document authorizations are not supported in the PDF/A standard.
    - **return -14**: Failed calling the function - external documents are not supported in the PDF/A standard.
    - **return -15**: Failed calling the function - 3D export is not supported in the PDF/A standard.
    - **return -16**: Failed calling the function - 3D export is not supported as of Acrobat Version 9.
    - **return -17**: Failed calling the function - the parameter ***password*** is not available for the desired Acrobat version specified with the parameter **options**.
      Use Acrobat Version 7 or newer.
    - **return -18**: Failed calling the function - no sheets found for PDF export.

**Reference:** `Designer-43788`

## Enhancement to *e3Pin - AddEndAttributeValue*

The behavior of the following function has been enhanced in the ***e3Pin*** interface:

**return = e3Pin.AddEndAttributeValue( name, value )**

- If the attribute **.ISOLATE** is successfully added with the function, the return value of the function outputs the ID of the **.ISOLATE** attribute at the pin.

## Enhancement to *e3Sheet - AlignObjects*, *DisplayEx* and *Export*

The behavior of the following function has been enhanced in the ***e3Sheet*** interface:

**return = e3Sheet.AlignObjects ( reference, ids, mode )**

- The function can be used to align objects on panel sheets to their respective carrier objects. Compared to the functionality from previous versions of ***E³.*****series**, the alignment works for all carrier objects and not just mounting rails.
  The following enhancements have been made for this purpose:
    - IDs of all objects that are used as carriers in the panel can be specified for the parameter **reference**.
      To align individual objects to the corresponding carrier **reference**, enter the IDs of all objects to be aligned under **ids**.
      To align all objects, which are placed on a carrier **reference**, enter the ID of the carrier under **ids**.
    - The parameter **mode** supports the following new values:
        - **7**: All selected objects are aligned with the left edge on the vertical center line of the carrier.
        - **8**: All selected objects are aligned on the vertical center line of the carrier.
        - **9**: All selected objects are aligned with the right edge on the vertical center line of the carrier.

    - **10**: All selected objects are aligned with the top edge on the horizontal center line of the carrier.
    - **11**: All selected objects are aligned on the horizontal center line of the carrier.
    - **12**: All selected objects are aligned with the bottom edge on the horizontal center line of the carrier.
  - The function has the following new return value:
    - **return -7**: Failed calling the function - an object that is to be aligned or the position in the panel is locked.

---

The behavior of the following function has been enhanced in the ***e3Sheet*** interface:

**return = e3Sheet.DisplayEx ( flags )**

- The parameter **flags** supports the following new values:
  - **0x0002** (Decimal value **2**): If the parameter has this value, the sheet is opened in the display mode ***3D (Sheet)***.
  - **0x0004** (Decimal value **4**): If the parameter has this value, the sheet is opened in the display mode ***3D (World)***.
  - **0x0008** (Decimal value **8**): If the parameter has this value, the sheet is opened in the display mode ***Projection on 2D sheet***.
    **Note:** Sheets cannot be displayed in several display modes at the same time. The new values for **flags** are therefore mutually exclusive and cannot be used simultaneously.
- The function has the following new return values:
  - **return -2**: Failed calling the function - the parameter **flags** is made up of values that are mutually exclusive.
  - **return -3**: Failed calling the function - the selected sheet is not a panel sheet and cannot be displayed in the display mode ***3D (Sheet)***, ***3D (World)*** or ***Projection on 2D sheet***.
  - **return -4**: Failed calling the function - the selected sheet is hidden via variants/options and cannot be opened with the function.

---

The behavior of the following function has been enhanced in the ***e3Sheet*** interface:

**return = e3Sheet.Export ( format, version, file, flags )**

- The parameter **flags** supports the following new value for the export formats DWG and DXF:
  - **0x0004** (Decimal value **4**): If the parameter has this value and additionally the parameter **format** has the value **DWG** or **DXF**, only the contents that are marked on the sheet are exported.

---

## Enhancement to *e3Slot - GetPosition*

The behavior of the following function has been enhanced in the ***e3Slot*** interface:

**return = e3Slot.GetPosition ( point, x, y, z )**

- The function has the following new return value:
  - **return -5**: Failed calling the function - the position cannot be determined because the area is a polygon and not a rectangle.
    Use the function **e3Slot.GetAreaPolygon ( )** or **e3DbeSlot.GetAreaPolygon ( )** to query the position.

## Enhancement to *e3Symbol* and *e3Text* - *SendToBackground* and *SendToForeground*

The behavior of the following functions has been enhanced in the ***e3Symbol*** and ***e3Text*** interfaces:

**return = e3Symbol.SendToBackground ( )**

**return = e3Symbol.SendToForeground ( )**

**return = e3Text.SendToBackground ( )**

**return = e3Text.SendToForeground ( )**

- The function has the following new return value:
    - **return -6**: Failed calling the function - the selected graphic object is locked.

## Enhancement to *e3Tree* - *GetVisibleObjectTypes* and *SetVisibleObjectTypes*

The behavior of the following functions has been enhanced in the ***e3Tree*** interface:

**return = e3Tree.GetVisibleObjectTypes ( type_array )**

**return = e3Tree.SetVisibleObjectTypes ( type_array )**

- The parameter **type_array** supports the following new value for Index **0**:
    - **12** (**DISPLAYED_TYPE_ADDITIONAL_PARTS**):
      In der Funktion **e3Tree.GetVisibleObjectTypes ( type_array )** the value indicates that information on the additional parts used is displayed in the device tree.
      In der Funktion **e3Tree.SetVisibleObjectTypes ( type_array )** the value can be used to display information on the additional parts used in the device.

## Behavior Changed for Diverse Functions in *e3DbeGraph* and *e3DbeText*

Some functions in ***e3DbeGraph*** and ***e3DbeText*** interfaces have been modified to be compatible with model sheets as well. In this context, the parameter **symbolid** was changed to **id**.
The following functions are affected by the change:

- **e3DbeGraph.CreateArc ( id, x, y, radius, start, end )**
- **e3DbeGraph.CreateBlob ( id, filename )**
- **e3DbeGraph.CreateCircle ( id, x, y, radius )**
- **e3DbeGraph.CreateCloud ( id, pnts, x, y )**
- **e3DbeGraph.CreateCurve ( id, pnts, x, y )**
- **e3DbeGraph.CreateImage ( id, xpos, ypos, xsize, ysize, filename, embed )**
- **e3DbeGraph.CreateLine ( id, x1, y1, x2, y2 )**
- **e3DbeGraph.CreatePolygon ( id, pnts, x, y )**

- **e3DbeGraph.CreateRectangle ( id, x1, y1, x2, y2 )**
- **e3DbeText.Create ( id, texttype, x, y, textvalue )**

# COM-Interface - New Interfaces and Functions

This chapter provides an overview of which interfaces and functions that are new in ***E³.*series** 2025.

All enhancements that could be assigned to a customer request, which was documented by ***E³.*series** support in the form of a work item, are listed separately at the beginning of the chapter.

For detailed documentation, refer to the Help integrated in ***E³.*series**:

---

## Customer Request: New Functions in *e3DbeModelPin* and *e3Pin* - *GetRoutingOffset* and *SetRoutingOffset*

The following function has been added to the ***e3DbeModelPin*** interface for querying the offset values of model pins in the database:

**return = e3DbeModelPin.GetRoutingOffset ( x, y, z, flags )**

**Parameter:**

- **x** `[out]`: Outputs the offset value for the pin in the direction of the X-axis.
- **y** `[out]`: Outputs the offset value for the pin in the direction of the Y-axis.
- **z** `[out]`: Outputs the offset value for the pin in the direction of the Z-axis.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 1**: Successfully called the function - **x**, **y** and **z** output the offset values of the pin.
- **return -1**: Failed calling the function - no valid pin selected or no project opened.
- **return -2**: Failed calling the function - the value for **flags** is invalid.

---

The following function has been added to the ***e3DbeModelPin*** interface for setting the offset values for model pins in the database:

**return = e3DbeModelPin.SetRoutingOffset ( x, y, z, flags )**

**Parameter:**

- **x**`[in]`: Sets the offset value for the pin in the direction of X-axis.
- **y**`[in]`: Sets the offset value for the pin in the direction of Y-axis.
- **z**`[in]`: Sets the offset value for the pin in the direction of Z-axis.
- **flags**`[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 1**: Successfully called the function - **x**, **y** and **z** output the offset values of the pin.
- **return -1**: Failed calling the function - no valid pin selected or no project opened.
- **return -2**: Failed calling the function - the value for **flags** is invalid.

---

The following function has been added to the ***e3Pin*** interface for querying the offset values of model pins in the database:

**return = e3Pin.GetRoutingOffset ( x, y, z, flags )**

**Parameter:**

- **x**`[out]`: Outputs the offset value for the pin in the direction of the X-axis.
- **y**`[out]`: Outputs the offset value for the pin in the direction of the Y-axis.
- **z**`[out]`: Outputs the offset value for the pin in the direction of the Z-axis.
- **flags**`[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 1**: Successfully called the function - **x**, **y** and **z** output the offset values of the pin.
- **return -1**: Failed calling the function - no valid pin selected or no project opened.
- **return -2**: Failed calling the function - the selected object is not a **Pin** object.
- **return -3**: Failed calling the function - the selected pin is not a model pin.
- **return -4**: Failed calling the function - the value for **flags** is invalid.

---

The following function has been added to the ***e3Pin*** interface for setting the offset values for model pins in the database:

**return = e3Pin.SetRoutingOffset ( x, y, z, flags )**

**Parameter:**

- **x**`[in]`: Sets the offset value for the pin in the direction of X-axis.
- **y**`[in]`: Sets the offset value for the pin in the direction of Y-axis.
- **z**`[in]`: Sets the offset value for the pin in the direction of Z-axis.
- **flags**`[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 1**: Successfully called the function - **x**, **y** and **z**output the offset values of the pin.
- **return -1**: Failed calling the function - no valid pin selected or no project opened.
- **return -2**: Failed calling the function - the selected object is not a **Pin** object.

- **return -3**: Failed calling the function - the selected pin is not a model pin.
- **return -4**: Failed calling the function - the value for **flags** is invalid.
- **return -5**: Failed calling the function - the selected pin is locked.
- **return -6**: Failed calling the function - an error occurred.

**Reference:** `Designer-17483`

## Customer Request: New Functions in *e3DbeGraph* and *e3Graph* - *SendToBackground* and *SendToForeground*

The following function has been added to the ***e3DbeGraph*** and ***e3Graph*** interfaces for moving graphic objects to the background:

**return = e3DbeGraph.SendToBackground ( )**
**return = e3Graph.SendToBackground ( )**

**Return Values:**

- **return 0**: Successfully called the function - graphic object was moved to the background.
- **return -1**: Failed calling the function - no project opened.
- **return -2**: Failed calling the function - no graphic object ID set.
- **return -3**: Failed calling the function - graphic object cannot be selected.
- **return -4**: Failed calling the function - sheet cannot be selected.
- **return -5**: Failed calling the function - object cannot be moved to the background.
- **return -6**: Failed calling the function - no valid license available. Function cannot be executed with ***E³.*view** .
- **return -7**: Failed calling the function - graphic object is locked.

The following function has been added to the ***e3DbeGraph*** and ***e3Graph*** interfaces for moving graphic objects to the foreground:

**return = e3DbeGraph.SendToForeground ( )**
**return = e3Graph.SendToForeground ( )**

**Return Values:**

- **return 0**: Successfully called the function - graphic object was moved to the foreground.
- **return -1**: Failed calling the function - no project opened.
- **return -2**: Failed calling the function - no graphic object ID set.
- **return -3**: Failed calling the function - graphic object cannot be selected.
- **return -4**: Failed calling the function - sheet cannot be selected.
- **return -5**: Failed calling the function - object cannot be moved to the foreground.
- **return -6**: Failed calling the function - no valid license available. Function cannot be executed with ***E³.*view** .
- **return -7**: Failed calling the function - graphic object is locked.

**References:** `Designer-17874`, `Designer-22449` and `Designer-30778`

## New Function in *e3Device* - *PlaceModelViewAsGraphic*

The following function has been added to the ***e3Device*** interface for placing model views as graphics:

**return = e3Device.PlaceModelViewAsGraphic ( sheetId, x, y, rotation, modelView, flags )**

**Parameter:**

- **sheetId** `[in]`: Specifies the ID of the sheet, on which the graphic of the model view is placed.
- **x** `[in]`: Outputs the X-position of the sheet.
  The position is interpreted in the unit of measurement that is set for the project.
  The graphic object is aligned with the bottom left corner at the position.
- **y** `[in]`: Outputs the Y-position of the sheet.
  he position is interpreted in the unit of measurement that is set for the project.
  The graphic object is aligned with the bottom left corner at the position.
- **rotation** `[in]`: Specifies the rotation and mirroring values with which the graphic is placed.
  See the integrated help from ***E³*.series** for a complete description of the parameter.
- **modelView** `[in]`: Specifies which model view is placed as a graphic.
  The parameter supports the following values:
  - **0**: Places the view **Front**.
  - **1**: Places the view **Back**.
  - **2**: Places the view **Right**.
  - **3**: Places the view **Left**.
  - **4**: Places the view **Top**.
  - **5**: Places the view **Bottom**.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  Presently, the parameter does not support any values.

**Return Values:**

- **return > 0**: Successfully called the function - specifies the ID of the graphic that was placed.
- **return -1**: Failed calling the function - no project opened or no device selected.
- **return -2**: Failed calling the function - the value for **sheetId** is invalid.
- **return -3**: Failed calling the function - the selected sheet is locked.
- **return -4**: Failed calling the function - the value for **flags** is invalid.
- **return -5**: Failed calling the function - the value for **modelView** is invalid.
- **return -6**: Failed calling the function - no outline found.
- **return -7**: Failed calling the function - no symbol found.
- **return -8**: Failed calling the function - the selected device has no model view **modelView**.
- **return -9**: Failed calling the function - an error occurred.

## New Function in *e3Device* and *e3Pin* - *ChangeAssignedOptionExpression*

The following function has been added to the ***e3Device*** and ***e3Pin*** interfaces for changing the Boolean expression assigned to variants/options.
Variant instances are not supported by the functions:

**return = e3Device.ChangeAssignedOptionExpression ( oldval, newval, oldflags, newflags )**
**return = e3Pin.ChangeAssignedOptionExpression ( oldval, newval, oldflags, newflags )**

**Parameter:**

- **oldval** `[in]`: Outputs the current Boolean expression assigned to the variants/options of the object.
- **newval** `[in]`: Outputs the new Boolean expression assigned to the variants/options of the object to be changed.

- **oldflags**`[in][optional]`: An optional parameter that further restricts the currently defined availability of the Boolean expression for existing variants.
  The parameter supports the following values:
  - **0x000** (Decimal value **0**): The availability of the option for existing variants is not specified further.
  - **0x001** (Decimal value **1**): The object is further specified for the assignment of variants/options and has the property **Does not exist**.
  - **0x002** (Decimal value **2**): The object is further specified for the assignment of variants/options and has the property **Only exists**.
- **newflags**`[in][optional]`: An optional parameter that further restricts the currently defined availability of the Boolean expression for existing variants.
  The parameter supports the following values:
  - **0x000** (Decimal value **0**): The availability of the option for existing variants is not specified further.
  - **0x001** (Decimal value **1**): The object is further specified for the assignment of variants/options and is assigned the property **Does not exist**.
  - **0x002** (Decimal value **2**): The object is further specified for the assignment of variants/options and is given the property **Only exists**.

**Return Values:**

- **return 1**: Successfully called the function - the Boolean expression for the assignment of variants/options has been changed.
- **return -1**: Failed calling the function - no project opened or no object selected.
- **return -2**: Failed calling the function - no license available for ***E³.*****logic** or ***E³.*****series** is started in Viewer mode.
- **return -3**: Failed calling the function - the value for **oldval** or **newval**is invalid.
- **return -4**: Failed calling the function - the selected object is not supported by the function.
- **return -5**: Failed calling the function - the selected object is a view.
- **return -6**: Failed calling the function - the Boolean expression **oldval** was not found.
- **return -7**: Failed calling the function - the Boolean expression **oldval** is a variant instance.
- **return -8**: Failed calling the function - the Boolean expression **newval** is syntactically incorrect or contains incorrect arguments.
- **return -9**: Failed calling the function - the Boolean expression **newval** describes a variant instance.
- **return -10**: Failed calling the function - the value for **oldflags** or **newflags** is invalid.
- **return -11**: Failed calling the function - the selected object is locked and cannot be changed.

## New Function in *e3Group - GetGroupType*

The following function has been added to the ***e3Group*** interface for querying the group type of groups:

**return = e3Group.GetGroupType ( )**

**Return Values:**

- **return > 0**: Successfully called the function - the group type is output as a flag value.
  The parameter supports the following values:
  - **0x0001** (Decimal value **1**): Object group
  - **0x0002** (Decimal value **2**): Sheet group

- **0x0004** (Decimal value **4**): Grouped subcircuit that originates from an ***E³.*series** file with the file extension `*.e3p`
- **0x0008** (Decimal value **8**): Grouped subcircuit that originates from the ***E³.*series** database
- **0x0010** (Decimal value **16**): Group with all objects of a sheet that was exported to a drawing with the file extension `*.e3p`
- **0x0020** (Decimal value **32**): Automatic group with all objects of a sheet

- **return 0**: Failed calling the function - an error occurred.

## New Functions for the Multiuser Admin Tool *e3MultiuserClient - GetProjectInformation* and *ResetExclusiveMode*

The following function has been added to the ***e3MultiuserClient*** interface for querying project information:

**return = e3MultiuserClient.GetProjectInformation ( information, project )**

**Parameter:**

- **information** `[out]`: A multidimensional array used to output the project information for each queried project.
  The following information is stored in the array:
  - **Name**: Name of the project
  - **FolderPath**: Path of the project
  - **Description**: Description of the project
  - **ExclusiveMode**: Indicates whether the project is in exclusive mode.
    If the value is **1**, the project is in exclusive mode.
    If the value is **0**, the project is not in exclusive mode.
  - **ExclusiveUser**: If the project is opened in exclusive mode, **ExclusiveUser** specifies the user who opened the project.
  - **Created**: Date and time the project was created.
  - **CreatedBy**: User who created the project.
  - **LastAccessed**: Date and time the project was last opened.
  - **ExclusiveSince**: Date and time since when the project has been in exclusive mode.
- **project** `[in]` `[optional]`: An optional parameter that can be used to query the project information of a specific project.
  If the parameter is not used, the project information of all projects is queried.

**Return Values:**

- **return > 0**: Successfully called the function - the parameter **information** was filled with the corresponding number of projects and associated information.
- **return 0**: Failed calling the function - the parameter **information** was not filled with any information.
- **return -1**: Failed calling the function - the parameter **project**is invalid.

The following function has been added to the ***e3MultiuserClient*** interface of Multiuser Admin Tools for resetting the exclusive mode of projects:

**return = e3MultiuserClient.ResetExclusiveMode ( project )**

**Parameter:**

- **project** `[in]`: Specifies the name of the project whose exclusive mode is reset.

**Return Values:**

- **return 1**: Successfully called the function - the exclusive mode of the project has been reset.
- **return 0**: Inconclusive - the project is not in exclusive mode or exclusive mode cannot be reset.
- **return -1**: Failed calling the function - the parameter **project** is invalid.
- **return -2**: Failed calling the function - the project is currently opened by another user.

---

## New Functions in *e3Application* and *e3DbeApplication* - *GetComponentList*, *GetConfigFile*, *GetModelList* and *SetConfigFile*

The following function has been added to the ***e3Application*** and ***e3DbeApplication*** interfaces for querying all components in the active database:

**return = e3Application.GetComponentList ( list, additionalAttributes, flags )**
**return = e3DbeApplication.GetComponentList ( list, additionalAttributes, flags )**

**Parameter:**

- **list** `[out]`: A multidimensional array with which the component information of each queried component is output.
  The following information is stored in the **inner** array:
  - **list[0]**: Component name
  - **list[1]**: Component version
  - **list[2]**: Component type
  - **list[3]**:The name of the model assigned to the component.
  - **list[4]**: A flag value, which specifies whether the component is *current* or *old*.
    If the value is **0x0** (Decimal value **0**), the component is marked as ***current*** in the component properties.
    If the value is **0x1** (Decimal value **1**), the component is marked as ***old*** in the component properties.
  - **list[5]**: Number of attributes and attribute values contained in the array.
  - **list[6] and every third entry after [6]**: Attribute, which is specified with **additionalAttributes**.
  - **list[7] and every third entry after [7]**: Value of the attribute
  - **list[8] and every third entry after [8]**: Index of the attribute, which is specified with **additionalAttributes**.
- **additionalAttributes** `[in] [optional]`: An optional parameter that defines an array of component attributes whose values are output.
  For attributes that are defined for the component but have no value, an empty string is output.
  For attributes that are invalid or not defined for the component, nothing is output.
  Database fields, that means attributes that are in a column of the component database, must be specified in square brackets, for example, ***[Description]***.

  Such attributes are identified in the properties dialogs with CM .
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether components with the component property ***old*** are output.
  If the value is **0x0** (Decimal value **0**), components with the component properties ***old*** are not output.
  The value **0x0** is used if no value is defined for **flags**.

If the value is **0x1** (Decimal value **1**), components with the component properties ***old*** are output.

**Return Values:**

- **return > 0**: Successfully called the function - the parameter **list** was filled with the corresponding number of components and associated information.
- **return 0**: Failed calling the function - the parameter **list** was not filled with any information or the parameter **additionalAttributes** is invalid.

---

The following function has been added to the ***e3Application*** and ***e3DbeApplication*** interfaces for querying which configuration file is used for which action:

**return = e3Application.GetConfigFile ( processType, flags )**
**return = e3DbeApplication.GetConfigFile ( processType, flags )**

**Parameter:**

- **processType** `[in]`: Specifies the action for which the configuration file used is queried.
  The parameter supports the following values:
  - **1**: Queries the configuration file used for the **DXF Import**.
  - **2**: Queries the configuration file used for the **DXF Export**.
  - **3**: Queries the configuration file used for the **DGN Import**.
  - **4**: Queries the configuration file used for the **DGN Export**.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  Presently, the parameter does not support any values.

**Return Values:**

- **return "ConfigurationFile"**: Successfully called the function - the function specifies which configuration file is used.
- **return <EmptyString>**: Failed calling the function - a general error occurred, for example because the value for **processType** is not supported.

---

The following function has been added to the ***e3Application*** and ***e3DbeApplication*** interfaces for querying all models in the active database:

**return = e3Application.GetModelList ( list, additionalAttributes, flags )**
**return = e3DbeApplication.GetModelList ( list, additionalAttributes, flags )**

**Parameter:**

- **list** `[out]`: A multidimensional array with which the model information of each queried model is output.
  The following information is stored in the **inner** array:
  - **list[0]**: Model name
  - **list[1]**: Model type
  - **list[2]**:A flag value for the future enhancement of the function.
    Currently only the value **0x0** (Decimal value **0**) is output.
    The value has no meaning.
  - **list[3]**: Number of attributes and attribute values contained in the array.
  - **list[4] and every third entry after [4]**: Attribute, which is specified with **additionalAttributes**.

    - **list[5] and every third entry after [5]**: Value of the attribute
    - **list[6] and every third entry after [6]**: Index of the attribute, which is specified with **additionalAttributes**.
- **additionalAttributes** `[in] [optional]`: An optional parameter that defines an array of component attributes whose values are output.
  For attributes, which are defined for the component but do not have any, an empty string is output.
  For attributes that are invalid or not defined for the component, nothing is output.
  Database fields, that means attributes that are in a column of the component database, must be specified in square brackets, for example, ***[Description]***.
  Such attributes are identified in the properties dialogs with CM .
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  Presently, the parameter does not support any values.

**Return Values:**

- **return > 0**: Successfully called the function - the parameter **list** was filled with the corresponding number of models and associated information.
- **return 0**: Failed calling the function - the parameter **list** was not filled with any information.

---

The following function has been added to the ***e3Application*** and ***e3DbeApplication*** interfaces for specifying which configuration file is used for which action:

**return = e3Application.SetConfigFile ( processType, filepath, flags )**
**return = e3DbeApplication.SetConfigFile ( processType, filepath, flags )**

**Parameter:**

- **processType** `[in]`: Determines the action for which the configuration file is set.
  The parameter supports the following values:
    - **1**: Determines the configuration file used for the **DXF Import**.
    - **2**: Determines the configuration file used for the **DXF Export**.
    - **3**: Determines the configuration file used for the **DGN Import**.
    - **4**: Determines the configuration file used for the **DGN Export**.
- **filepath** `[in]`: The absolute path to the configuration file to be used.
  The set configuration file is only valid for the instance of ***E³.*****series**, in which it was set and is used until another configuration file is set.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  Presently, the parameter does not support any values.

**Return Values:**

- **return "Configuration File"**: Successfully called the function - the function specifies which configuration file was used before the change was made by the function.
- **return <Empty String>**: Failed calling the function - a general error has occurred, for example, because the specified configuration file does not exist or the value for **processType** is not supported.

## New Functions in *e3DbeApplication* - *CreateDbeSlotObject* and *RemoveUndoInformation*

The following function has been added to the ***e3DbeApplication*** interface for instantiating a slot object:

**return = e3DbeApplication.CreateDbeSlotObject ( )**

---

The following function has been added to the ***e3DbeApplication*** interface for deleting the history of the last actions that can be reverted using the ***Undo*** command in the database editor:

**return = e3DbeApplication.RemoveUndoInformation ( flags )**

**Parameter:**

- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 0**: Successfully called the function - the information from the last actions has been deleted from the working memory.
- **return -1**: Failed calling the function - no project opened in the database editor.
- **return -2**: Failed calling the function - the value for **flags** is invalid.

---

## New Functions in *e3DbeComponent* - *GetModelName* and *SetModelName*

The following function has been added to the ***e3DbeComponent*** interface for querying model names assigned to components in the database editor:

**return = e3DbeComponent.GetModelName ( flags )**

**Parameter:**

- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return "ModelName"**: Successfully called the function - the return value specifies the name of the model assigned to the component.
  If the model has a characteristic, this is written with a paragraph mark (*¶*) after the model name, for example, ***ModelName¶Characteristic***.
- **return <EmptyString>**: Inconclusive - no component ID is set or no model is assigned to the component.

---

The following function has been added to the ***e3DbeComponent*** interface for assigning component models in the database editor:

**return = e3DbeComponent.SetModelName ( modelName, flags )**

**Parameter:**

- **modelName** `[in]`: Specifies the name of the model that is assigned to the component.
  If the parameter remains empty, the currently assigned model is removed from the component.

- **flags** `[in]` `[optional]`: An optional parameter with which the function can be enhanced.
  Presently, the parameter does not support any values.

**Return Values:**

- **return 0**: Successfully called the function - the model has been assigned to the component, or the assigned model has been removed from the component.
  If the model is too large for the sheet that is currently open, the sheet format is automatically changed to ***MAXBOARD***.
- **return -1**: Failed calling the function - no project opened in the database editor.
- **return -2**: Failed calling the function - no component ID set.
- **return -3**: Failed calling the function - there is no model with the name **modelName** or the character length of the model name exceeds the maximum number of 52 characters.

---

## New Functions in *e3DbeModel - AddAttributeValue, DeleteModelView, DeleteStepModel, GetAttributeIds* and *GetSlotIds*

The following function has been added to the ***e3DbeModel*** interface for adding attributes and values in the database editor:

**return = e3DbeModel.AddAttributeValue ( name, value, flags )**

**Parameter:**

- **name** `[in]`: Specifies the name of the attribute to be added.
- **value** `[in]`: Specifies the value that is added to the **name** attribute.
- **flags** `[in]` `[optional]`: An optional parameter with which the function can be enhanced.
  Presently, the parameter does not support any values.

**Return Values:**

- **return > 0**: Successfully called the function - specifies the ID of the attribute that was added.
- **return -1**: Failed calling the function - no database editor opened or no model selected.
- **return -2**: Failed calling the function - the attribute value could not be added.
- **return -3**: Failed calling the function - the value for **flags** is invalid.
- **return -4**: Failed calling the function - the character length of the attribute value is too long.

---

The following function has been added to the ***e3DbeModel*** interface for deleting model views:

**return = e3DbeModel.DeleteModelView ( flags )**

**Parameter:**

- **flags** `[in]`: Specifies the model view that is deleted.
  The parameter supports the following values:
  - **0x01** (Decimal value **1**): Model view ***Front***
    **Note:** Currently, the ***Front*** model view cannot be deleted.
  - **0x02** (Decimal value **2**): Model view ***Back***
  - **0x04** (Decimal value **4**): Model view ***Right***
  - **0x08** (Decimal value **8**): Model view ***Left***
  - **0x10** (Decimal value **16**): Model view ***Top***

- **0x20** (Decimal value **32**): Model view ***Bottom***
  **Example:** If flags has the value **0x3E** (Decimal value **62**), all model views except the ***Front*** view are deleted.

**Return Values:**

- **return 1**: Successfully called the function - the selected model views were deleted.
- **return -1**: Failed calling the function - no project opened in the database editor.
- **return -2**: Failed calling the function - no model sheet selected.
- **return -3**: Failed calling the function - the value for **flags** is invalid.
- **return -4**: Failed calling the function - no model views selected that are to be deleted.
- **return -5**: Failed calling the function - at least one of the selected model views cannot be deleted.
- **return -6**: Failed calling the function - the model view ***Front*** cannot be deleted.

---

The following function has been added to the ***e3DbeModel*** interface for deleting STEP models of models:

**return = e3DbeModel.DeleteStepModel ( flags )**

**Parameter:**

- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 1**: Successfully called the function - the STEP model was deleted.
- **return -1**: Failed calling the function - no project opened in the database.
- **return -2**: Failed calling the function - no model sheet selected.
- **return -3**: Failed calling the function - the value for **flags** is invalid.
- **return -4**: Failed calling the function - the model has no STEP model.
- **return -5**: Failed calling the function - the STEP model cannot be deleted.

---

The following function has been added to the ***e3DbeModel*** interface for querying attribute values of models in the database editor:

**return = e3DbeModel.GetAttributeIds ( ids, attnam, flags )**

**Parameter:**

- **ids** `[out]`: Outputs an array with IDs of attributes that the model has.
- **attnam** `[in] [optional]`: Specifies an optional filter with which only attributes with a specific name are output.
  If **attnam** has no value, all attributes are output.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return ≥ 0**: Successfully called the function - the array **ids** contains the corresponding number of attribute IDs.
- **return -1**: Failed calling the function - no database editor opened or no model selected.
- **return -3**: Failed calling the function - the value for **flags** is invalid.

The following function has been added to the ***e3DbeModel*** interface for querying slot IDs:

**return = e3DbeModel.GetSlotIds ( ids, flags )**

**Parameter:**

- **ids** `[in]`: Specifies the slot IDs assigned to the model.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return ≥ 0**: Successfully called the function - the information of the last actions is deleted from the working memory.
- **return -1**: Failed calling the function - no project opened in the database editor.
- **return -2**: Failed calling the function - no sheet selected, on which a model is placed.
- **return -3**: Failed calling the function - the value for **flags** is invalid.

## New Interface *e3DbeSlot* for Editing Slots with the API

The following function has been added to the ***e3DbeSlot*** interface for assigning or changing slot names:

**return = e3DbeModel.SetSlotName ( name, flags )**

**Parameter:**

- **name** `[in]`: Defines the name of the slot.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 1**: Successfully called the function - the slot name was assigned or changed.
- **return -1**: Failed calling the function - no slot selected or no project opened.
- **return -2**: Failed calling the function - slots cannot be read.
- **return -3**: Failed calling the function - the slot name **name** is invalid because it contains invalid characters or is too long.
  Make sure that no invalid characters are used (**{**, **}**, **<**, **>**, **,** or **'**) and that the slot name does not exceed the maximum length of **252 characters**.
- **return -4**: Failed calling the function - the value for **flags** is invalid.

The ***e3DbeSlot*** interface has been added so that slots can be edited with the API.
The following functions have been adopted from other interfaces:

**return = e3DbeSlot.GetAreaPolygon ( xarr, yarr, zarr, flags )** with analog functionality to
**return = e3Slot.GetAreaPolygon ( xarr, yarr, zarr, flags )**

**return = e3DbeSlot.GetDirection ( x, y, z )** with functionality analogous to
**return = e3Slot.GetDirection ( x, y, z )**

**return = e3DbeSlot.GetFixType ( )** with functionality analogous to
**return = e3Slot.GetFixType ( )**

**return = e3DbeSlot.GetId ( )** with functionality analogous to
**return = e3Slot.GetId ( )**

**return = e3DbeSlot.GetMountType ( )** with functionality analogous to
**return = e3Slot.GetMountType ( )**

**return = e3DbeSlot.GetRotation ( angle )** with functionality analogous to
**return = e3Slot.GetRotation ( angle )**

**return = e3DbeSlot.GetSlotName ( flags )** with functionality analogous to
**return = e3Slot.GetSlotName ( flags )**

**return = e3DbeSlot.SetId ( id )** with functionality analogous to
**return = e3Slot.SetId ( id )**

---

The following function has been added to the ***e3DbeSlot*** interface and is also available in ***e3Slot*** with slightly modified functionality:

**return = e3DbeSlot.GetPosition ( point, x, y, z )** with similar functionality to
**return = e3Slot.GetPosition ( point, x, y, z )** but with the following changes:

**Parameter:**

- **point** `[in]`: Determines which position of the slot is to be queried.
  The parameter supports the following values:
  - **1**: Queries the first position of slots.
    The value can be used for slots of the type ***Point***, ***Line*** and ***Area***.
  - **2**: Queries the second position of slots.
    The value can be used for slots of the type ***Line*** and ***Area***.
- **x** `[out]`: The value of the parameter refers to the position relative to the model's origin.
- **y** `[out]`: The value of the parameter refers to the position relative to the model's origin.
- **z** `[out]`: The value of the parameter refers to the position relative to the model's origin.

**Return Values:**

- **return 1**: Successfully called the function - parameters **x**, **y** and **z** indicate the position of the slot.
- **return -1**: Failed calling the function - no slot selected or no project opened.
- **return -2**: Failed calling the function - slot data cannot be read.
- **return -3**: Failed calling the function - the value for **point** is invalid.
- **return -4**: Failed calling the function - the position cannot be determined because the area is a polygon and not a rectangle.
  In this case, use the function **e3Slot.GetAreaPolygon ( )** or **e3DbeSlot.GetAreaPolygon ( )**.

---

## New Functions in *e3Pin* - *GetColourDescriptionByInstallationLanguage* and *GetWiringDirection*

The following function has been added to the ***e3Pin*** interface for querying the wire color designation in the desired interface language of ***E³.*****series** :

**return = e3Pin.GetColourDescriptionByInstallationLanguage ( installationLanguage, flags )**

**Parameter:**

- **installationLanguage** `[in]`: Specifies the language code of the language in which the wire color is to be output, for example, **49** for the wire color in **German**.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return "Wire Color Designation"**: Successfully called the function - the function outputs the name of the wire color used.
  If a translation code is used for the wire color, for example, **&#253;** for **red**, only the translation code is output.
- **return "MULTIPLE_VALUES"**: Failed calling the function - the wire has different colors in different options.
- **return <EmptyString>**: Inconclusive - the wire has no color or a general error occurred.

---

The following function has been added to the ***e3Pin*** interface for querying the angles in the routing path of the wire:

**return = e3Pin.GetWiringDirection ( pinid, angle_1, angle_2, carrierid, flags )**

**Parameter:**

- **pin_item** `[in]`: Specifies the ID of the pin whose wire connection is being checked.
- **angle_1** `[out]`: Depending on whether an offset is defined for the routing, **angle_1** outputs the angle of the first or second wire segment in the wiring direction.
  If no routing offset is defined, **angle_1** outputs the angle of the first wire segment.
  If a routing offset is defined, **angle_1** outputs the angle of the second wire segment.
- **angle_2** `[out]`: Depending on whether an offset is defined for the routing, **angle_2** outputs the angle of the third or fourth wire segment in the wiring direction.
  If no routing offset is defined, **angle_2** outputs the angle of the third wire segment.
  If a routing offset is defined, **angle_2** outputs the angle of the fourth wire segment.
- **carrierid** `[in] [optional]`: An optional parameter that can be used to define the ID of a slot where the pin of the connected wire is located.
  If an ID is specified, the angle between the slot and the wire is specified as it is shown in the 2D view.
  If no ID is specified, the angle of the wire run is calculated according to its course on the panel sheet.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced. Presently, the parameter does not support any values.

**Return Values:**

- **return 3**: Successfully called the function - **angle_1** outputs the angle of the first or second wire segment, but **angle_2** cannot be determined because the wire runs in the Z direction and the X and Y coordinates remain the same.
- **return 2**: Successfully called the function - **angle_1** outputs the angle of the first or second wire segment, but **angle_2** cannot be determined because the wire has fewer than three wire segments.

- **return 1**: Successfully called the function - **angle_1** and **angle_2** output the angles of the wire.
- **return -1**: Failed calling the function - no project opened or no pin selected.
- **return -2**: Failed calling the function - pin object is not a wire.
- **return -3**: Failed calling the function - **pin_item** is not a valid pin.
- **return -4**: Failed calling the function - the value for **flags** is invalid.
- **return -5**: Failed calling the function - the selected wire is not active.
- **return -6**: Failed calling the function - the selected pin is not active.
- **return -7**: Failed calling the function - the object with the ID **carrierid** is not a slot.
- **return -8**: Failed calling the function - the selected slot is not active.
- **return -9**: Failed calling the function - the selected wire is not routed.
- **return -10**: Failed calling the function - the selected wire is not connected to the pin **pin_item**.
- **return -11**: Failed calling the function - a general error occurred.

## New Functions in *e3Slot* - *GetSlotName* and *GetAreaPolygon*

The following function has been added to the ***e3Slot*** interface for querying slot names:

**return = e3Slot.GetSlotName ( flags )**

**Parameter:**

- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  The value controls how translation codes are output.
  The parameter supports the following values:
  - **0x0000** (Decimal value **0**): Translation codes in slot names, for example **&#3;**, are translated and output as they are displayed in the user interface.
  - **0x0001** (Decimal value **1**): Translation codes in slot names, for example **&#3;**, are **not** translated but output in the form of the translation code.

**Return Values:**

- **return "Slot Name"**: Successfully called the function - the function outputs the name of the slot.
- **return <Empty String>**: Failed calling the function - a general error occurred.

---

The following function has been added to the ***e3Slot*** interface for querying the position of slots of the type ***Area*** that are defined as a polygon:

**return = e3Slot.GetAreaPolygon ( xarr, yarr, zarr, flags )**

**Parameter:**

- **xarr** `[out]`: An array in which the X-coordinates of all points of the polygon are specified.
- **yarr** `[out]`: An array in which the Y-coordinates of all points of the polygon are specified.
- **zarr** `[out]`: An array in which the Z-coordinates of all points of the polygon are specified.
- **flags** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  Presently, the parameter does not support any values.

**Return Values:**

- **return ≥ 0**: Successfully called the function - outputs the number of points of the polygon.
- **return -1**: Failed calling the function - no slot selected or no project opened.
- **return -2**: Failed calling the function - slot data cannot be read.

- **return -3**: Failed calling the function - the selected slot does not have the type ***Area***.
- **return -4**: Failed calling the function - the value for **flags** is invalid.

## General Changes

### Changes to Updating the Version Number of *E³.*series

The system for numbering the main version of ***E³.*series** has been changed.

On the one hand, this harmonizes the numbering of the major and minor versions while on the other hand, it eliminates a potential source of misunderstanding when communicating version numbers.

With the new versioning rule, the last two digits of the main version correspond to the first two digits of the build number.

**Example:**

The current version is the main version **2025**.
The last two digits are correspondingly **25**.

The current build has the number **25.00**.
The first two digits are correspondingly **25**.

### Set Color Definition for Contents in Different Windows

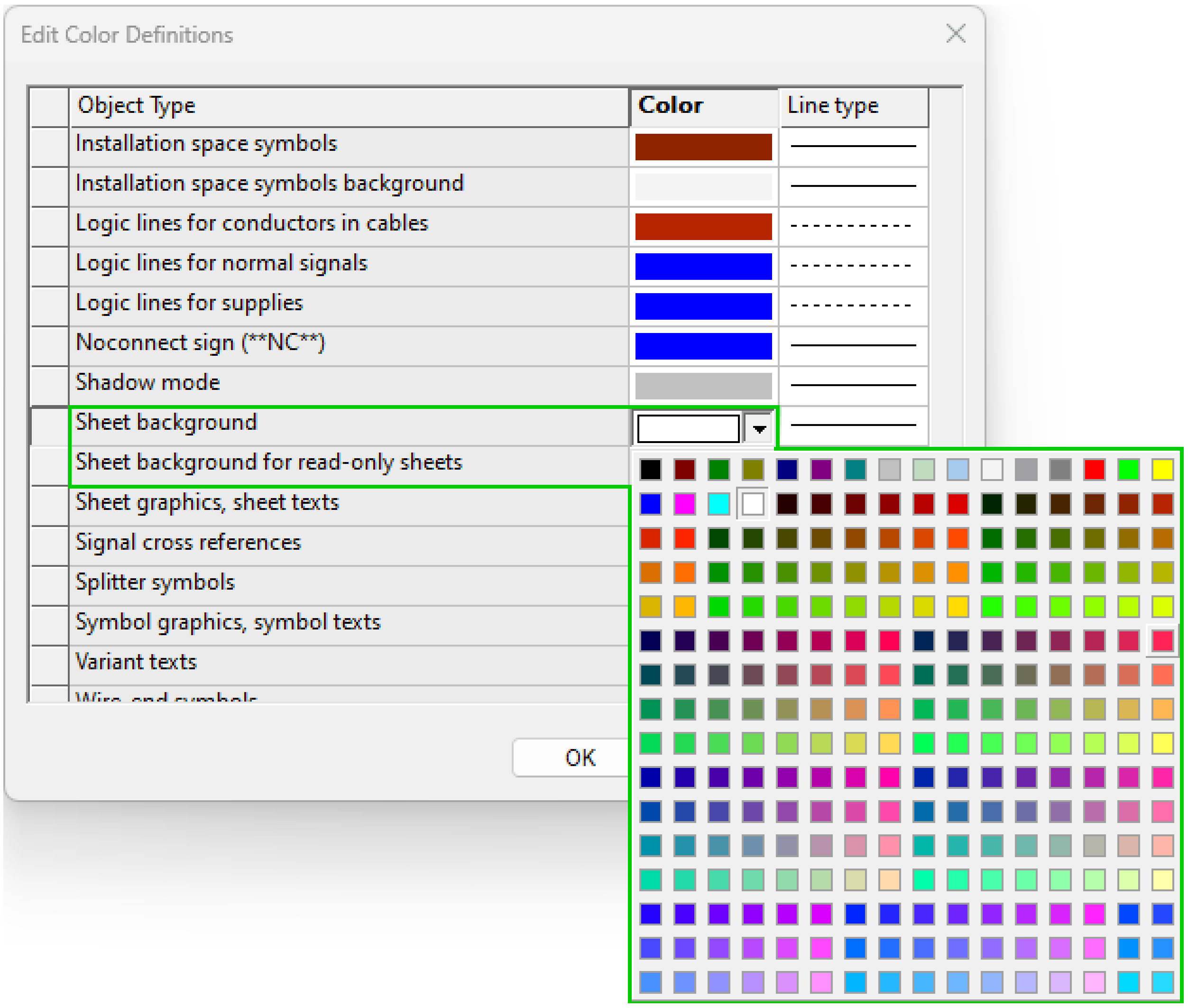
The rule for the display color of content that is displayed in different windows has been reviewed and revised.
In this context, two object types in the ***Edit Color Definitions*** table have also been renamed.

As a result, the behavior of the color definitions is now easier to understand.
The content in the different windows is also based on the sheet content that is displayed, provided it makes sense in terms of content.

To change the color definition of sheet contents, select ***Tools*** → ***Start Database Editor*** in the main menu.
The database editor opens.

Under ***Format*** → ***Color Definitions...*** select the colors that are used for the contents in different windows.
The ***Edit Color Definitions*** dialog opens.

Double-click in the ***Color*** cell of the ***Sheet background*** or ***Sheet background for read-only sheets*** object type to change the respective color:



Edit Color Definitions

| | Object Type | Color | Line type |
|---|---|---|---|
| | Installation space symbols | 🟥 | ——— |
| | Installation space symbols background | ⬜ | ——— |
| | Logic lines for conductors in cables | 🟥 | - - - - - - |
| | Logic lines for normal signals | 🟦 | - - - - - - |
| | Logic lines for supplies | 🟦 | - - - - - - |
| | Noconnect sign (**NC**) | 🟦 | ——— |
| | Shadow mode | ⬜ | ——— |
| | Sheet background | ⬜ | ——— |
| | Sheet background for read-only sheets | | |
| | Sheet graphics, sheet texts | | |
| | Signal cross references | | |
| | Splitter symbols | | |
| | Symbol graphics, symbol texts | | |
| | Variant texts | | |
| | Wire- and symbols | | |

**Note:** The object types ***Sheet background*** and ***Sheet background for read-only sheets*** are called ***Window background*** and ***Window background for read-only sheets*** in previous versions of ***E³.*****series** 2025.

Content in the different windows is displayed according to the following rules:

- The contents of the ***Preview*** window are displayed in the color that is set for the object type ***Sheet background*** or ***Sheet background for read-only sheets***.
  The colors can be inverted with ***Settings → Display → Invert display color***.
  **Note:** If the color definition is changed, the display in existing projects is only adjusted when the configuration is updated.

- The contents of the ***3D Preview*** and ***Preview*** windows in the Device Properties are always displayed with a white background, regardless of the ***Settings → Display → Invert display color*** or the definition under ***Edit Color Definitions***.

## Behavior Changed when Exchanging Components of Complete Connectors Connected to Mating Connectors

The behavior when exchanging components of complete connectors that are connected to a mating connector has been changed.

This means that complete connectors with more or less pins than the original connector can also be selected when exchanging components, making the result of the exchanged components more traceable.

If the component to be exchanged has as many or more pins as the original component, the exchange takes place without any restrictions.

If the new component has fewer pins, the component can only be exchanged if the following conditions are met:

- The connector and the connected mating connector are each complete connectors.
- The connected mating connector is defined as the active mating connector.
- Both sides must have the same number of pins.
- The new connector must have an active mating connector with the same number of pins.
- The pins, which are deleted when exchanging the component, must not have a plugged pin view.

**Reference:** `Designer-16357`

## Output Version Information in Files without Starting *E³.***series**

Version information from *E³.***series**, which is output with the `/v` or `/version` start parameter, can be output in JSON or text files without starting *E³.***series** .

This allows you to quickly make all essential information about the installation of *E³.***series**, for example, version number and available, compatible plug-ins, available in one file:

**`"<Installationsverzeichnis_von_E3.series\E3.series.exe" /v "Zielverzeichnis_Ausgabedatei\Dateiname.json"`** or
**`"<Installationsverzeichnis_von_E3.series\E3.series.exe" /v "Zielverzeichnis_Ausgabedatei\Dateiname.txt"`**

The version information is then stored in the target directory without starting *E³.***series** .

If the target directory is invalid, the version information is displayed in the console.

## Database Editor

### Application E³.CopyDatabaseEntries: Overwrite Only Outdated Elements in the Target Database

There is a new setting in the application **E³.CopyDatabaseEntries** that only overwrites the components, models and symbols in the target database that are newer in the source database.

This allows you to control that only those elements are overwritten during import that have an older date in the target database than in the source database.

The import behavior setting can be configured both via the user interface and with scripts:

1. To overwrite only obsolete database entries interactively via the **E³.CopyDatabaseEntries** user interface, first select the copy mode (***Export*** or ***Import***) and use ***Select source DB*** to set the database to be used as the source when copying.

Then drag and drop the desired elements into the right-hand window and activate the setting ***Edit → Only overwrite outdated elements***.

Finally, select a file or a target database to which the objects are to be copied.
While copying, only those objects are copied that have a more recent date in the source database than in the target database.
**Note:** If the option ***Overwrite existing items*** is also active, objects that exist in the source but not in the target are not copied.

2. To copy only obsolete database entries to the target database using a script, first write a script that describes the framework conditions for the copy process.
The script defines, for example, which source and target databases are used and which objects are copied.

In addition, use the following method to overwrite only outdated objects and execute the script:
`cde.OverwriteOutdatedItems ( True )`

**Note:** If the method `cde.OverwriteExistingItems ( True )` is also executed in the script, objects that are present in the source but not in the target are not copied.

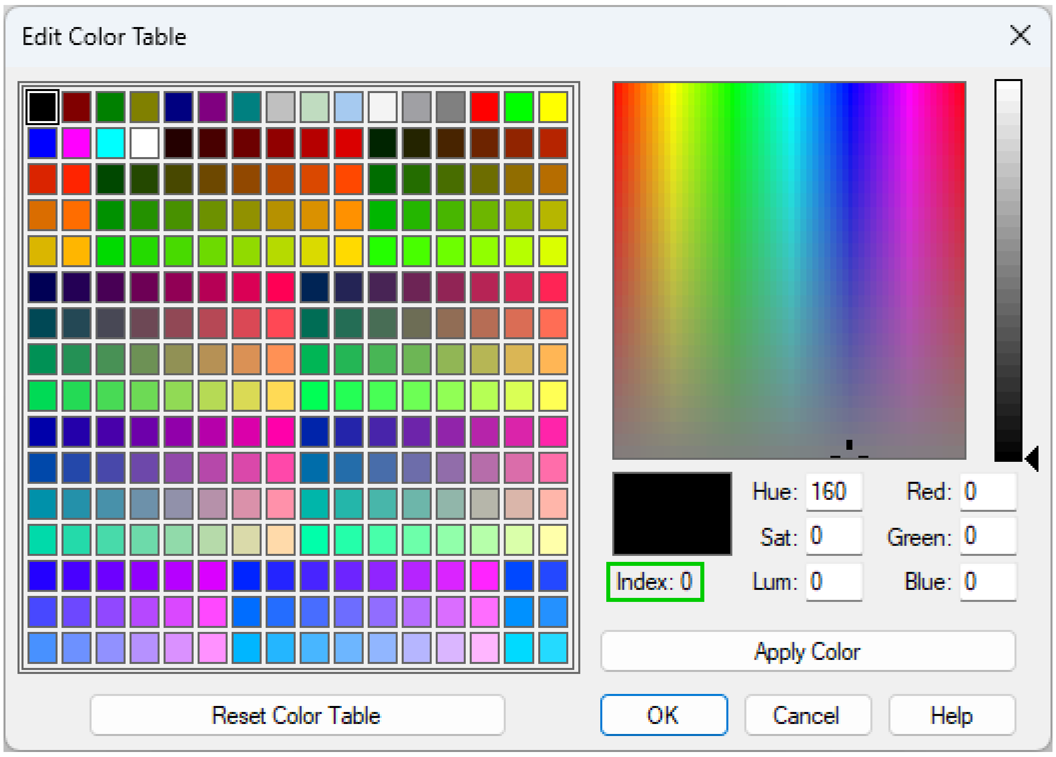
### Display the Color Index in the Color Table of the Database Editor

The index of each stored color is displayed in the color table, which can be edited in the database editor.

This allows you to see the unique index of the color you have selected.

To check the color index in the color table, select ***Tools → Start Database Editor*** in the main menu. The database editor opens.

Then open the color table under ***Format → Color Table...***, which is used in ***E³.*****series** .
The ***Edit Color Table*** dialog opens:



The color index of the currently selected color is displayed under ***Index***.
All changes you make to the color with the displayed index also affect the objects that use the corresponding color, for example, texts and graphics.

**Reference:** `Designer-43822`

## Define Mating Connector Group for Use as Active Mating Connector

In the database, connectors can be grouped into mating connector groups, which can then be used as active mating connectors.
Predefined combinations of connectors can be defined as individual components in the database in order to obtain a preselection of possible mating connectors when selecting the active mating connector.

This makes it easier, for example, to update and adapt connectors that are often used together as connector inserts for modular connectors.
In addition, these mating connector groups can be replaced more conveniently, saving time and reducing susceptibility to errors.

To define mating connector groups, select ***Tools → Start Database Editor*** in the main menu.
The database editor opens.

Right-click in the database editor component tree and select the context menu command ***New Component***.
The component wizard opens.

Then select the component type ***Mating Connector Group*** in the ***Component Wizard - Identification*** dialog and define the properties of the mating connector group on the following page. Confirm the dialog in each case with ***Next***.

On the last page ***Component Wizard - Finish***, select the option ***Edit component graphically*** and confirm the selection with ***Finish***.
The mating connector group is displayed in the component tree for further editing and can be edited on the worksheet.

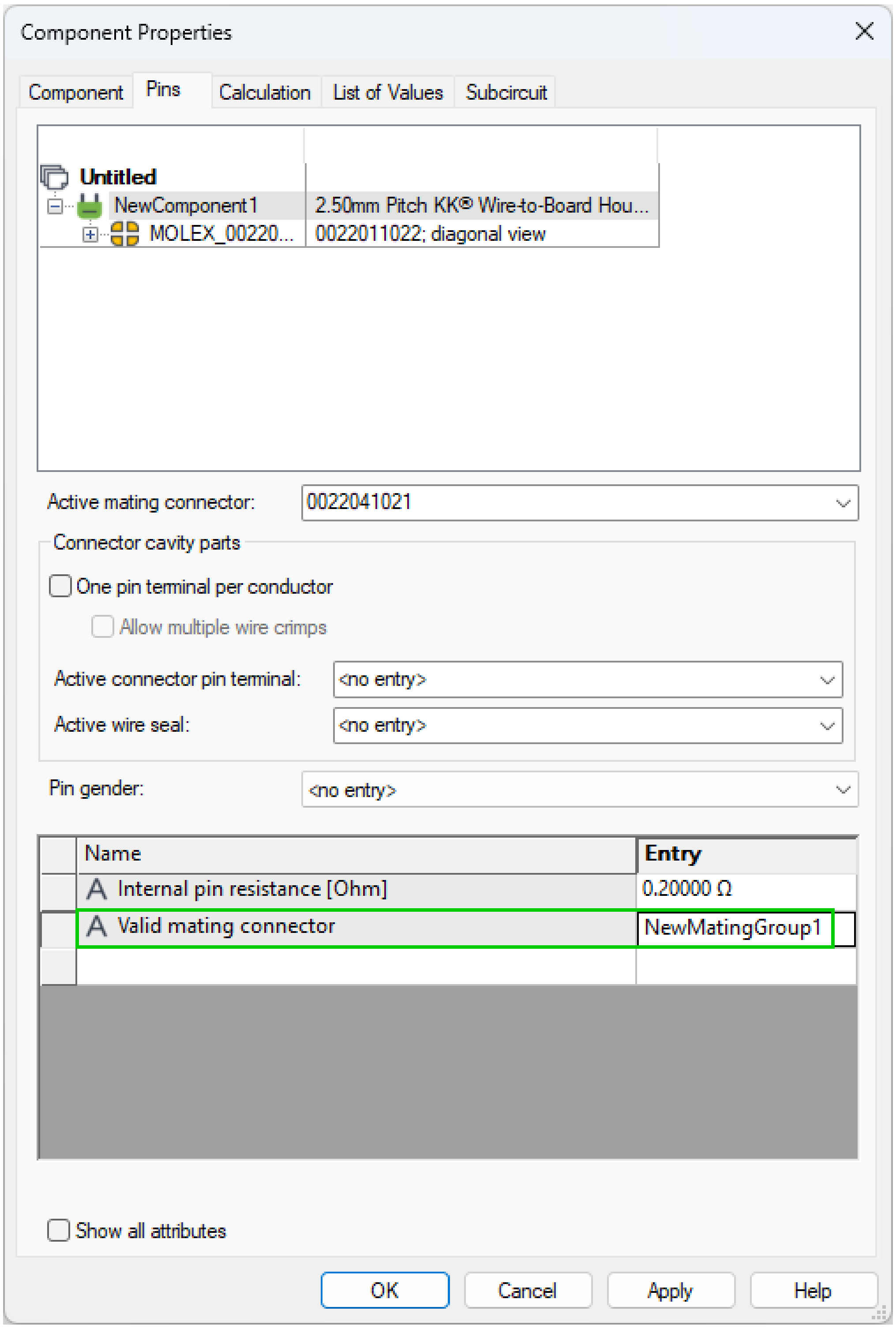
Place all connectors that are to belong to the mating connector group on the worksheet using drag & drop or the ***Place*** command and then right-click on a free space in the worksheet.

Select the context menu command ***Save to database***.
The mating connector group is saved in the database and the individual elements of the group can be used as active mating connectors.

---

To assign a mating connector of a predefined mating connector group to connectors as an active mating connector, open the component properties of the connector in the database editor.

Then add the name of the desired mating connector group in the ***Pins*** tab as a value for the attribute ***Valid mating connector*** and then select it from the drop-down list under ***Active mating connector***:



Component Properties

| | |
|---|---|
| Untitled | |
| NewComponent1 | 2.50mm Pitch KK® Wire-to-Board Hou... |
| MOLEX_00220... | 0022011022; diagonal view |

Active mating connector: 0022041021

Connector cavity parts

- [ ] One pin terminal per conductor
- [ ] Allow multiple wire crimps

Active connector pin terminal: <no entry>

Active wire seal: <no entry>

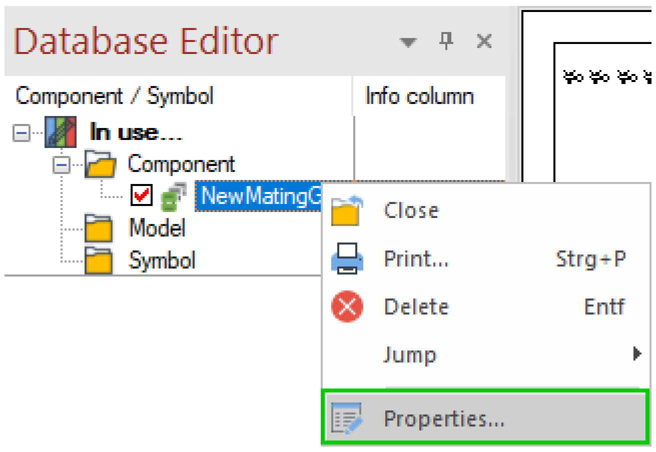
Pin gender: <no entry>

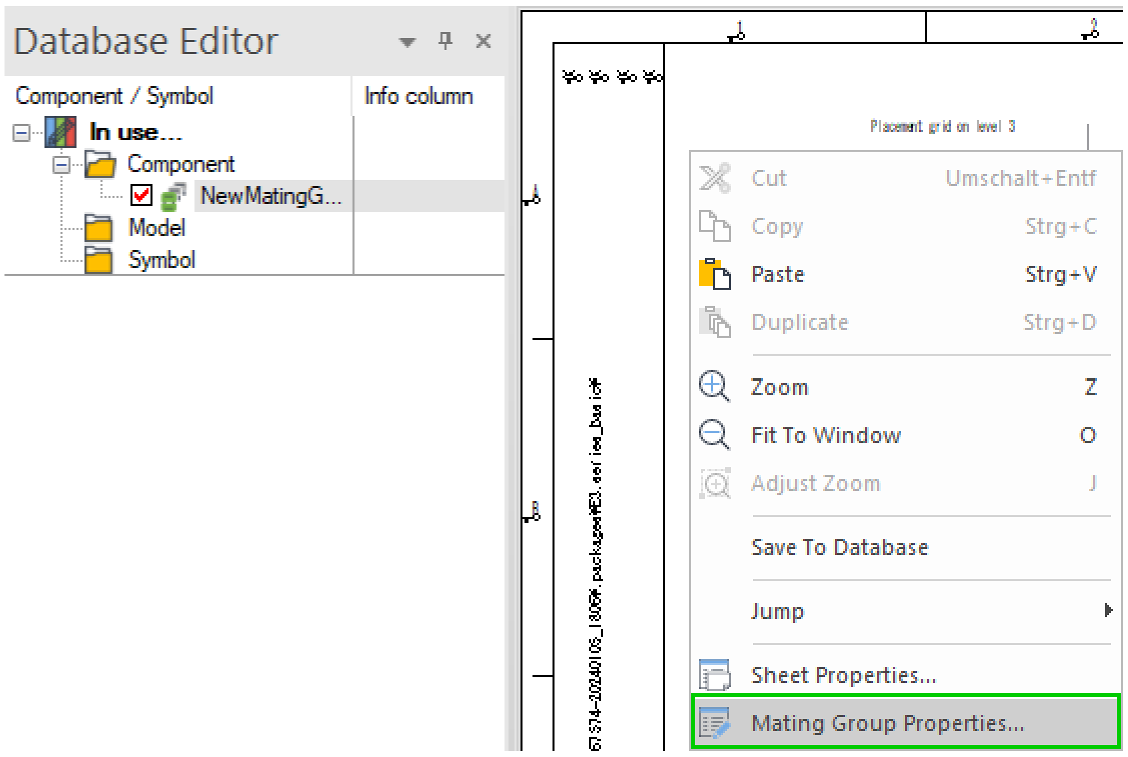
| Name | Entry |
|---|---|
| Internal pin resistance [Ohm] | 0.20000 Ω |
| Valid mating connector | NewMatingGroup1 |
| | |

- [ ] Show all attributes

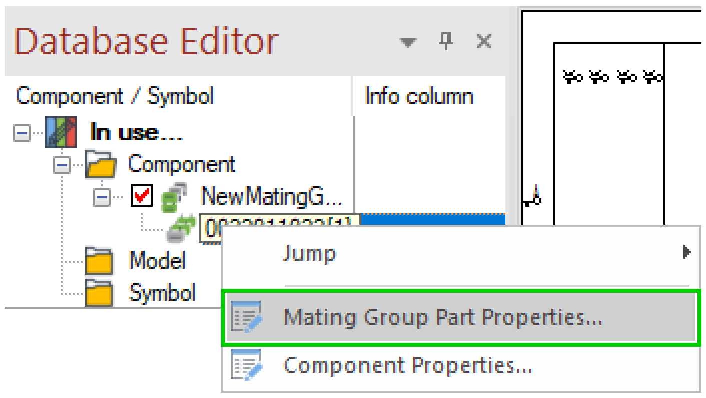
The individual elements of the mating connector group can then be selected as active mating connectors in the database editor and project.

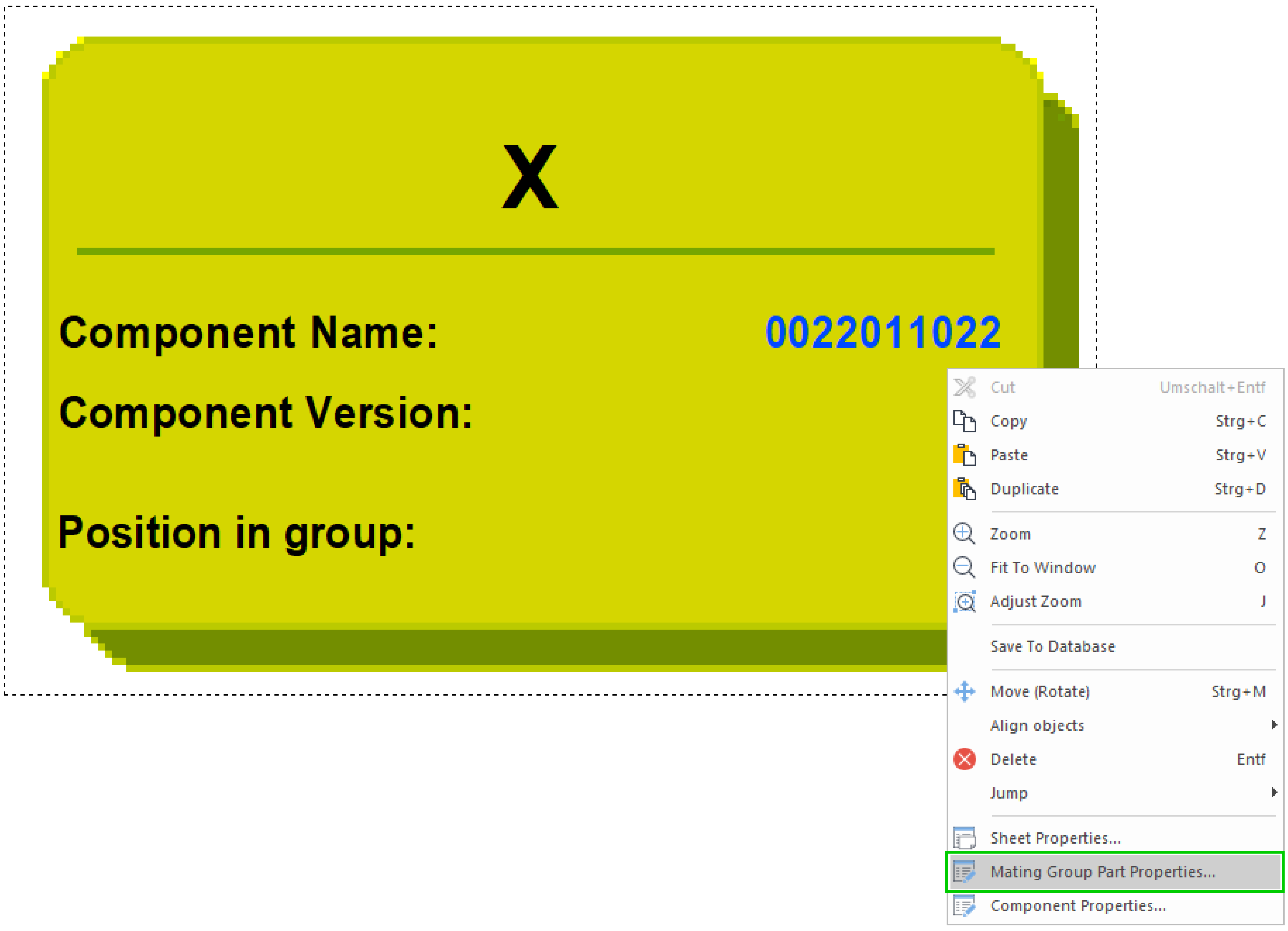
To change the properties of the complete mating connector group, either open the context menu of the insert group in the component tree and select ***Properties...*** or open the context menu anywhere on the worksheet and select ***Mating Group Properties...***:





To change the properties of individual connectors in a mating connector group, open the context menu for the relevant connector either in the component tree or on the worksheet and select ***Mating Group Part Properties...***:





**References:** `Designer-07495` and `Designer-20398`

## Importing and Exporting Data

### Importing Data

### Exporting Data

### Transfer Cutouts, Drill Holes, Threaded Holes and Space Requirements from Imported DXF Files to E³.series

Cutouts, drill holes, threaded holes and the space required, which are recognized as graphics with a defined color when importing DXF files, can be transferred to ***E³.*series** .
This means that all types of contours imported by DXF files can be functionally displayed and used in ***E³.*series** .

To highlight cutouts, drill holes, threaded holes and the space required from imported DXF files in color, adapt the configuration file `ImportDXF.cfg` in the installation directory of ***E³.*series** or create a new configuration file that can be specified via the API using the functions **e3Application.SetConfigFile()** or **e3DbeApplication.SetConfigFile()**.

The highlighting of contours and the space required is controlled with the following keys in the configuration file:

| Key | Format | Description |
|---|---|---|
| `[Outline]` | `Outline` = <ACI color from `AUTOCAD`><br><br>Example:<br><br>`Outline = 4` recognizes graphic objects in cyan and converts them accordingly | The key specifies an ACI color that is used to recognize lines and polygons in ***E³.*series** and convert them into the space required.<br><br>All lines and polygons with the defined color are converted into the space required.<br><br>If `Outline` has the value -1, the space requirement is not determined when importing DXF files. |
| `[Cutout]` | `Cutout` = <ACI color from `AUTOCAD`><br><br>Example:<br><br>`Cutout = 6` recognizes graphic objects in magenta and converts them accordingly | The key specifies an ACI color that is used to recognize lines, circles and three-quarter circles in ***E³.*series** and convert them to cutouts, drilled holes or threaded holes.<br><br>All lines with the defined color are converted to cutouts.<br>All circles with the defined color are converted to drill holes.<br>All circles that lie in three-quarter circles and have the defined color are converted to threaded holes.<br><br>If `Cutout` has the value -1, no cutouts, drill holes or threaded holes are created when importing DXF files. |

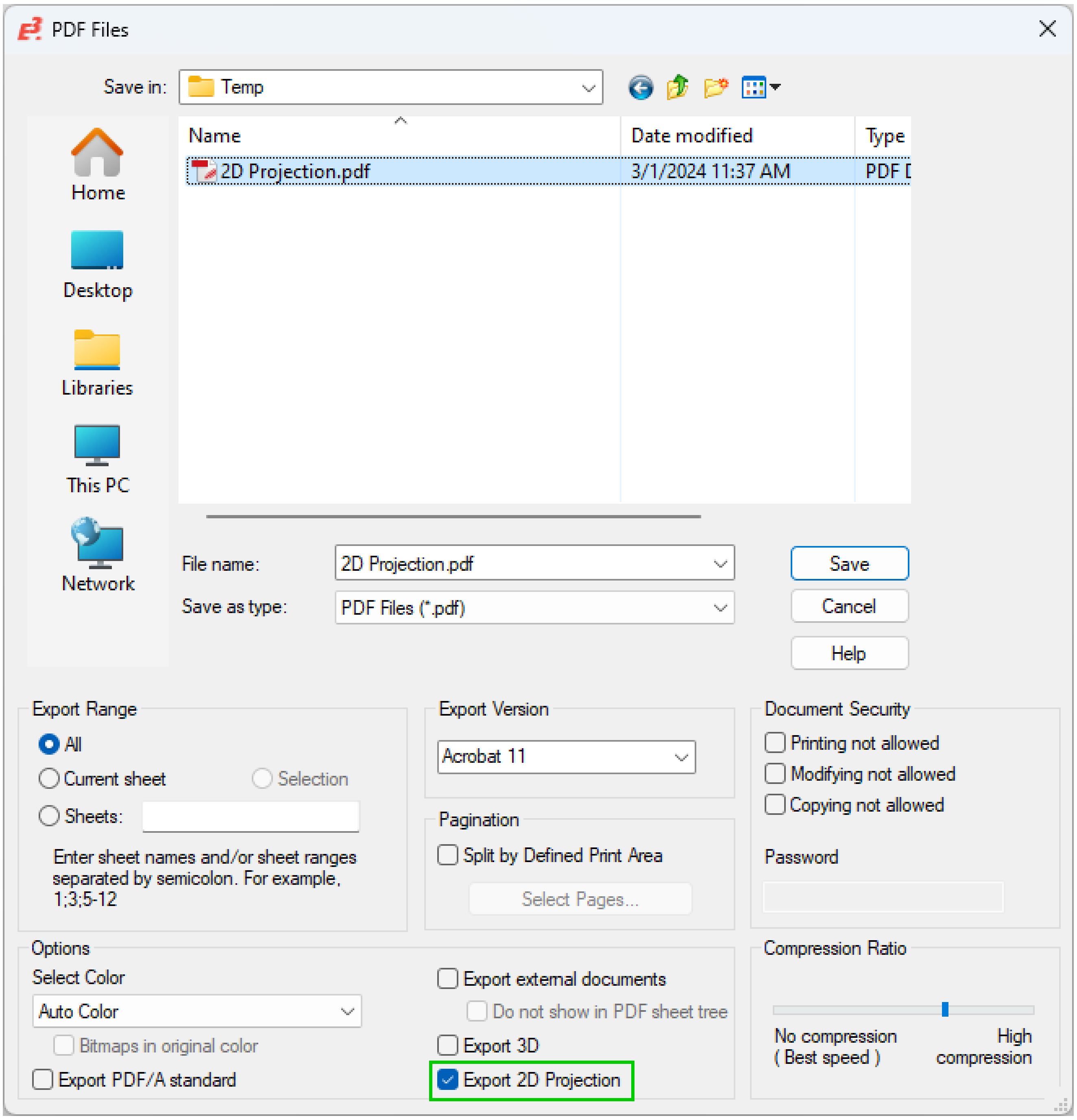
## Export Sheet Contents as 2D Projection in PDF Files

Sheet contents can be displayed as a 2D projection when exporting to PDF.

This allows, for example, the contents of all sheets to be displayed, regardless of whether relevant devices are not visible in the set 3D view.

To export sheet contents as a 2D projection in PDF files, select ***File → Export → PDF...*** in the main menu.
The ***PDF Files*** dialog opens.

Then set all export parameters and activate the option ***Export 2D Projection***:


PDF Files

Save in: Temp

| Name | Date modified | Type |
|---|---|---|
| 2D Projection.pdf | 3/1/2024 11:37 AM | PDF D |

File name: 2D Projection.pdf

Save as type: PDF Files (*.pdf)

Export Range: [✓ All] Current sheet Selection Sheets:

Enter sheet names and/or sheet ranges separated by semicolon. For example, 1;3;5-12

Export Version: Acrobat 11

Pagination: Split by Defined Print Area

Document Security: Printing not allowed, Modifying not allowed, Copying not allowed

Password

Options

Select Color: Auto Color

Bitmaps in original color

Export PDF/A standard

Export external documents

Do not show in PDF sheet tree

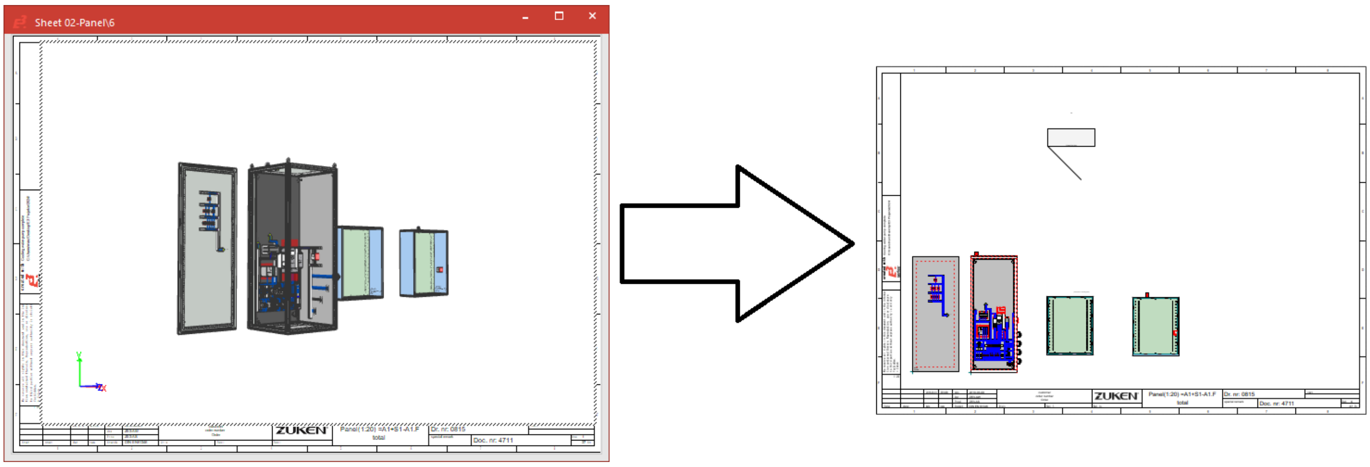
Export 3D

[✓] Export 2D Projection

Compression Ratio: No compression ( Best speed ) — High compression

The contents of panel sheets are exported to the PDF file as a 2D projection.
The display in the project is not changed.

**Example:**



## Set the Use of Level Configuration Files when Printing and Exporting to PDF and SVG

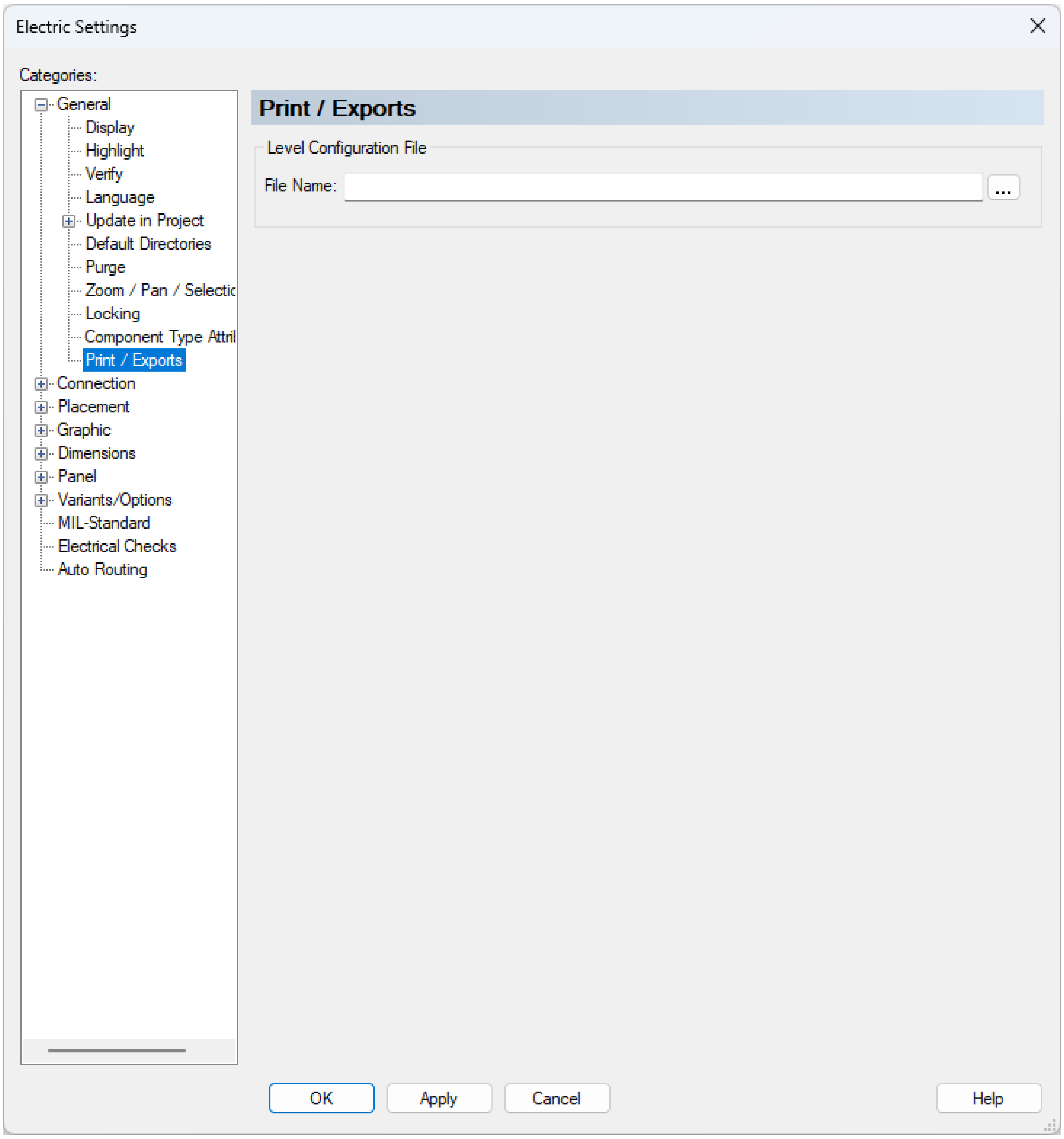
A level configuration file in `*.vis` format can be defined in the settings, which is used when printing and exporting PDF and SVG files.

This allows the visibility of objects that are placed on certain levels in ***E³.*series** to be controlled.
For example, information that is not relevant in printed projects or interferes with the clarity can be hidden by default.

To specify a configuration file for levels, select ***Tools → Settings...*** in the main menu.
The ***Electric Settings*** dialog opens.

**Note:** The setting can be made via the ***Electric Settings*** and ***Fluid Settings*** dialogs.

Then specify under ***General → Print / Exports → Level Configuration File***, which configuration file is used when printing and exporting to PDF and SVG.
The setting is project-specific:



Print / Exports

Level Configuration File

File Name:

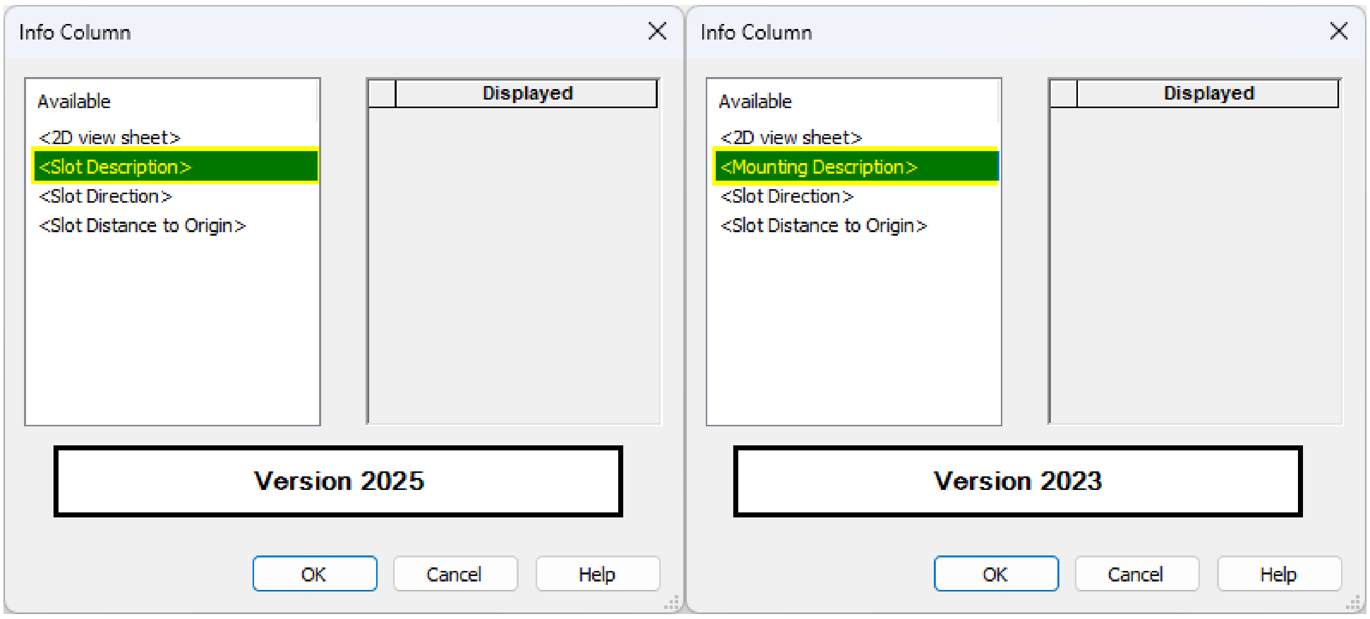
**Note:** The configuration file can be defined and saved using the ***Levels*** dialog. The dialog can be opened via the main menu under ***View*** → ***Levels...***.

**Reference:** `Designer-03148`

## Project Handling

### Changes to Info Column of Slots in Tree Structure

The attribute ***<Mounting Description>***, that is available when configuring the info column of slots in the tree structure, was renamed ***<Slot Description>***.
The information, previously labelled ***<Mounting Description>*** and starting from ***E³.*series** 2025 25.00 labelled ***<Slot Description>*** refers in both cases to the slot description:



This change makes it so the attribute available in the configuration dialog reflects the information that is displayed in the info column of the device tree.

### Model Attributes Are Automatically Converted

In previous versions of ***E³.*series** 2025, model attributes cannot be distinguished from attributes that were added or changed in projects.

To enable the same behavior with model attributes as with device attributes, projects with models are automatically converted when opened with ***E³.*series** 2025 but created with previous versions.

**Note:** For ***E³.*series** database editor projects, no conversion is necessary.

During conversion, all project instances of model attributes are deleted.

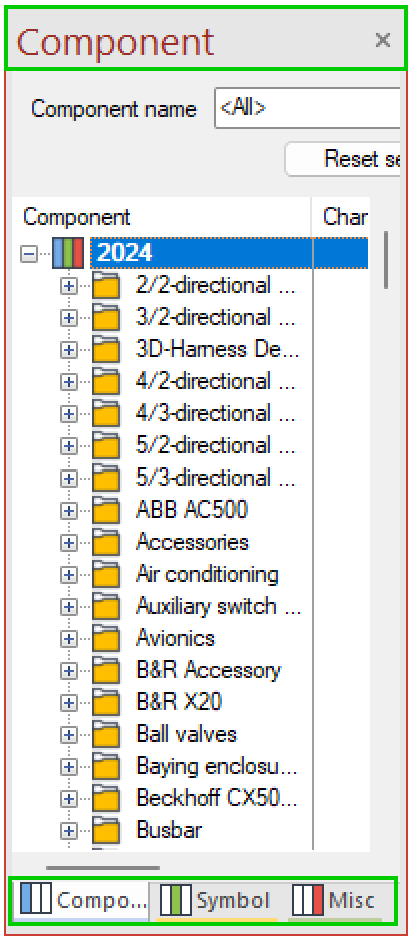
The attribute owner ***Model*** refers to the model created in the database.

## Search for Blocks in the Component Database based on the Block Connectors Used

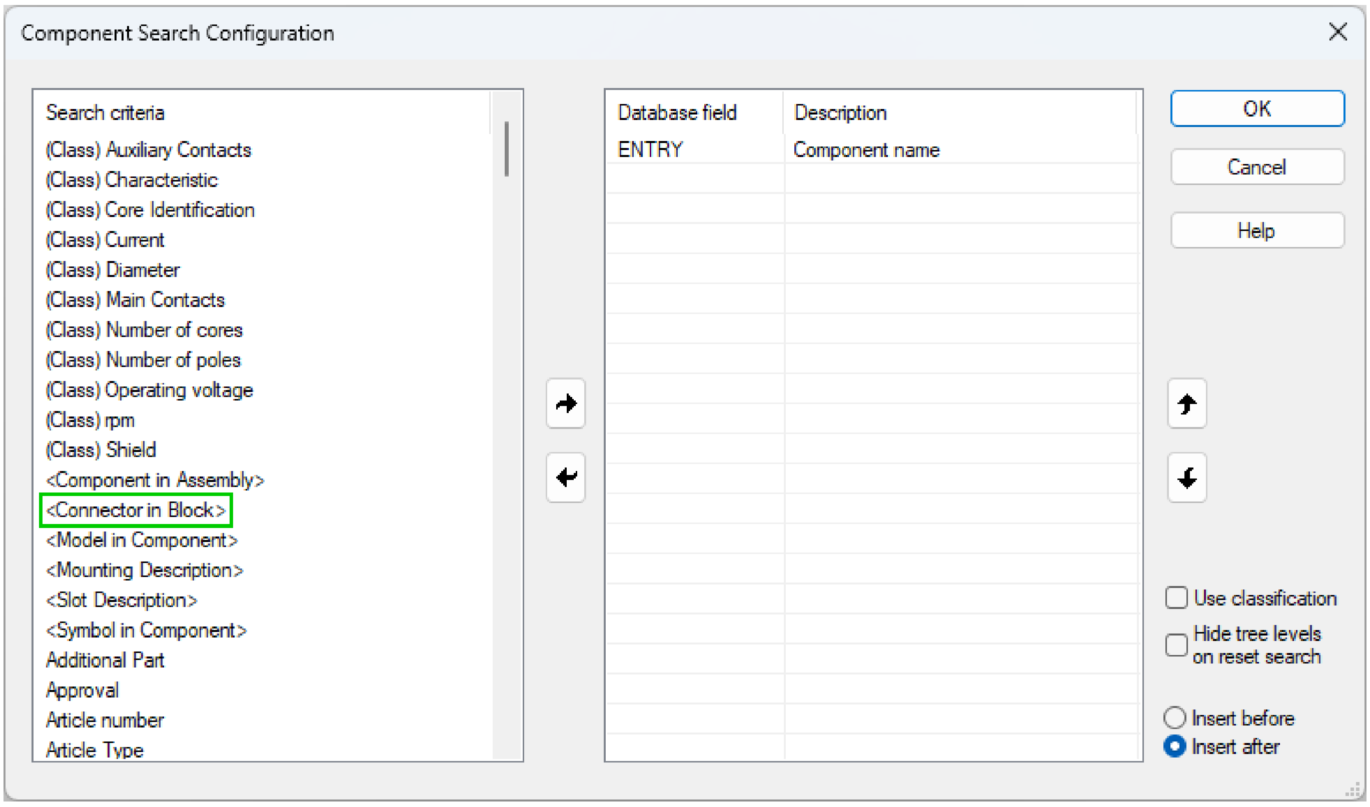
Blocks can be searched for in the component database based on the block connectors used.
The search result shows all block components containing the block connector being looked for.

This improves searching for elements of block components, thereby making the workflow with blocks more efficient.

To search for blocks based on the block connectors used in the component search, open the context menu by right-clicking on the upper frame or a tab of the database window:



Select the context menu command ***Component Search Configuration...***
The ***Component Search Configuration*** dialog opens:


Component Search Configuration

| Search criteria |
|---|
| (Class) Auxiliary Contacts |
| (Class) Characteristic |
| (Class) Core Identification |
| (Class) Current |
| (Class) Diameter |
| (Class) Main Contacts |
| (Class) Number of cores |
| (Class) Number of poles |
| (Class) Operating voltage |
| (Class) rpm |
| (Class) Shield |
| <Component in Assembly> |
| <Connector in Block> |
| <Model in Component> |
| <Mounting Description> |
| <Slot Description> |
| <Symbol in Component> |
| Additional Part |
| Approval |
| Article number |
| Article Type |

| Database field | Description |
|---|---|
| ENTRY | Component name |

☐ Use classification
☐ Hide tree levels on reset search
Insert before [✓ Insert after]

Then select the ***<Connector in Block>*** criterion under ***Search Criteria*** on the left-hand side and move it to the right-hand side by double-clicking or using the ➦ icon.
It is also possible to add additional search criteria or remove existing search criteria by double-clicking or using ⬅ .

Finally, confirm any changes with ***OK***.

**Reference:** `Designer-43634`

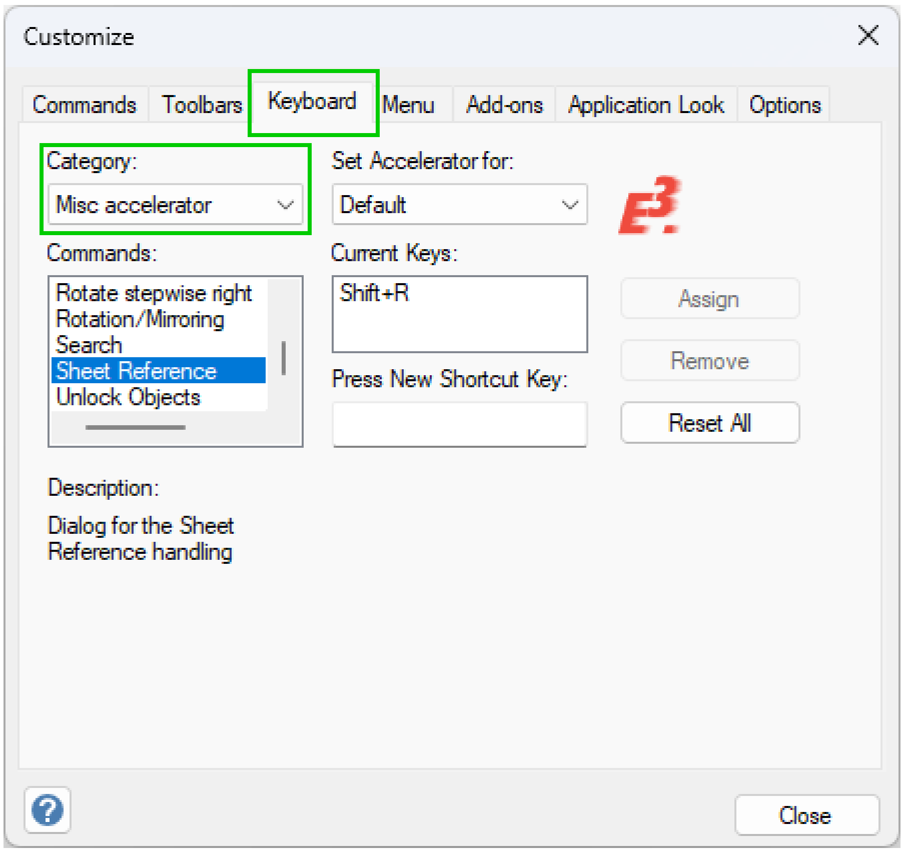
## Use Hotkey for *Sheet References* Dialog

A hotkey is available for the ***Sheet References*** dialog.

This makes editing sheet references easier and more user-friendly.
By default, the hotkey is defined as **Shift + R**.

To open the dialog with a hotkey, it is also optionally possible to change the hotkey assigned to the ***Sheet References*** dialog.
To do so, select ***Tools*** → ***Customize...*** in the main menu.
The ***Customize*** dialog opens.

Then change the desired key combination under ***Misc accelerator*** in the ***Keyboard*** tab to open the ***Sheet References*** dialog with a hotkey:


Customize

Commands | Toolbars | Keyboard | Menu | Add-ons | Application Look | Options

Category: Misc accelerator

Set Accelerator for: Default

Commands:
- Rotate stepwise right
- Rotation/Mirroring
- Search
- Sheet Reference
- Unlock Objects

Current Keys: Shift+R

Press New Shortcut Key:

Description:
Dialog for the Sheet Reference handling

Afterwards, left click on a placed reference symbol and use the selected hotkey.

Depending on the selected reference one of the following dialog for sheet references opens.
If the reference is of the type ***Auto Point to Point***, the ***Change Type of Reference*** dialog opens.
If the reference is of another type, the ***Sheet References*** dialog opens.

**References:** `Designer-07090`, `Designer-3078`, `Designer-33083` and `Designer-38148`

## Use Symbol Graphics for Dimensions

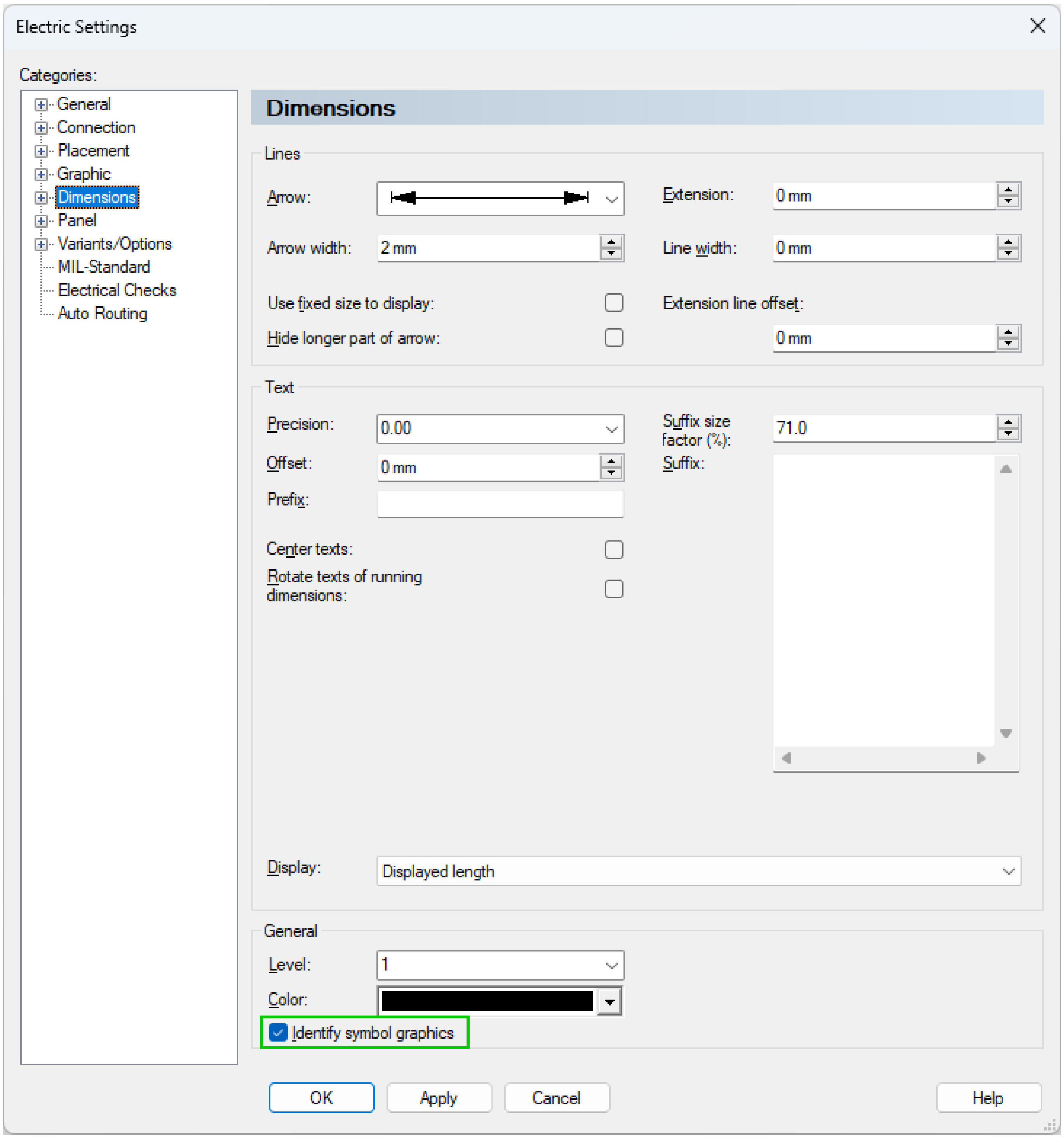
A setting can be used to control whether symbol graphics can be used for dimensions.

This allows each user to individually determine whether points for dimensions can be set on symbol graphics.
Symbol graphics can be used for dimensions as of ***E³.*****series** Version 2025.

To make changes to the setting for symbol graphics as reference objects for dimensions, select ***Tools → Settings...*** in the main menu.
The ***Electric Settings*** dialog opens.

**Note:** The setting can be made in the ***Electric Settings*** and the ***Fluid Settings*** dialogs.

Deactivate or reactivate the setting ***Identify symbol graphics*** under ***Dimensions***.
The setting is user-specific:


Electric Settings

Categories:
- General
- Connection
- Placement
- Graphic
- Dimensions
- Panel
- Variants/Options
- MIL-Standard
- Electrical Checks
- Auto Routing

Dimensions

Lines

Arrow:
Extension: 0 mm
Arrow width: 2 mm
Line width: 0 mm
Use fixed size to display: ☐
Extension line offset: 0 mm
Hide longer part of arrow: ☐

Text

Precision: 0.00
Suffix size factor (%): 71.0
Offset: 0 mm
Suffix:
Prefix:
Center texts: ☐
Rotate texts of running dimensions: ☐
Display: Displayed length

General

Level: 1
Color: ■
☑ Identify symbol graphics

If the setting is inactive, symbol graphics cannot be used for dimensions.
The values of dimensions can therefore depend on whether the setting is active when dimensions are created.

**Note:** Dimensions, which use symbol graphics, are not changed or removed if the setting is subsequently deactivated.

**Reference:** `Designer-43233`

## Use Angular Dimensions

The angle at which objects are placed in relation to each other can also be calculated and displayed with ***E³.*****series** .

This enables additional information to be obtained in ***E³.*****series** , for example, that is relevant for producing formboards or commissioning.

To use angular dimensions, open a project with objects between which the angle can be determined. The following objects are available for angle dimensioning:

- connection lines
- segments of the graphic type ***Rectangles***
- segments of the graphic type ***Polygons***
- segments of the graphic type ***Cloud***
- graphic type ***Line***
- graphic type ***Arc*** and ***Arc (3 Points)***
  **Note:** Angular dimensions can only be generated between the respective start and end points for arcs.

Select ***Insert*** → ***Angular Dimension*** in the main menu or the command ***Angular Dimension*** ( ) in the ***Graphic*** toolbar.
**Note:** Double-clicking on the command in the toolbar allows several angle dimensions to be inserted in succession. To exit this mode, press the **Esc** key.

Click on the two objects, in which an arc will be generated between them with angular dimensioning. The objects may not be parallel to one another.

Finally, click where the dimension text with the angle specification shall be placed. Depending on which side of the intersection the dimension text is placed, either the internal angle or the external angle is displayed.

**Note:** To stop the automatic switching between the inner and outer angle, hold down the **Shift** key and move the mouse to the desired position.

If the setting ***General*** → ***Display*** → ***Display rotated texts acc. to standard*** is active, dimension texts that display values between 0° and 180° are displayed outside the arc and values greater than 180° to 360° are displayed inside the arc.

Angular dimensions will be available when exporting in the following file formats:

- CGM
- SVG
- PDF
- DGN
  **Note:** Angle dimensions are only taken into account graphically when importing from DGN and exporting to DGN and are not available as geometric elements in the target application.

- DXF
  **Note:** Angle dimensions are only taken into account graphically when importing from DXF and exporting to DXF and are not available as geometric elements in the target application.

**References:** `Designer-16325` and `Designer-34952`

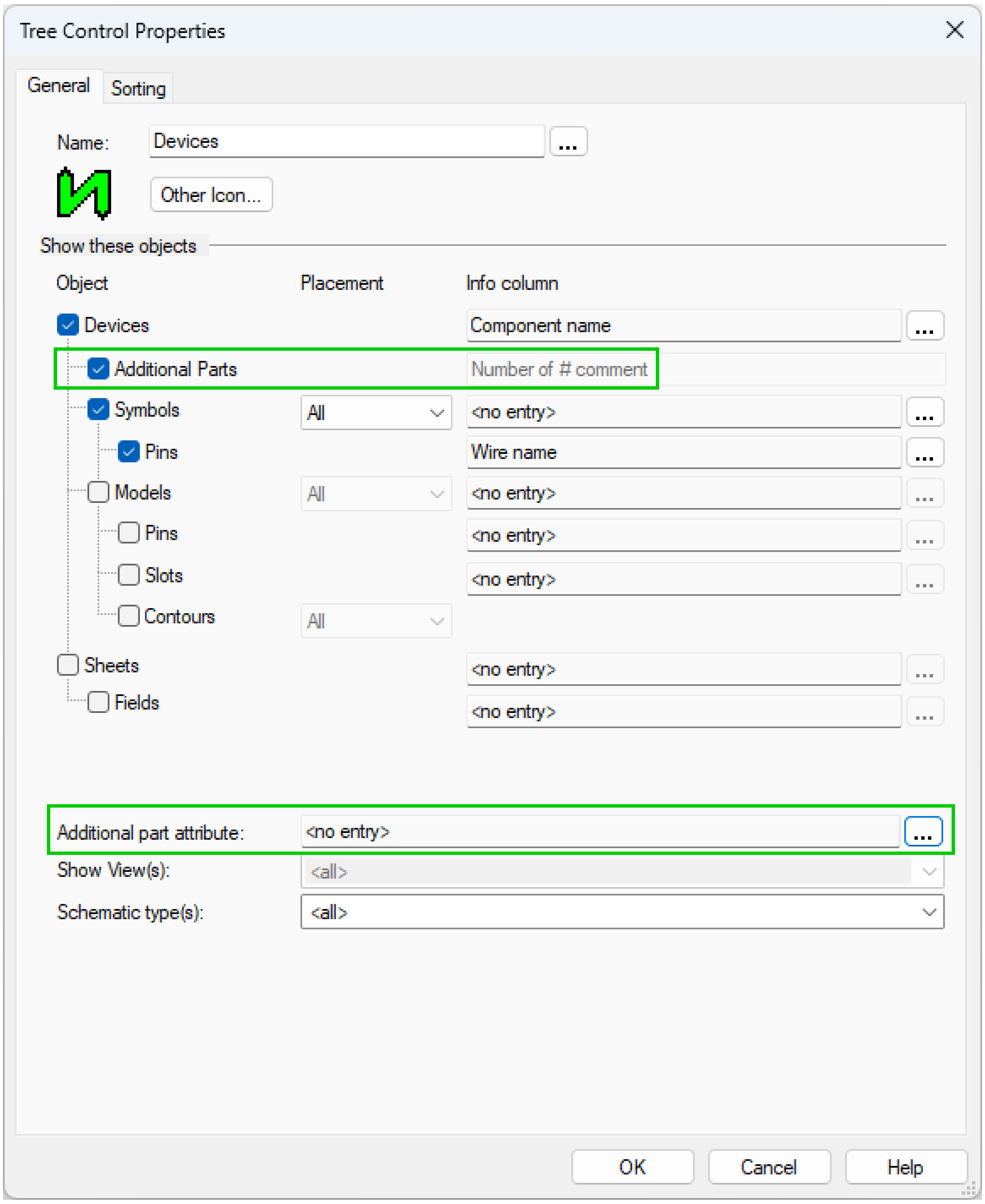
## Display Additional Parts and Their Attributes in the Device Tree

Device trees can be customized to display the additional parts and their attributes that belong to a device.

This makes it easier to see, for example, if devices have additional parts and also makes it clearer which objects are output in parts lists.

To display additional parts in the device tree, open the context menu on the uppermost level of the device tree or on a device and select ***Tree Control Properties...***
The ***Tree Control Properties*** dialog opens.

Activate the ***Additional parts*** check box.
Define an attribute that is used to identify additional parts under ***Additional parts attributes*** that is used to assign additional parts to devices.
Only one attribute can be selected at a time:



Name: Devices

Show these objects

| Object | Placement | Info column |
|---|---|---|
| ✓ Devices | | Component name |
| ✓ Additional Parts | | Number of # comment |
| ✓ Symbols | All | <no entry> |
| ✓ Pins | | Wire name |
| Models | All | <no entry> |
| Pins | | <no entry> |
| Slots | | <no entry> |
| Contours | All | |
| Sheets | | <no entry> |
| Fields | | <no entry> |

Additional part attribute: <no entry>

Show View(s): <all>

Schematic type(s): <all>

**Note:** If no attribute is specified under ***Additional part attribute***, additional parts are not displayed in the device tree.

All attributes that have the type ***Text*** and one of the following owners can be defined for identification purposes:

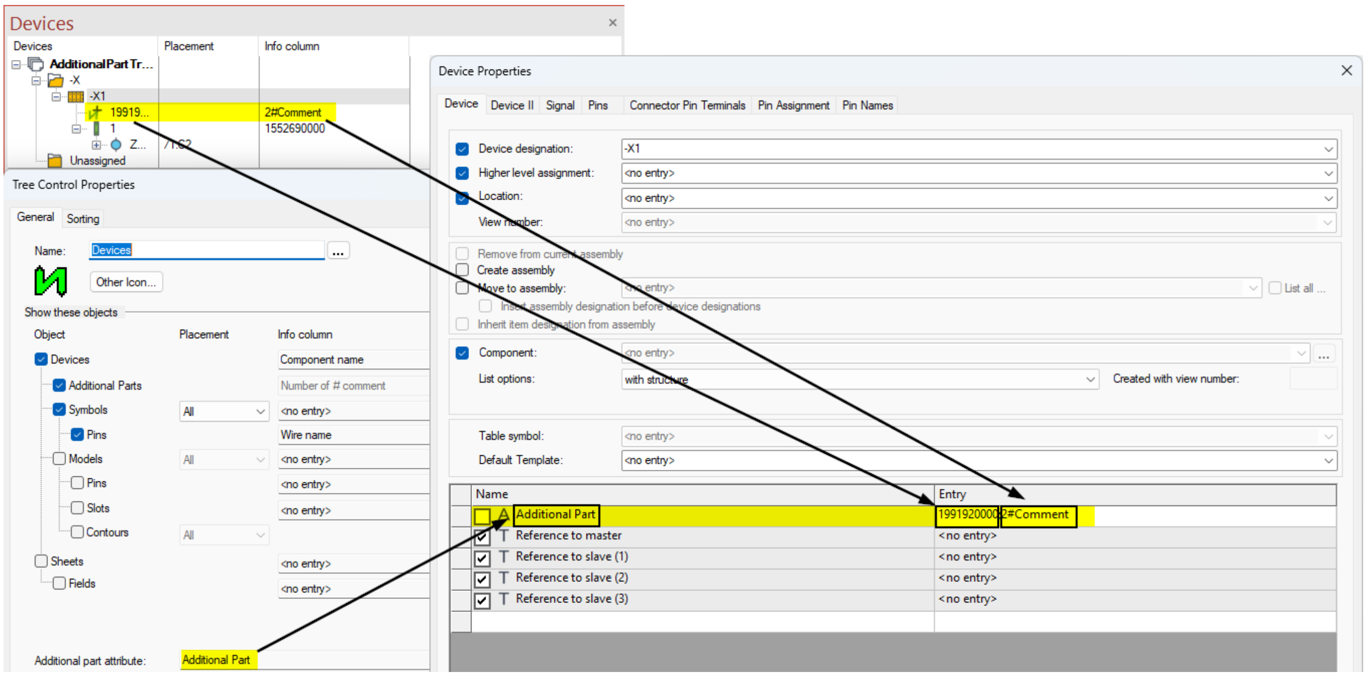
- Device
- Cable
- Connector

- Block Connector
- Block
- Hose/Tube
- Busbar

The information of the additional part is determined via the value of the attribute.
To display the information in the device tree correctly, the value of the attribute must have the following structure:
***<Component Name>;<Number>#<Optional Commentary>***

Components that are defined as a value for the defined attribute are displayed in the device tree with the following icon:

**Example:**



Devices

| Devices | Placement | Info column |
|---|---|---|
| AdditionalPart Tr... | | |
| -X | | |
| -X1 | | |
| 19919... | | 2#Comment |
| 1 | | 1552690000 |
| Z... | /1.C2 | |
| Unassigned | | |

Tree Control Properties

General | Sorting

Name: Devices

Other Icon...

Show these objects

| Object | Placement | Info column |
|---|---|---|
| ✓ Devices | | Component name |
| ✓ Additional Parts | | Number of # comment |
| ✓ Symbols | All | <no entry> |
| ✓ Pins | | Wire name |
| ☐ Models | All | <no entry> |
| ☐ Pins | | <no entry> |
| ☐ Slots | | <no entry> |
| ☐ Contours | All | |
| ☐ Sheets | | <no entry> |
| ☐ Fields | | <no entry> |

Additional part attribute: Additional Part

Device Properties

Device | Device II | Signal | Pins | Connector Pin Terminals | Pin Assignment | Pin Names

| | |
|---|---|
| ✓ Device designation: | -X1 |
| ✓ Higher level assignment: | <no entry> |
| ✓ Location: | <no entry> |
| View number: | <no entry> |
| ☐ Remove from current assembly | |
| ☐ Create assembly | |
| ☐ Move to assembly: | <no entry> ☐ List all ... |
| ☐ Insert assembly designation before device designations | |
| ☐ Inherit item designation from assembly | |
| ✓ Component: | <no entry> |
| List options: | with structure |
| Created with view number: | |
| Table symbol: | <no entry> |
| Default Template: | <no entry> |

| Name | Entry |
|---|---|
| ☐ Additional Part | 19919200002#Comment |
| ✓ Reference to master | <no entry> |
| ✓ Reference to slave (1) | <no entry> |
| ✓ Reference to slave (2) | <no entry> |
| ✓ Reference to slave (3) | <no entry> |

**Note:** To add additional parts in the ***E³.*****series** database, the attribute ***Additional Parts*** is used by default. However, other attributes can also be used to display additional parts in the tree view if they are defined for the component and entered under ***Additional part attribute*** accordingly.

Observe the following rules for displaying information on additional parts:

- Each copy of the selected attribute on the device is displayed as an additional part and shown in a new line.
- The ***<Component Name>*** left of the ***;*** is displayed in the ***Device*** column of the Device Tree.
- The ***<Number>*** and the ***<Optional Commentary>*** to the right of ***#*** is displayed in the ***Info Column*** of the Device Tree.
- Attribute values are ignored and not displayed as an additional part in the following cases:

    - if they begin with ***#***,
    - if they are empty, or
    - if there is no content before the ***;***.
- For attribute values that do not contain a ***;***, the text is displayed up to the first space in the ***Device*** column.
  The complete text after the first blank space is displayed in the ***Info Column***.
- Translation codes, for example ***&5015;*** for the translated text ***Add additional part***, are only translated if they appear in the ***Info Column***.
  The translation code is not interpreted as translated text and displayed directly in the ***Device*** column.

## Panel (also: 3D)

### Display Dimensions in 3D

Dimensions can be displayed in 3D representations, albeit in a reduced form.

This means that information which is relevant for producing or designing control cabinets, for example, is also partially visible in the 3D models.

To display dimensions in 3D, open a panel sheet on which models with dimensions are placed.

Switch the display mode to ***3D sheet*** ( ) or ***Projection on 2D sheet*** ( ) in the ***Panel*** toolbar.

**Note:** Dimensions are only visible in ***3D sheet*** display mode if the model is aligned from the front and the display mode ***Switch projection*** ( ) is turned off.
To align a rotated model from head on, for example, use the context menu command ***Display in original justification - from Front***or the function ***Fit To Window***, which is executed by default when pressing the key **O**.

### Align Devices with corresponding Carrier Objects on Panel Sheets

Devices, which are placed on panel sheets and mounted on carriers, can be aligned on their respective carrier object.
As carrier objects, in addition to mounting rails as in earlier versions of ***E³.*series**, for example, mounting plates can also be used.

This means that objects that have only been provisionally placed on their respective carrier via a tree view, for example, can be distributed evenly to make use of the available space in the panel.

To align a device with its carrier object, open the panel sheet with the placed devices.

Then select the desired devices and their common carrier object and execute one of the following commands in the context menu or the ***Align objects*** toolbar:

- **Align left**
- **Align right**
- **Distribute Horizontally**
- **Distribute Vertically**
- **Align top**
- **Align bottom**

The following new commands can also be used for aligning:

- **Align the Right Edge to the Vertical Center Line**
- **Align the Center to the Vertical Center Line**
- **Align the Left Edge to the Vertical Center Line**
- **Align the Bottom Edge to the Horizontal Center Line**
- **Align the Center to the Horizontal Center Line**
- **Align the Top Edge to the Horizontal Center Line**

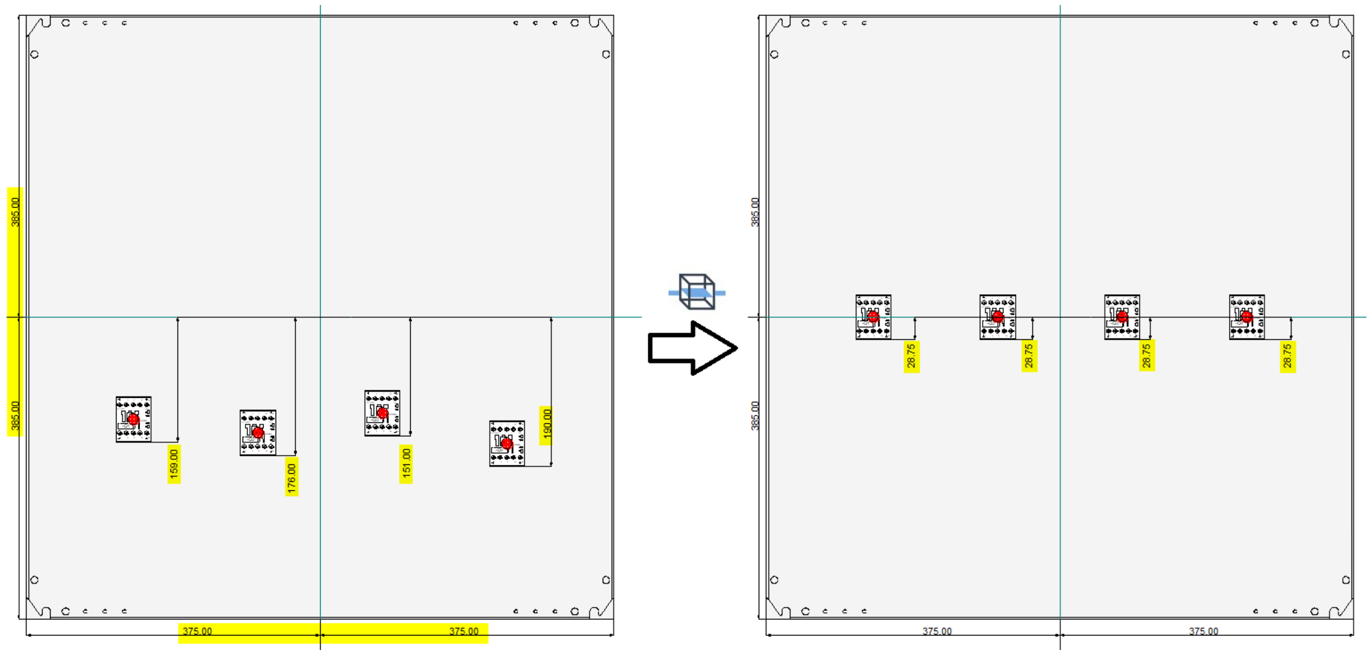
To align objects which are placed on a carrier object, select only the carrier and execute the desired command.

The devices are aligned on the carrier according to the selected command.

**Example:**

The four devices in the left half of the following illustration are placed in varying distances from the middle of each device to the horizontal center line of the mounting plate (see yellow highlighting).

Then select the carrier object and execute the command ***Align the Center to the Horizontal Center Line***.
The devices are aligned centrally on the horizontal center line of the mounting plate.
The results can be seen in the right half of the illustration:



## Place and Move Objects in the Panel via Tree Views

Objects can be placed and moved in the panel via tree views.
Slots of the type ***Point*** or models with a compatible ***Point*** type slot can be selected as a placement target.

**Filter for Slot Properties** can also be used in the tree search.

This makes designing panels with ***E³.*****series** even more user-friendly, especially for example when there are a large number of slots in projects or several slots lie close together.

## Place/Move Objects

To place or move objects in the panel using a tree view, select an object to be placed in a project window, for example, ***Device***, ***Panel - Placed***, ***Panel - Unplaced*** or in the database window ***Component***.
The following objects can be placed via tree views:

- models
- devices with models
- components with models

The following objects can be used as placement targets when placing via tree views:

- ***Point*** type slots
- models with ***Point*** type slots

---

Objects can be placed via tree views in two different ways:

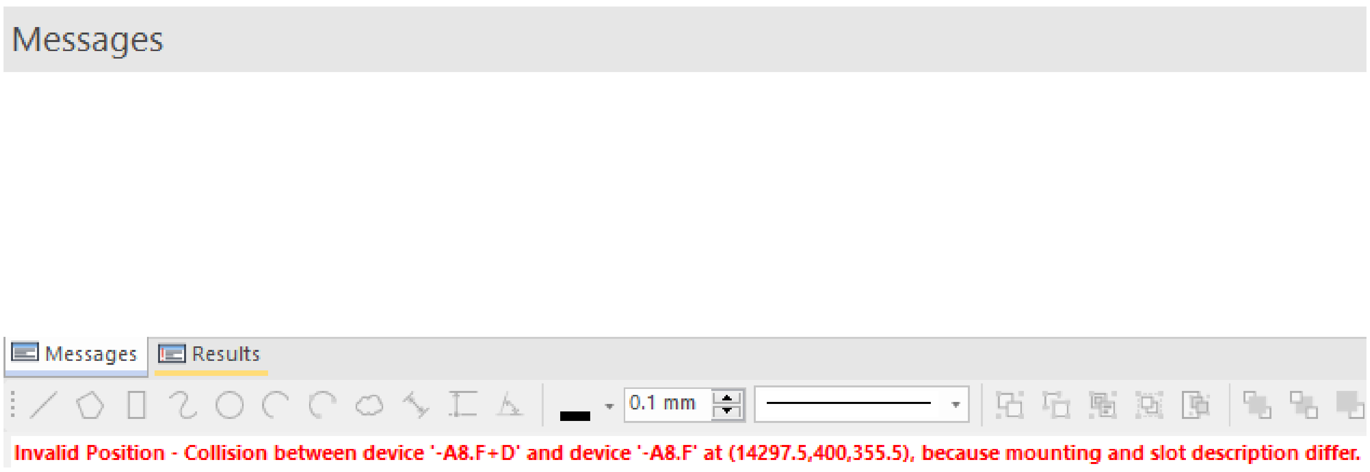
- drag and drop the object, which is to be placed or moved, on the placement target, or
- right-click on the object, select the context menu command ***Place*** and then place at the desired location

The placement is only possible in the following cases:

- the slot descriptions are identical
- the object can be placed without any collisions
- the slot, on which the object is to be placed, has the type ***Point*** or the model, on which the object is to be placed, has at least one slot of the type ***Point***
  **Note:** If objects are placed on models via tree views, the first compatible slot of the model is used for the placement.

If it is not possible to place at the desired location, the mouse cursor changes to the prohibited sign:
🚫

A message is also displayed in the status bar indicating the reason why the placement is not possible.



Invalid Position - Collision between device '-A8.F+D' and device '-A8.F' at (14297.5,400,355.5), because mounting and slot description differ.

### Change the Rotation of Objects Placed via Tree Views

If the rotation is not changed, objects are rotated automatically when placing so that the slot direction and rotation match.

Optionally , the rotation and inclination of an object can also be changed before placing or placed with its original orientation.
To do so, open a panel sheet, drag the object to be placed from a tree view on the sheet and change the rotation or inclination with the following key combinations:

- **R**: **Rotate** the object to the **right**
- **Shift + ↑**: pivots the object backwards
  or **Ctrl + ↑**: pivots the object backwards in 15° steps
- **Shift + →** : pivots the object to the right
  or **Ctrl + →** : pivots the object to the right in 15° steps
- **Shift + ↓**: pivots the object forwards
  or **Ctrl + ↓**: pivots the object forwards in 15° steps
- **Shift + ←** : pivots the object to the left
  or **Ctrl + ←** : pivots the object to the left in 15° steps

**Note:** If the object is placed on a slot with the command ***Place***,the rotation can also be changed using the ***Rotation/Mirroring*** dialog.
In this case, activate the setting ***Suppress automatic rotation on slots***.
When placing with modified rotation values, the following restrictions apply to assemblies and connectors with inserts:

- assemblies can only be rotated if the command ***Place Single Model*** is used for the placement
- for connectors with inserts that should be placed as rotated, only the inserts are rotated

Finally, press and hold the **Shift** key and place the object on the slot.
The object is placed with the set rotation and does not assume the direction and rotation of the slot.
If the rotation has not been manually changed, the object is placed with the values defined in the database.

### Use Slot Properties as Filter for the Tree Search

Slots can be searched for and filtered based on their properties in the tree view.

To configure the tree search, right-click on the project window or on the header of a project tree column and select the context menu command ***Search Configuration...***
The ***Tree Control Search Configuration*** opens.

Select all slot properties, which should be available for filtering in the tree search.
Slots can be searched and filtered based on the properties listed in the following table:

| Property | Attribute name in configuration file of tree search |
|---|---|
| ***Higher Level Assignment*** | `HigherLevelAssignment` |
| ***Device Designation*** | `DeviceDesignation` |
| ***Location*** | `Location` |
| ***Slot Name*** | `SlotName` |

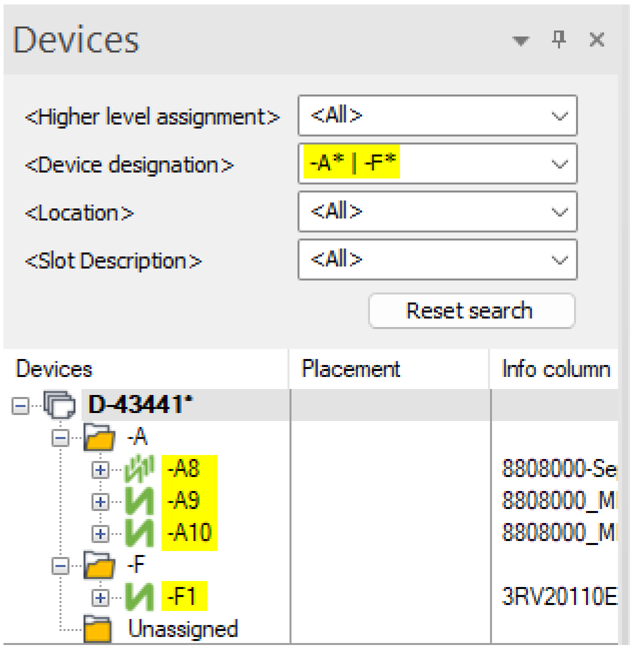
| | |
|---|---|
| ***Slot Type*** | `SlotType` |
| ***Slot Distance to Origin*** | `SlotDistanceToOrigin` |
| ***Slot Description*** | `SlotDescription` |
| ***Slot Direction*** | `SlotDirection` |

The property ***Slot distance to origin*** refers to the position in the direction of the X, Y or Z axis defined for the slot.

The wildcard ***** can be used to replace an unlimited number of characters when filtering by entries. Furthermore, several entries can also be linked with the logical **OR**, which is expressed with the character **|**.

**Note:** Objects, which are placed on a compatible slot via the tree view, are then displayed in the tree view even if they have no matching filter criteria.
The filtered view is updated when the filter criteria are changed, reset or reloaded from a configuration file.

**Example:**



| | |
|---|---|
| <Higher level assignment> | <All> |
| <Device designation> | -A* \| -F* |
| <Location> | <All> |
| <Slot Description> | <All> |

| Devices | Placement | Info column |
|---|---|---|
| D-43441* | | |
| -A | | |
| -A8 | | 8808000-Se |
| -A9 | | 8808000_M |
| -A10 | | 8808000_M |
| -F | | |
| -F1 | | 3RV20110E |
| Unassigned | | |

The screenshot above shows all devices, whose ***Device Designation*** begins with ***-A*** (***-A****) or (**|**) with ***-F*** (***-F****).

The following operators can also be used for the property ***Slot distance to origin*** when filtering:

- **<**: "less than"
- **<=**: "less than or equal to"
- **>**: "greater than"
- **>=**: "greater than or equal to"
- **=**: "equal to"

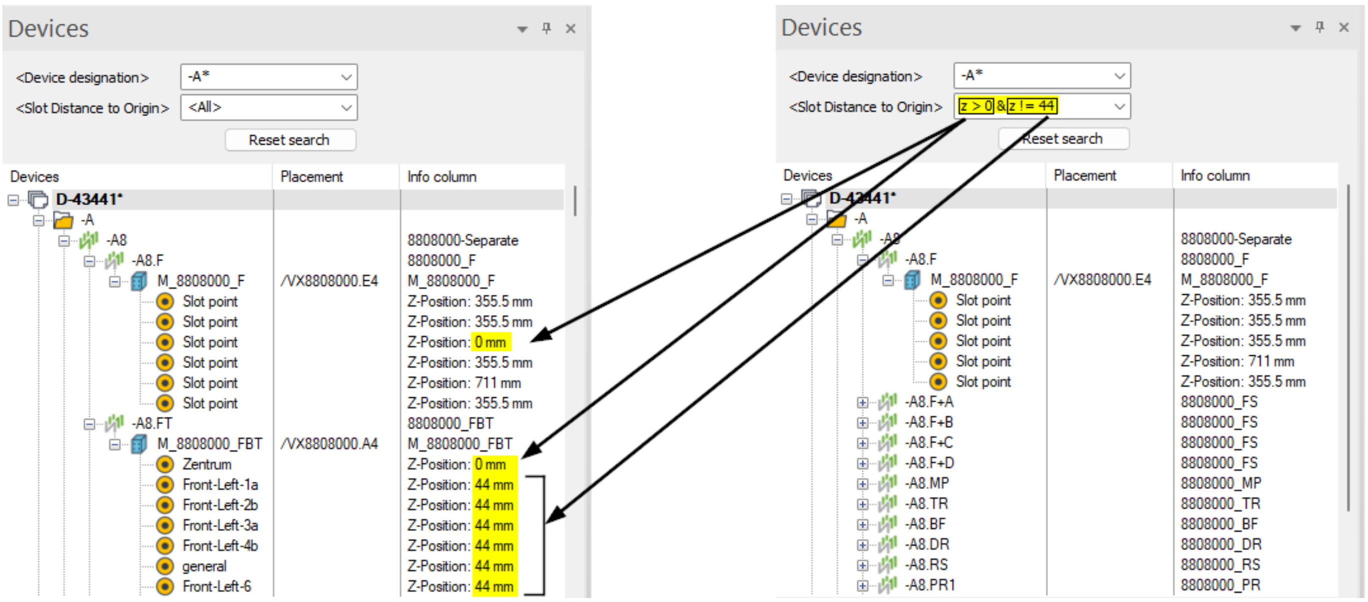
- **!=**: "not equal to"
- **&**: "logical AND"

A decimal point (**.**) is used as the separator for all languages.

**Note:** The unit does not have to be specified when filtering. The numerical value is automatically converted to the standard unit.
If a unit is specified, it is converted to the standard unit. This means, for example, the following search criteria will return the same results:

- x < 100
- x < 100 mm
- x < 10 cm
- x < 0.1 m

**Example:**



Devices

| | |
|---|---|
| <Device designation> | -A* |
| <Slot Distance to Origin> | <All> |

Reset search

| Devices | Placement | Info column |
|---|---|---|
| D-43441* | | |
| -A | | |
| -A8 | | 8808000-Separate |
| -A8.F | | 8808000_F |
| M_8808000_F | /VX8808000.E4 | M_8808000_F |
| Slot point | | Z-Position: 355.5 mm |
| Slot point | | Z-Position: 355.5 mm |
| Slot point | | Z-Position: 0 mm |
| Slot point | | Z-Position: 355.5 mm |
| Slot point | | Z-Position: 711 mm |
| Slot point | | Z-Position: 355.5 mm |
| -A8.FT | | 8808000_FBT |
| M_8808000_FBT | /VX8808000.A4 | M_8808000_FBT |
| Zentrum | | Z-Position: 0 mm |
| Front-Left-1a | | Z-Position: 44 mm |
| Front-Left-2b | | Z-Position: 44 mm |
| Front-Left-3a | | Z-Position: 44 mm |
| Front-Left-4b | | Z-Position: 44 mm |
| general | | Z-Position: 44 mm |
| Front-Left-6 | | Z-Position: 44 mm |

Devices

| | |
|---|---|
| <Device designation> | -A* |
| <Slot Distance to Origin> | z > 0 & z != 44 |

Reset search

| Devices | Placement | Info column |
|---|---|---|
| D-43441* | | |
| -A | | |
| -A8 | | 8808000-Separate |
| -A8.F | | 8808000_F |
| M_8808000_F | /VX8808000.E4 | M_8808000_F |
| Slot point | | Z-Position: 355.5 mm |
| Slot point | | Z-Position: 355.5 mm |
| Slot point | | Z-Position: 355.5 mm |
| Slot point | | Z-Position: 711 mm |
| Slot point | | Z-Position: 355.5 mm |
| -A8.F+A | | 8808000_FS |
| -A8.F+B | | 8808000_FS |
| -A8.F+C | | 8808000_FS |
| -A8.F+D | | 8808000_FS |
| -A8.MP | | 8808000_MP |
| -A8.TR | | 8808000_TR |
| -A8.BF | | 8808000_BF |
| -A8.DR | | 8808000_DR |
| -A8.RS | | 8808000_RS |
| -A8.PR1 | | 8808000_PR |

All devices, whose ***Device designation*** begins with ***-A***, are displayed in the left half of the screenshot. In the right half, the filter ***z > 0 & z != 44*** further restricts the search results and only shows slots whose value for the Z position is greater than***0*** and not equal to ***44***.

## Define the Routing Offset for Model Pins

A routing offset can be defined for model pins.
Conductors/wires, which are connected to pins that have a routing offset, are correspondingly connected to the "shifted" pins.

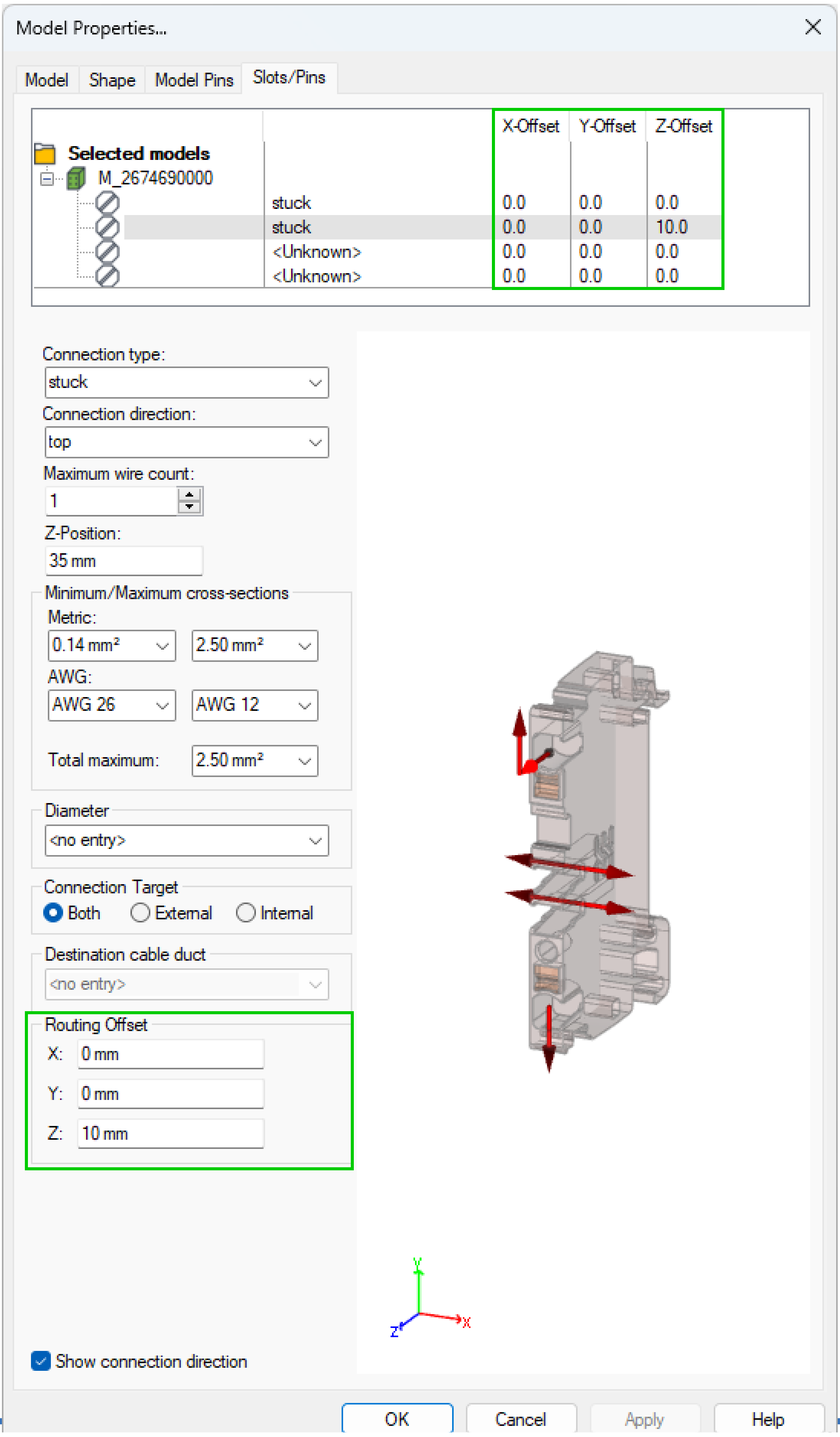
This means, for example, that connections with pins that are difficult to access in the control cabinet can be routed via a specific path.
The offset is taken into account in the length of the connection, which can lead to a longer or shorter installation path.

**Note:** To take collisions into account between the pin and the point defined with the offset for automatic connections, the setting ***Panel → Checks → Autoconnect → Wire/Conductor from pin ↔ Component/Restricted*** must be active.

To define a routing offset for model pins in the database, select ***Tools → Start Database Editor*** in the main menu.
The database editor opens.

Search for a model whose pins should be defined a routing offset and then open the Model Properties. Switch to the ***Slots/Pins*** tab.

Under ***Routing Offset*** enter the values (***X***, ***Y*** and ***Z***), in which the pin is to be "offset".
The values are displayed in the pin table:



Model Properties...

Model | Shape | Model Pins | Slots/Pins

Selected models
M_2674690000

| | | X-Offset | Y-Offset | Z-Offset |
|---|---|---|---|---|
| | stuck | 0.0 | 0.0 | 0.0 |
| | stuck | 0.0 | 0.0 | 10.0 |
| | <Unknown> | 0.0 | 0.0 | 0.0 |
| | <Unknown> | 0.0 | 0.0 | 0.0 |

Connection type: stuck

Connection direction: top

Maximum wire count: 1

Z-Position: 35 mm

Minimum/Maximum cross-sections

Metric: 0.14 mm² 2.50 mm²

AWG: AWG 26 AWG 12

Total maximum: 2.50 mm²

Diameter: <no entry>

Connection Target: [✓ Both] External Internal

Destination cable duct: <no entry>

Routing Offset

X: 0 mm

Y: 0 mm

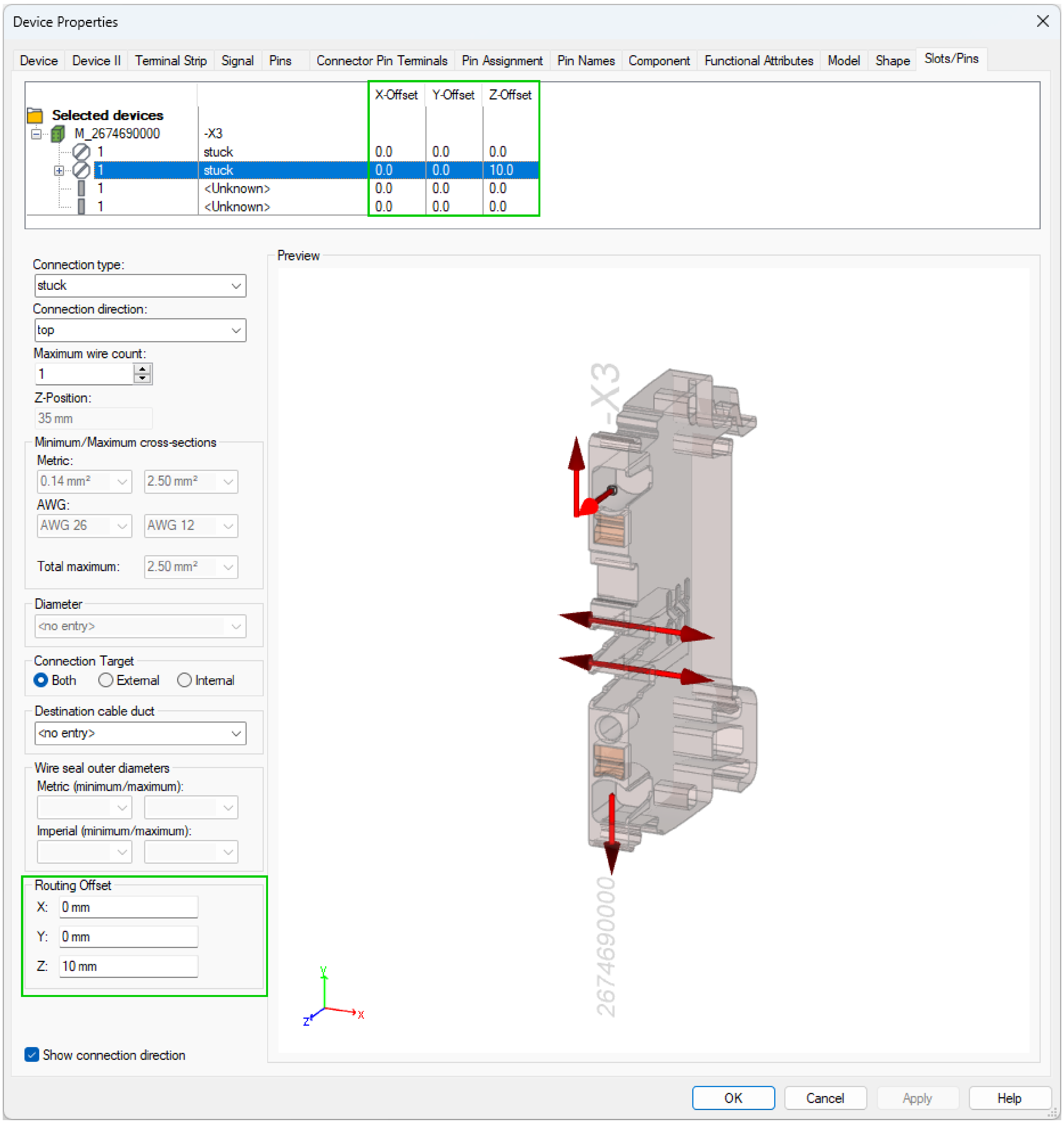
Z: 10 mm

[✓] Show connection direction

All devices, which are used in new projects and the model, will automatically have pins with the defined offset values.

The values can also be changed later in the project via the device properties.

Switch to the ***Slot/Pins*** tab and change or set the ***Routing Offset*** for the desired pin:


Device Properties
Device | Device II | Terminal Strip | Signal | Pins | Connector Pin Terminals | Pin Assignment | Pin Names | Component | Functional Attributes | Model | Shape | Slots/Pins
Preview
-X3
2674690000
OK | Cancel | Apply | Help

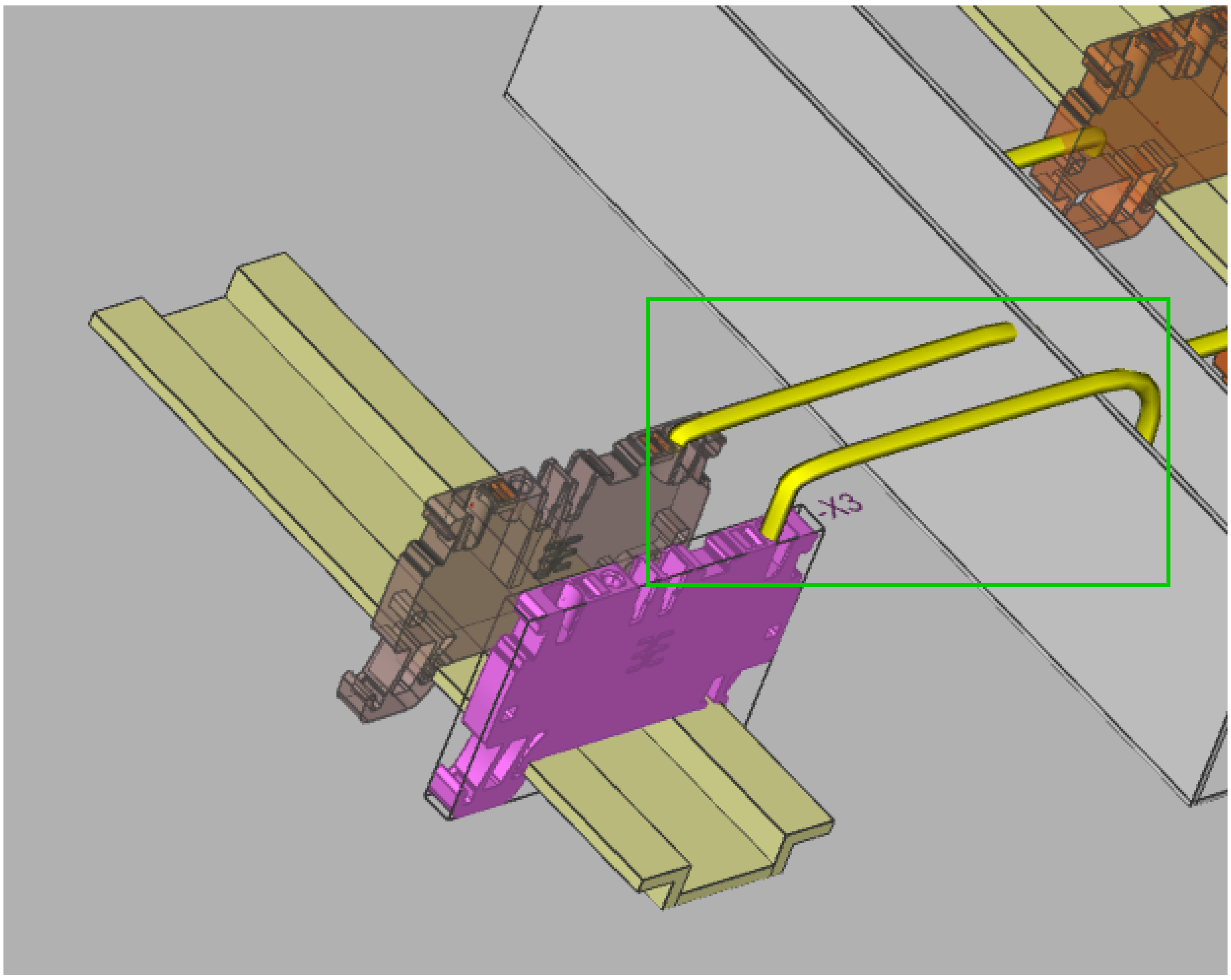
| | | X-Offset | Y-Offset | Z-Offset |
|---|---|---|---|---|
| Selected devices | | | | |
| M_2674690000 | -X3 | | | |
| 1 | stuck | 0.0 | 0.0 | 0.0 |
| 1 | stuck | 0.0 | 0.0 | 10.0 |
| 1 | <Unknown> | 0.0 | 0.0 | 0.0 |
| 1 | <Unknown> | 0.0 | 0.0 | 0.0 |

Connection type: stuck
Connection direction: top
Maximum wire count: 1
Z-Position: 35 mm
Minimum/Maximum cross-sections
Metric: 0.14 mm² / 2.50 mm²
AWG: AWG 26 / AWG 12
Total maximum: 2.50 mm²
Diameter: <no entry>
Connection Target: [✓ Both] External Internal
Destination cable duct: <no entry>
Wire seal outer diameters
Metric (minimum/maximum):
Imperial (minimum/maximum):
Routing Offset
X: 0 mm
Y: 0 mm
Z: 10 mm
[✓] Show connection direction

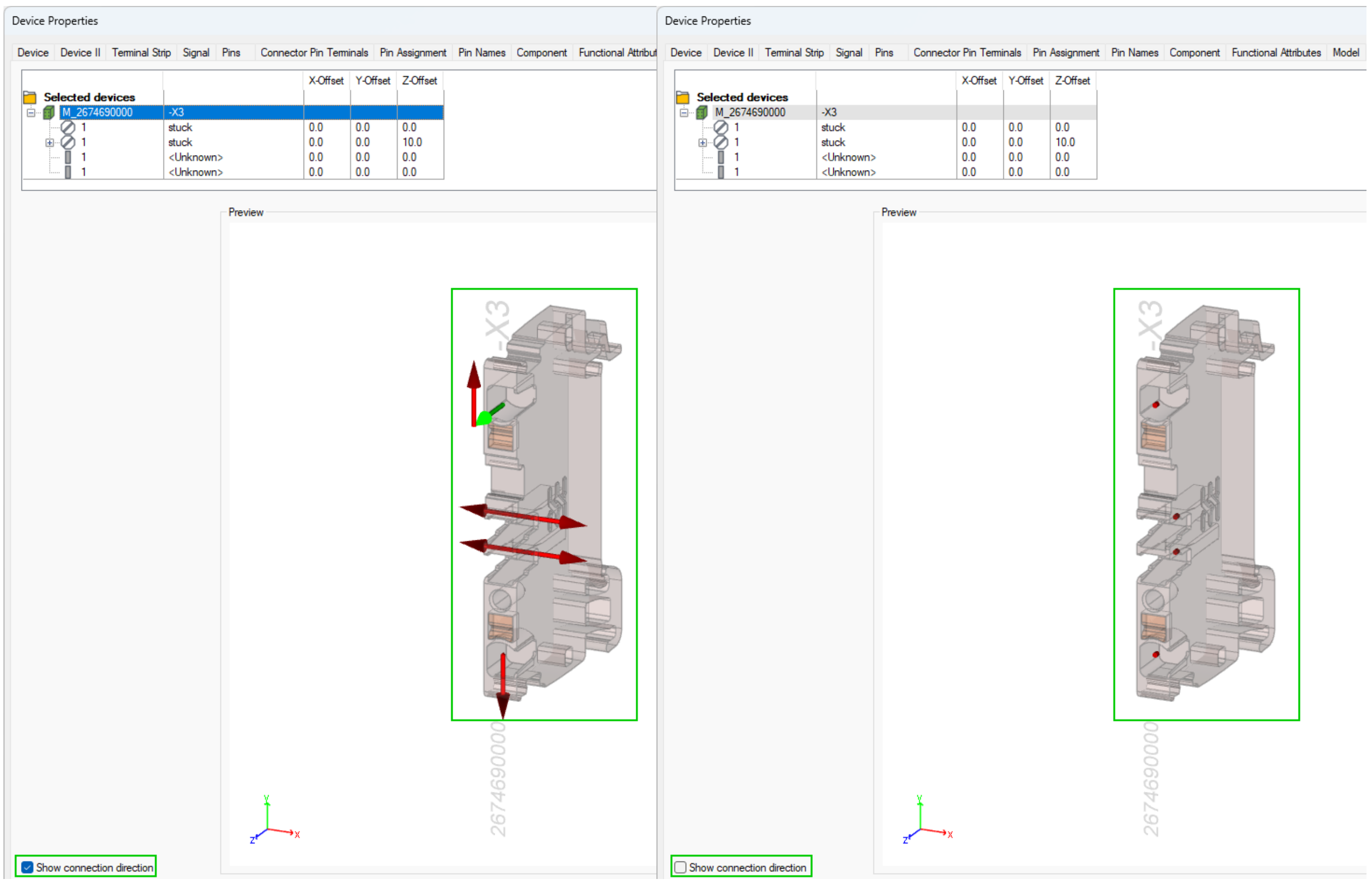
Confirm the changes with ***Apply***.
If the pin is already connected, the connection is recalculated.

**Example:**

Terminal **-X3** in the following illustration has a routing offset on the connected pin.
Compared to the connection of the second terminal, which is also highlighted in the illustration, the wire takes a "detour" and is connected in an "arc":



The setting ***Show connection direction*** controls how connection directions and routing offsets can be displayed in the preview:


Device Properties
Device | Device II | Terminal Strip | Signal | Pins | Connector Pin Terminals | Pin Assignment | Pin Names | Component | Functional Attribut
Preview
-X3
2674690000
☑ Show connection direction

| | | X-Offset | Y-Offset | Z-Offset |
|---|---|---|---|---|
| Selected devices | | | | |
| M_2674690000 | -X3 | | | |
| 1 | stuck | 0.0 | 0.0 | 0.0 |
| 1 | stuck | 0.0 | 0.0 | 10.0 |
| 1 | <Unknown> | 0.0 | 0.0 | 0.0 |
| 1 | <Unknown> | 0.0 | 0.0 | 0.0 |

Device Properties
Device | Device II | Terminal Strip | Signal | Pins | Connector Pin Terminals | Pin Assignment | Pin Names | Component | Functional Attributes | Model
Preview
-X3
2674690000
☐ Show connection direction

| | | X-Offset | Y-Offset | Z-Offset |
|---|---|---|---|---|
| Selected devices | | | | |
| M_2674690000 | -X3 | | | |
| 1 | stuck | 0.0 | 0.0 | 0.0 |
| 1 | stuck | 0.0 | 0.0 | 10.0 |
| 1 | <Unknown> | 0.0 | 0.0 | 0.0 |
| 1 | <Unknown> | 0.0 | 0.0 | 0.0 |

The connection direction is displayed as follows:

- The connection direction ***automatic*** is represented by a horizontal line whose arrowhead points in the direction in which a connection point is sought.
  If the pin is above the center of the model, a cable duct is searched for above the pin.
  If the pin is below the center of the model, a cable duct is searched for below the pin.
- The connection direction ***right*** is displayed by a horizontal line with an arrowhead to the right.
- The connection direction ***left*** is displayed by a horizontal line with an arrowhead to the left.
- The connection direction ***above*** is displayed by a vertical line with an arrowhead pointing upward.
- The connection direction ***below*** is displayed by a vertical line with an arrowhead pointing downward.
- The connection direction ***horizontal*** is represented by a horizontal line with an arrowhead to the left and right.
- The connection direction ***vertical*** is displayed by a vertical line with an arrowhead pointing upward and downward.
- The connection direction ***all*** is displayed by a cross with an arrowhead to the left, right, above and below.

Depending on the selected slots and the setting ***Show connection direction***, the connection directions and the routing offset are displayed differently:

- If the **setting is active** and **no slot is selected**, the connection directions and the routing offset of all pins are displayed.
  Connection directions are displayed in red and routing offsets in green.
- If the **setting is active** and **slots are selected**, the display depends on whether the respective slot is selected or not.
  For **slots, which are selected,** the connection directions and routing offset are displayed in red.
  For **slots, which are not selected,** the connection directions are displayed in red and the routing offset in green.
- If the **setting is inactive**, only the connection directions and routing offset of the selected pins are displayed.
  Connection directions and routing offset are displayed in red.

**Reference:** `Designer-17483`

## Define Slots of the Type *Area* with Polygons

Slots of the type ***Area*** can be defined with polygons.

This allows slots to be created that are more compatible with non-rectangular devices.

Collision checks for slots can be turned on and off with the setting ***Panel* → *Checks* → *Placement* → *Complete Component* → *Slot Area/Line***.
If the check is active, devices can only be placed on polygonal slots if the device lies completely

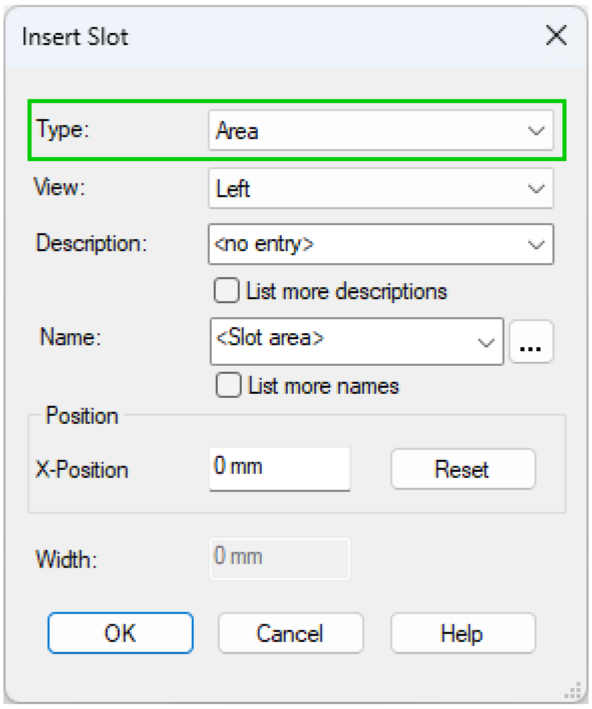
within the area of the polygon.

**Note:** Unlike slot areas that are defined by a rectangle, devices that are placed on polygonal slot areas are not automatically dragged completely into the area of the slot.

To define **new** slots using polygons, select ***Tools* → *Start Database Editor*** in the main menu.
The database editor opens.

Create a new model or edit an existing one and select ***Insert* → *Slot...*** in the main menu.
The ***Insert Slot*** dialog opens.

Specify the slot type ***Area*** in the dialog, select all additional properties of the slot and then confirm with ***OK***:


Insert Slot

Type: Area

View: Left

Description: <no entry>

List more descriptions

Name: <Slot area>

List more names

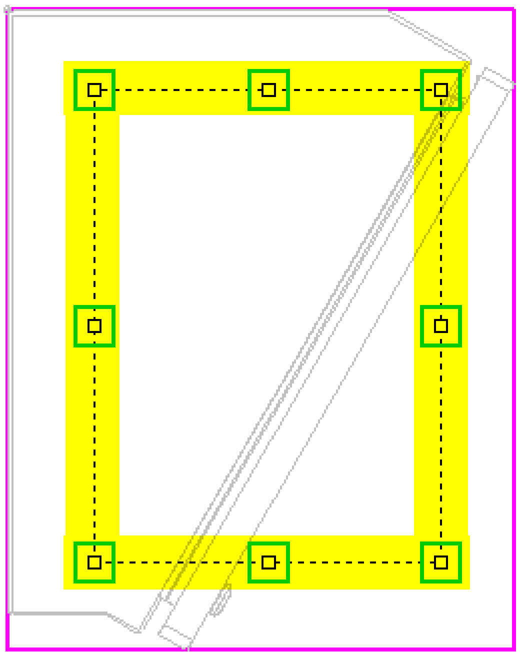
Position

X-Position 0 mm

Width: 0 mm

Then hold down the left mouse button and draw a rectangle in the workspace with the approximate size that the slot should have at the end.
**Note:** The slot is always defined as a rectangle first. When subsequently converting it to a polygon, the position, size and form of the final slot can be adjusted.

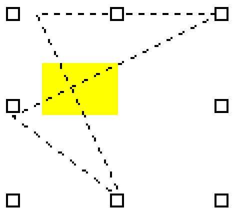
Then select the slot in the workspace or in the tree view, open the context menu while the mouse is positioned over one of the corner points or one of the boundary lines of the rectangle and select ***Edit points***.
The editing mode is activated:



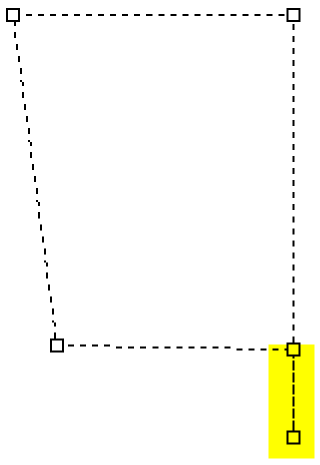
Move points of the polygon in editing mode to adjust the shape of the slot as desired.
To add additional points to the polygon, open the context menu while the mouse is positioned over one of the corner points and select ***Add point***.

To be able to save the slot after editing, the polygon must fulfill the following conditions:

- the polygon must be closed
- the polygon must not have any overlapping lines



- the polygon must consist of at least 3 points
- the polygon must describe an area and must not contain any corner points that are lines from the area



To change **existing** slots of the type ***Area*** to a polygonal slot, select ***Tools*** → ***Start Database Editor*** in the main menu.
The database editor opens.

Select a model, whose slots you want to change.

Then select the slots in the workspace or in the tree view and **edit the corner points of the rect-angle**.

## Assign and Change Slot Names

Names can be assigned to slots, which are displayed in tree views and properties dialogs.

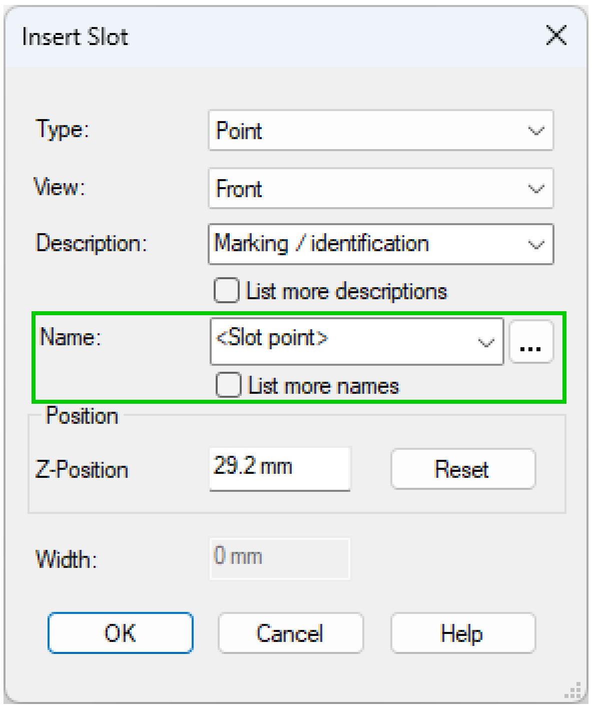
This allows you to give slots names, for example, that describe the position or purpose of the slot even more precisely, making it quicker and easier to identify where the slot is located or what it is used for.

---

To assign names to **new** slots, select ***Tools → Start Database Editor*** in the main menu.
The database editor opens.

Create a new model or edit an existing one and select ***Insert → Slot...*** in the main menu.
The ***Insert Slot*** dialog opens.

Specify the name of the slot, which is displayed in the device properties, tool tip and optionally in the device tree:


Insert Slot

Type: Point
View: Front
Description: Marking / identification
☐ List more descriptions
Name: <Slot point>
☐ List more names
Position
Z-Position: 29.2 mm
Width: 0 mm

If no individual name is assigned, the slot type in angle brackets is used as the name, ***<Slot point>***, ***<Slot line>*** or ***<Slot area>***.
If the slot name is deleted, the name is correspondingly reset to its type ***<Slot point>***, ***<Slot line>*** or ***<Slot area>***.

The name may be a maximum of 252 characters long.
The following characters are invalid in slot names: **{}<>,'**:

Activate the check box ***List more names*** to list all slot names, which are available in the database.

---

To assign names to **existing** slots, select ***Tools → Start Database Editor*** in the main menu.
The database editor opens.

**Note:** Slot names are only displayed in project mode and cannot be assigned or changed.

Select a model, whose slots you want to assign names.
Slot names can be changed in the model properties or in the workspace tree of the database editor.
To do so, press the **F2** key or click on an already selected slot to open a field for editing the name.

---

To use translatable texts from the ***Translation Table*** for slot names, enter the translation code of the desired text, for example ***&#3;***, or click on ... in the editing field and select the desired text in the ***Translation Table*** dialog.
The translation code is only translated in the first project language set.

For translated texts, which have a description in the translation table, the format is ***&Number;***, for example **&5045;** for **Device added**.
For translated texts, which have **no** description in the translation table, the format is ***&#Number;***, for example **&#3;** for **Cover panel**.
The translation code is only translated in the first project language set.

## Tables

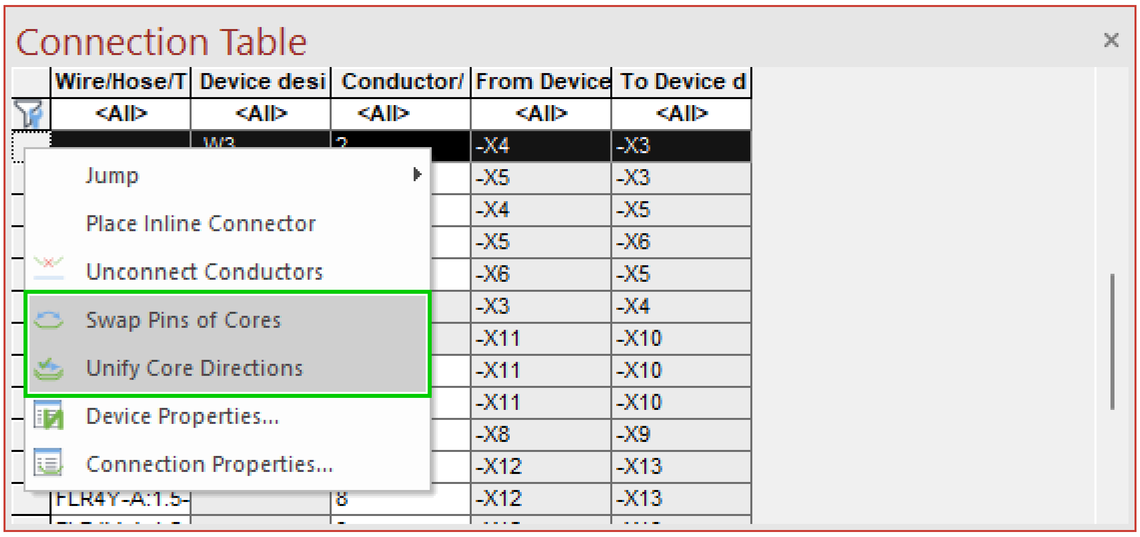
### Swap Core Ends in Connection Table and Unify Core Ends in Cables

For cores in connections, the core ends can be swapped in the connection table.

This allows the connection information in the table view to be unified and displayed more clearly.
The unification can be used, for example, for optimized work steps in production or commissioning.

To swap core ends in the connection table, select ***View → Connection Table*** in the main menu.
The connection table opens.

Right-click on one or more rows and select the context menu command ***Swap Pins of Cores*** or ***Unify Core Directions***.
The ***Unify Core Directions*** command is only available if exactly one core is selected:

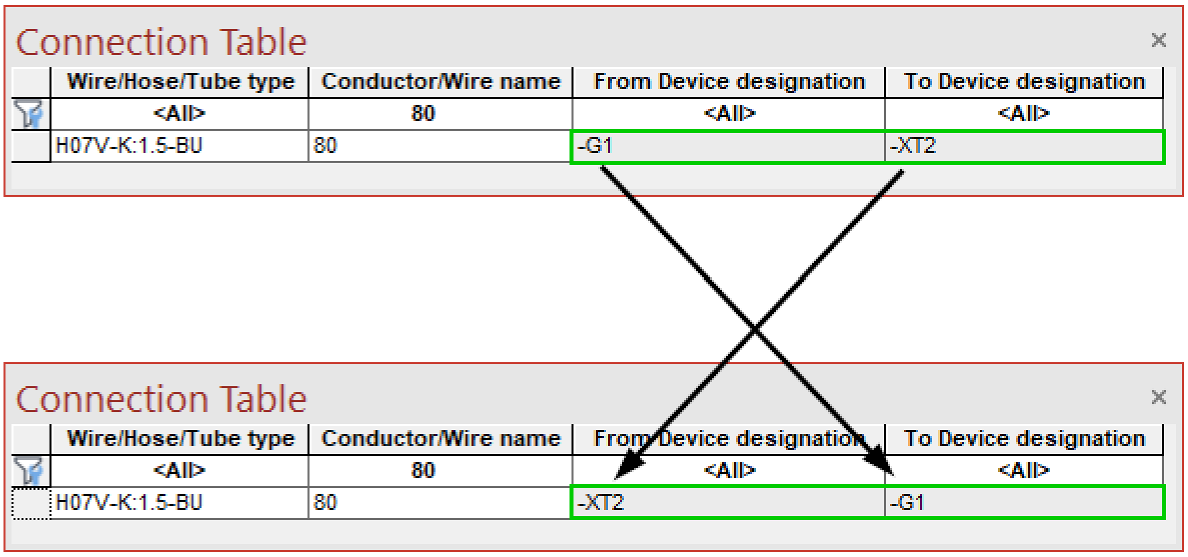


| Wire/Hose/T | Device desi | Conductor/ | From Device | To Device d |
|---|---|---|---|---|
| <All> | <All> | <All> | <All> | <All> |
|  | W3 | 2 | -X4 | -X3 |
|  |  |  | -X5 | -X3 |
|  |  |  | -X4 | -X5 |
|  |  |  | -X5 | -X6 |
|  |  |  | -X6 | -X5 |
|  |  |  | -X3 | -X4 |
|  |  |  | -X11 | -X10 |
|  |  |  | -X11 | -X10 |
|  |  |  | -X11 | -X10 |
|  |  |  | -X8 | -X9 |
|  |  |  | -X12 | -X13 |
| FLR4Y-A:1.5- |  | 8 | -X12 | -X13 |

The command is not available for the following cores/wires:

- The selected core or the selected wire is locked indirectly , for example, via the connected device, or directly.
- The core or wire is locked with the option ***Wire/bundle connectivity*** under ***Device Properties → Cables/Conductors***.
- Cores, which are defined as fixed core connections in assemblies.

The ***Swap cores*** command swaps ***End 1*** and ***End 2*** of the core:



Connection Table

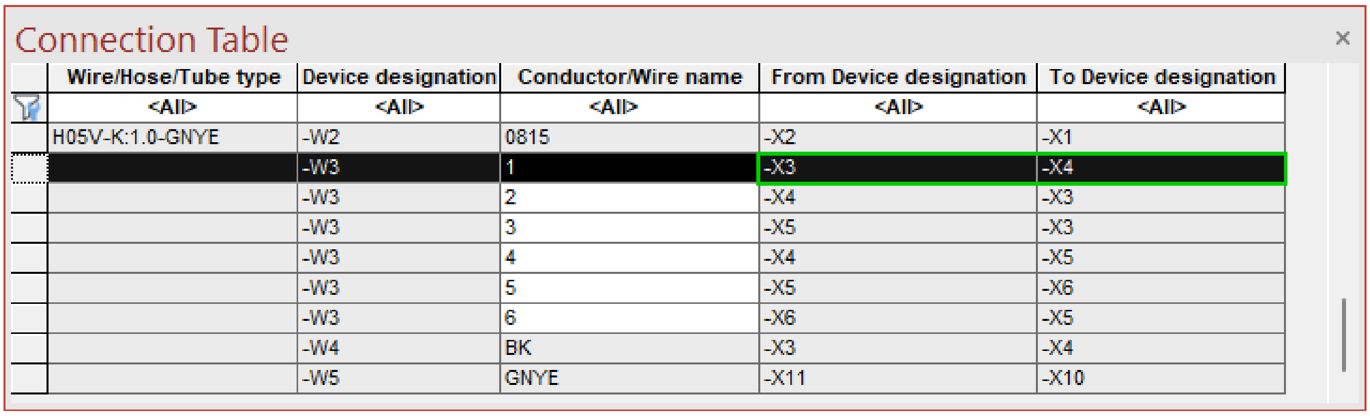
| Wire/Hose/Tube type | Conductor/Wire name | From Device designation | To Device designation |
|---|---|---|---|
| <All> | 80 | <All> | <All> |
| H07V-K:1.5-BU | 80 | -G1 | -XT2 |

Connection Table

| Wire/Hose/Tube type | Conductor/Wire name | From Device designation | To Device designation |
|---|---|---|---|
| <All> | 80 | <All> | <All> |
| H07V-K:1.5-BU | 80 | -XT2 | -G1 |

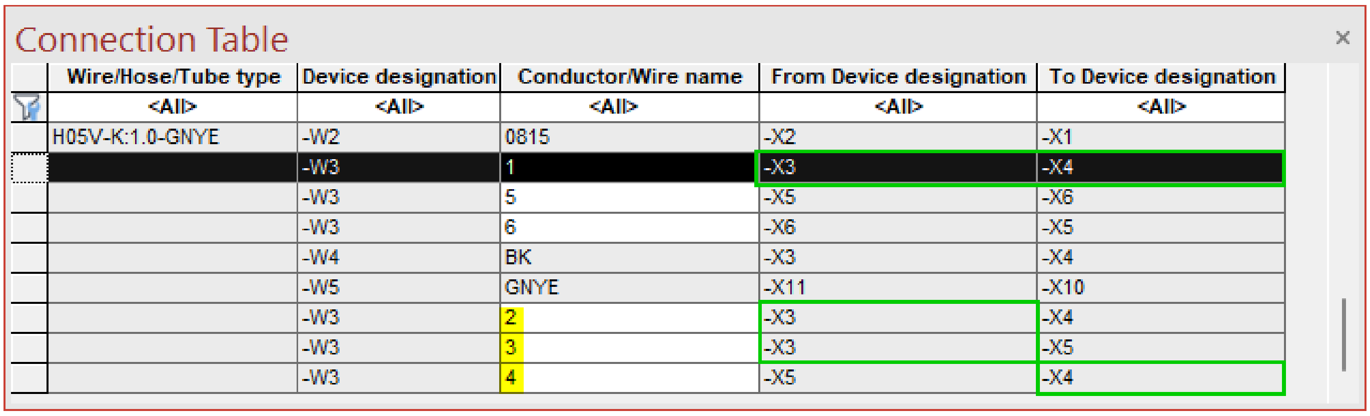
With the ***Unify Core Directions*** command, the ends of all cores of the cable are unified. The unification is implemented according to the following logic:

**Example:**



Connection Table

| Wire/Hose/Tube type | Device designation | Conductor/Wire name | From Device designation | To Device designation |
|---|---|---|---|---|
| <All> | <All> | <All> | <All> | <All> |
| H05V-K:1.0-GNYE | -W2 | 0815 | -X2 | -X1 |
|  | -W3 | 1 | -X3 | -X4 |
|  | -W3 | 2 | -X4 | -X3 |
|  | -W3 | 3 | -X5 | -X3 |
|  | -W3 | 4 | -X4 | -X5 |
|  | -W3 | 5 | -X5 | -X6 |
|  | -W3 | 6 | -X6 | -X5 |
|  | -W4 | BK | -X3 | -X4 |
|  | -W5 | GNYE | -X11 | -X10 |



Connection Table

| Wire/Hose/Tube type | Device designation | Conductor/Wire name | From Device designation | To Device designation |
|---|---|---|---|---|
| <All> | <All> | <All> | <All> | <All> |
| H05V-K:1.0-GNYE | -W2 | 0815 | -X2 | -X1 |
|  | -W3 | 1 | -X3 | -X4 |
|  | -W3 | 5 | -X5 | -X6 |
|  | -W3 | 6 | -X6 | -X5 |
|  | -W4 | BK | -X3 | -X4 |
|  | -W5 | GNYE | -X11 | -X10 |
|  | -W3 | 2 | -X3 | -X4 |
|  | -W3 | 3 | -X3 | -X5 |
|  | -W3 | 4 | -X5 | -X4 |

1. The device (***-X3*** in the example) that is connected to ***End 1*** of a core (***Core 1*** in the example) is determined.

2. All other cores of the cable that are connected to a pin of the same device (***-X3***) via ***End 2*** are swapped.
   For the affected cores, ***End 1*** is connected to the device (***-X3***).
3. The device (**-X4** in the example), which is connected via ***End 2*** of a core (***Core 1*** in the example), is determined.
4. All other cores of the cable that are connected to a pin of the same device ***(-X4)*** via ***End 1*** are swapped.
   For the affected cores, ***End 1*** is connected to the device ***(-X4)***.
5. Cores whose ends cannot be swapped are displayed in the output window.

**Reference:** `Designer-41464`

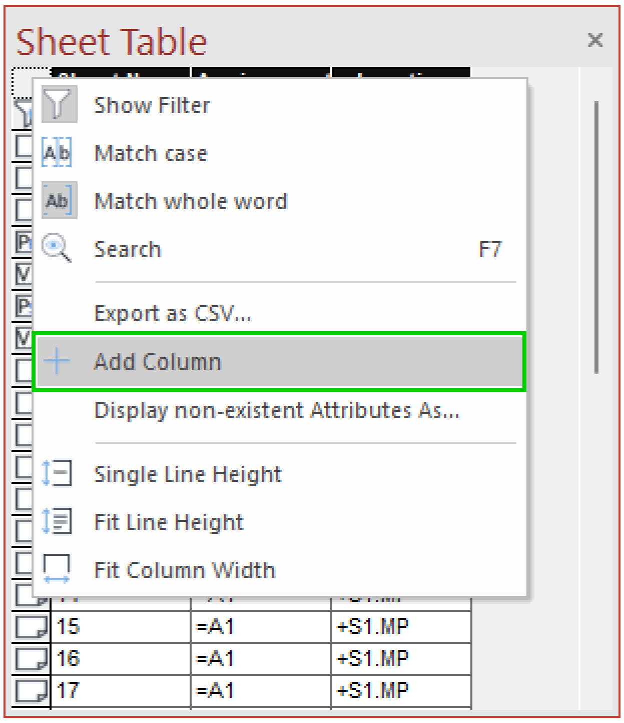
## Display Sheet Format in Sheet Table

The sheet formats can be displayed in the sheet table.

This means, for example, that the search function in the sheet table can be used to search for sheets with a specific sheet format and then rename them.

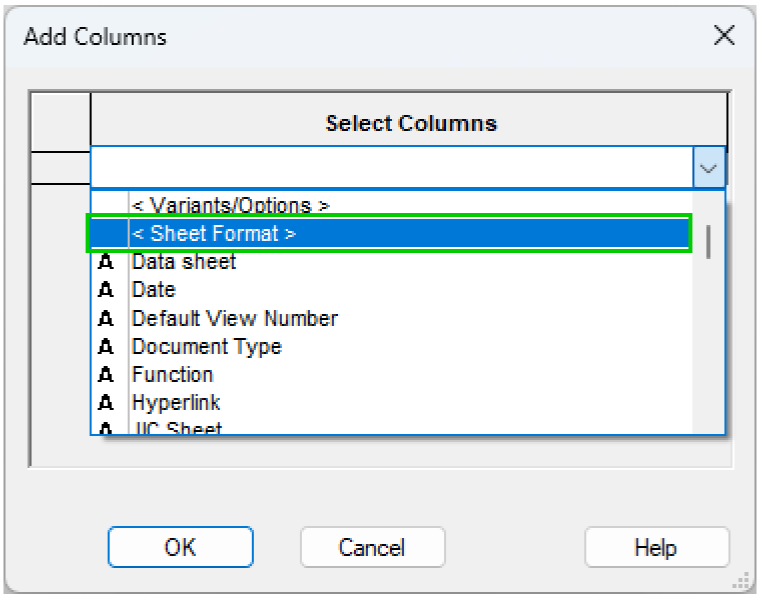
To expand the sheet table, select ***View → Sheet Table*** in the main menu.
The sheet table opens.

Right-click on a header and select the context menu command ***Add Column***:



The ***Add Columns*** dialog opens.

Click on an empty row and select ***Sheet Format*** in the drop-down menu.
Repeat the process for all additional columns to be added:



The desired columns are added to the table.
The enhancement can be used to search for and rename sheet formats in the table.
Sheet characteristics are indicated in angle brackets at the end of the sheet format, for example, ***DINA3<_JIC>***.

**Reference:** `Designer-17857`

## Character Encoding Changed when Exporting as CSV from Tables

When exporting CSV files from tables, the character encoding has been changed to **UTF-8** with **Byte Order Mark (BOM)**.

This fixes a compatibility problem with Excel that sometimes causes line breaks to lead to undesired results.

Please note the changed character encoding if you process CSV files with programs other than Excel.

**Reference:** `Designer-40657`

## Variants/Options

- Display Reference Texts for Inactive Options

### Display Reference Texts for Inactive Options

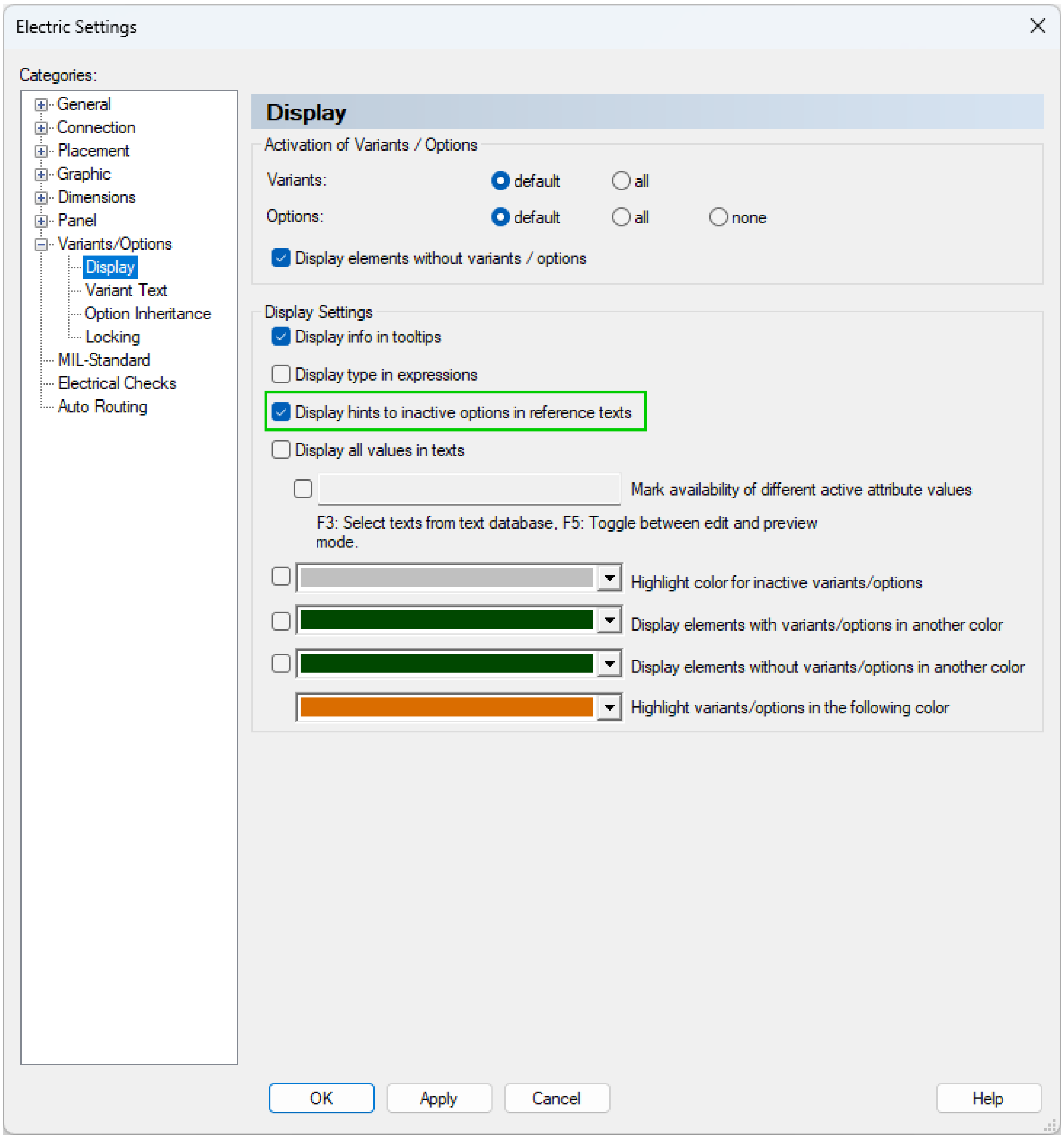
A setting can be used to control whether reference texts are displayed when they refer to inactive objects.

This allows each user to individually define the behavior of reference texts in relation to variants/options.
Starting with ***E³.*****series** Version 2025 reference texts are displayed that refer to inactive objects. The reference to inactive objects is identified by the addition of ***(Optional in: ...)***.

To make changes to the setting for reference texts, select ***Tools → Settings...*** in the main menu. The ***Electric Settings*** dialog opens.

**Note:** The setting can be made via the ***Electric Settings*** and ***Fluid Settings*** dialogs.

Deactivate or reactivate the setting ***Display hints to inactive options in reference texts*** under ***Variants/Options → Display***.
The setting is user-specific:


Electric Settings

Categories:
- General
- Connection
- Placement
- Graphic
- Dimensions
- Panel
- Variants/Options
  - Display
  - Variant Text
  - Option Inheritance
  - Locking
- MIL-Standard
- Electrical Checks
- Auto Routing

**Display**

Activation of Variants / Options

Variants: [✓ default] all

Options: [✓ default] all none

[✓] Display elements without variants / options

Display Settings

[✓] Display info in tooltips

[ ] Display type in expressions

[✓] Display hints to inactive options in reference texts

[ ] Display all values in texts

[ ] Mark availability of different active attribute values

F3: Select texts from text database, F5: Toggle between edit and preview mode.

[ ] Highlight color for inactive variants/options

[ ] Display elements with variants/options in another color

[ ] Display elements without variants/options in another color

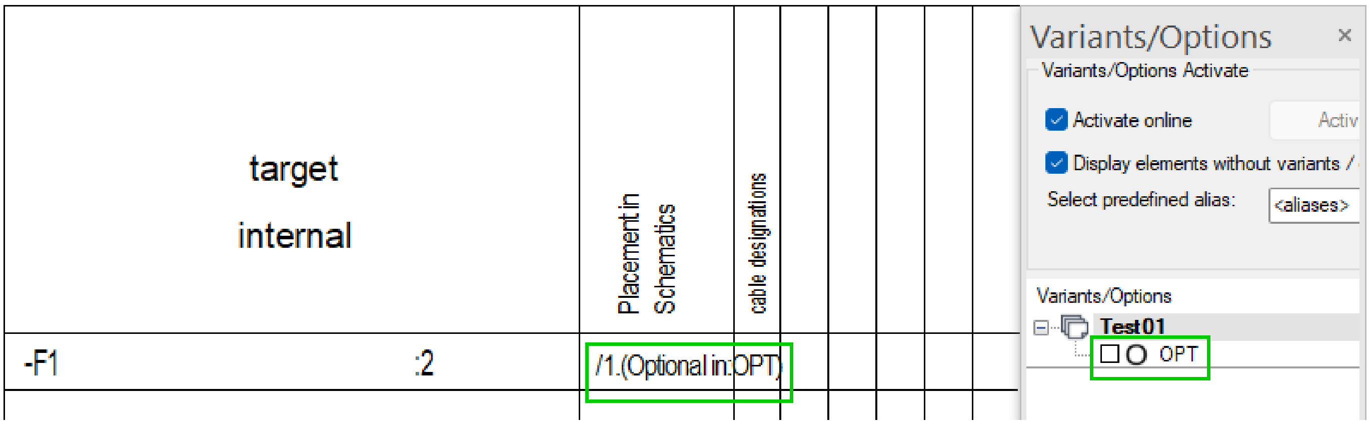
Highlight variants/options in the following color

OK Apply Cancel Help

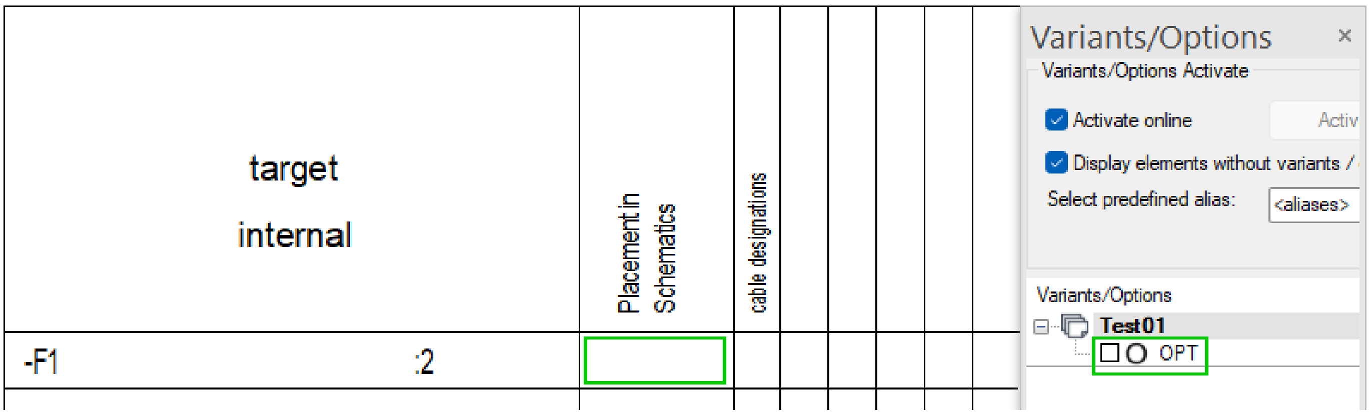
If the setting is inactive, reference texts are not displayed if targets to which they refer are not available with the respective active variants/options.

Example:

**Reference to inactive object with active setting**



**Reference to inactive object with inactive setting**



## Connection/Busses, Signals/Logic Lines, Supply

### Use Correction Value for Conductor/Wire Length for Connections to Pins

Attributes can be defined for pins that have a correction value for the calculated conductor/wire length.

This allows the values for conductor/wire lengths to be corrected, which are based on ideal routing paths when routing or in block connectors with interchangeable connector inserts, for example, which are almost impossible to implement in practice.

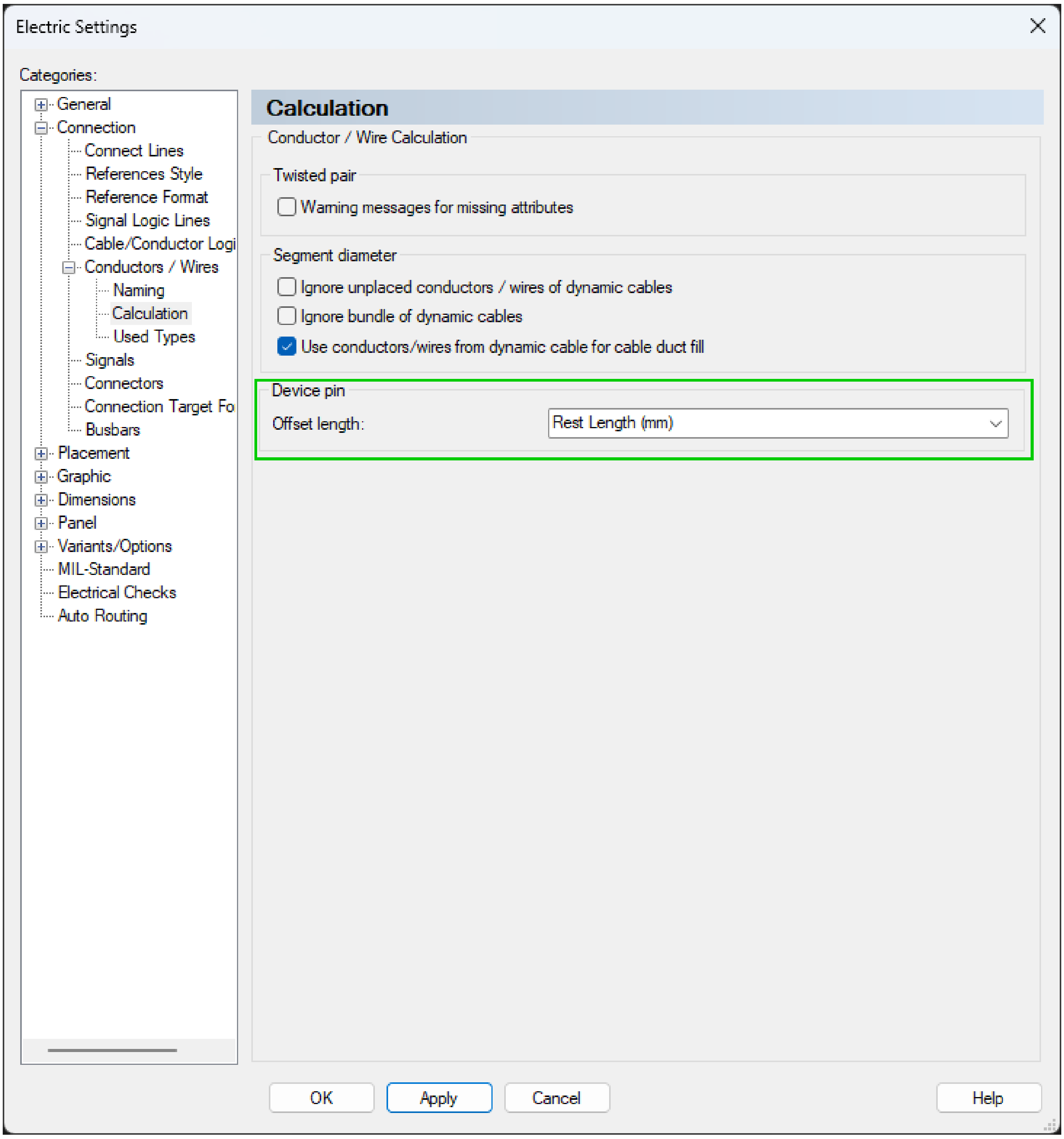
To use an attribute with a correction value for conductor/wire lengths, open a project in ***E³.*****series**, in which pins are used, for which a correction value is to be used.

Assign an attribute to the pins that have the type ***Linear measure*** and the characteristic ***Single instance***.
The attribute must also have one of the following owners:

- ***Component pin***
- ***Connector pin***
- ***Block pin***
- ***Device pin***

Select ***Tools*** → ***Settings...*** in the main menu.
The ***Electric Settings*** dialog opens.

Under ***Connection*** → ***Conductors/Wires*** → ***Calculation*** → ***Offset Length*** set the desired attribute that has the correction value for the length calculation.
The setting is project-specific:


Electric Settings

Categories:

- General
- Connection
  - Connect Lines
  - References Style
  - Reference Format
  - Signal Logic Lines
  - Cable/Conductor Logi
  - Conductors / Wires
    - Naming
    - Calculation
    - Used Types
  - Signals
  - Connectors
  - Connection Target Fo
  - Busbars
- Placement
- Graphic
- Dimensions
- Panel
- Variants/Options
- MIL-Standard
- Electrical Checks
- Auto Routing

**Calculation**

Conductor / Wire Calculation

Twisted pair

- [ ] Warning messages for missing attributes

Segment diameter

- [ ] Ignore unplaced conductors / wires of dynamic cables
- [ ] Ignore bundle of dynamic cables
- [x] Use conductors/wires from dynamic cable for cable duct fill

Device pin

Offset length: Rest Length (mm)

For all conductors/wires that are connected to pins with the selected attribute, the value of the selected attribute is added to the calculated routing path.

The length offset can be displayed in the connection table with the columns ***< From pin offset length >*** and ***< To pin offset length >***.

**Reference:** `Designer-19830`

## New Information in Device Properties "Connector Pin Terminals" Tab

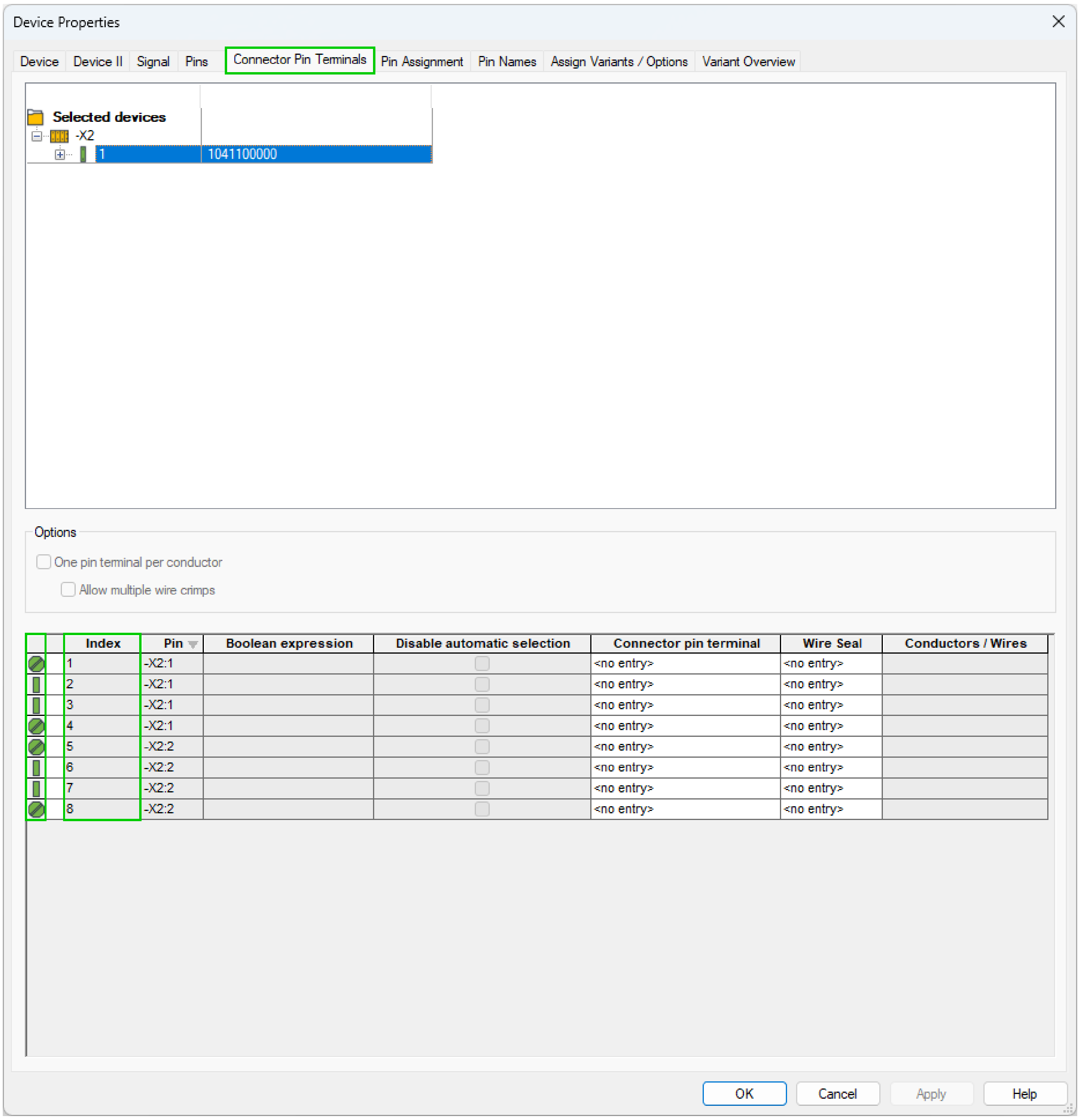
The ***Connector Pin Terminals*** tab in the Device Properties has been revised and shows the index of pins and the connection types ***screwed*** and ***clamped***.

This provides a better overview and makes the dialog even more helpful.

To have an overview of the connector pin terminals that are assigned to devices, open a project in ***E³.*****series**, in which the device with connector pin terminals is used.

Then open the context menu of the corresponding device on the sheet or in the device tree and select ***Device Properties...***
The ***Device Properties*** dialog opens.

Switch to the ***Connector Pin Terminals*** tab.
An overview of the assigned connector pin terminals of the device is displayed:



One pin terminal per conductor [ ]
Allow multiple wire crimps [ ]

| | Index | Pin | Boolean expression | Disable automatic selection | Connector pin terminal | Wire Seal | Conductors / Wires |
|---|---|---|---|---|---|---|---|
| ● | 1 | -X2:1 | | ☐ | <no entry> | <no entry> | |
| ▮ | 2 | -X2:1 | | ☐ | <no entry> | <no entry> | |
| ▮ | 3 | -X2:1 | | ☐ | <no entry> | <no entry> | |
| ● | 4 | -X2:1 | | ☐ | <no entry> | <no entry> | |
| ● | 5 | -X2:2 | | ☐ | <no entry> | <no entry> | |
| ▮ | 6 | -X2:2 | | ☐ | <no entry> | <no entry> | |
| ▮ | 7 | -X2:2 | | ☐ | <no entry> | <no entry> | |
| ● | 8 | -X2:2 | | ☐ | <no entry> | <no entry> | |

The left-hand column of the table shows whether the pin is clamped or screwed.
The following icons are used:

- ▮ stands for an **active**, **clamped** wire instance
- ▯ stands for an **inactive**, **clamped** wire instance
- ● stands for an **active**, **screwed** wire instance
- ○ stands for an **inactive**, **screwed** wire instance

The number of the respective pin is also displayed in the column ***Index***.

## Unify the Routing Direction of Newly Routed Cable Conductors with regard to the Devices at the Conductor Ends and Optionally Not Distributing Conductors/Wires Over Equivalent Pins

The routing direction for conductors of cables that are routed on connections starting with ***E³*.series** Version 2025 is unified so that the device involved in the connection is always connected to end 1 or end 2 if possible.
**Example:** If there are no further restrictions, all conductors of a cable that are routed on a new connection between devices **-A3** and **-A4** are connected to **-A3** via end 1 and to **-A4** via end 2.

As a result, conductor ends from cables that are newly routed are uniformly connected and can be used, for example, for optimized work steps in production or commissioning.

The following rules apply for unifying the routing direction of newly routed cable conductors:

- If a device is connected to all conductors via the same end, all other conductors that are connected are aligned at the unique end.
  **Example: -A3** is connected to **End 1** of the routed conductors. Each new conductor that is connected to **-A3** is also automatically connected via **End 1**.
- If none of the ends are clearly connected across all conductors, newly routed conductors cannot be connected with a uniform routing direction.
- All conductors that were previously checked for a uniform routing direction are used for testing all conductors that are subsequently routed.
- For conductors that are connected with an open end, only the connected side is checked.

Conductors that are routed and not uniformly connected can be unified via the **Connection Table**.

The routing direction of newly routed conductors in cables is **not** generated in the unified form in the following cases:

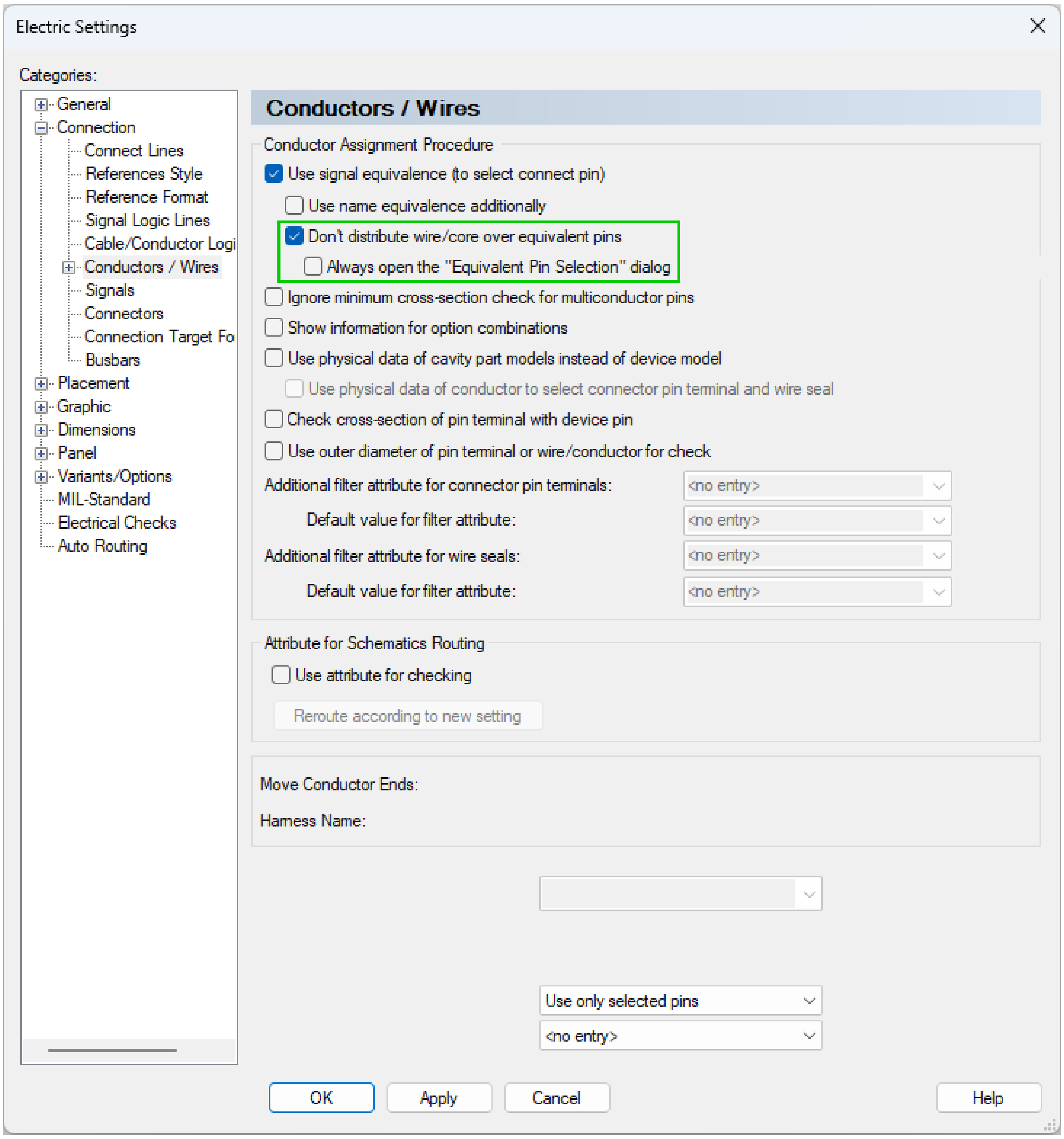
- If a conductor end is deliberately changed or connected to a device.
- If the cable has conductors that are already connected and the connection directions are not clear.
  For example, because:
  - two conductors are connected via the respective end 1 with **-A3** and end 2 with **-A4** and
  - two conductors are connected via the respective end 1 with **-A4** and end 2 with **-A3**.

---

To unify the routing direction in multiple representations, for example in the device tree and terminal plan, it can optionally be defined not to distribute conductors or wires over equivalent pins when they are connected to pins.

To prevent the distribution over equivalent pins, either use the connection mode ***Don't distribute wire/core over equivalent pins*** ( ) or first select ***Tools*** → ***Settings...*** in the main menu.
The ***Electric Settings*** dialog opens.

Then activate the setting ***Connection*** → ***Conductors / Wires*** → ***Don't distribute wire/core over equivalent pins***.
The setting is user-specific:



Electric Settings

Categories:

- General
- Connection
  - Connect Lines
  - References Style
  - Reference Format
  - Signal Logic Lines
  - Cable/Conductor Logi
  - Conductors / Wires
  - Signals
  - Connectors
  - Connection Target Fo
  - Busbars
- Placement
- Graphic
- Dimensions
- Panel
- Variants/Options
- MIL-Standard
- Electrical Checks
- Auto Routing

**Conductors / Wires**

Conductor Assignment Procedure

- [✓] Use signal equivalence (to select connect pin)
  - [ ] Use name equivalence additionally
  - [✓] Don't distribute wire/core over equivalent pins
    - [ ] Always open the "Equivalent Pin Selection" dialog
- [ ] Ignore minimum cross-section check for multiconductor pins
- [ ] Show information for option combinations
- [ ] Use physical data of cavity part models instead of device model
  - [ ] Use physical data of conductor to select connector pin terminal and wire seal
- [ ] Check cross-section of pin terminal with device pin
- [ ] Use outer diameter of pin terminal or wire/conductor for check

| | |
|---|---|
| Additional filter attribute for connector pin terminals: | <no entry> |
| Default value for filter attribute: | <no entry> |
| Additional filter attribute for wire seals: | <no entry> |
| Default value for filter attribute: | <no entry> |

Attribute for Schematics Routing

- [ ] Use attribute for checking

Reroute according to new setting

| | |
|---|---|
| Move Conductor Ends: | Use only selected pins |
| Harness Name: | <no entry> |

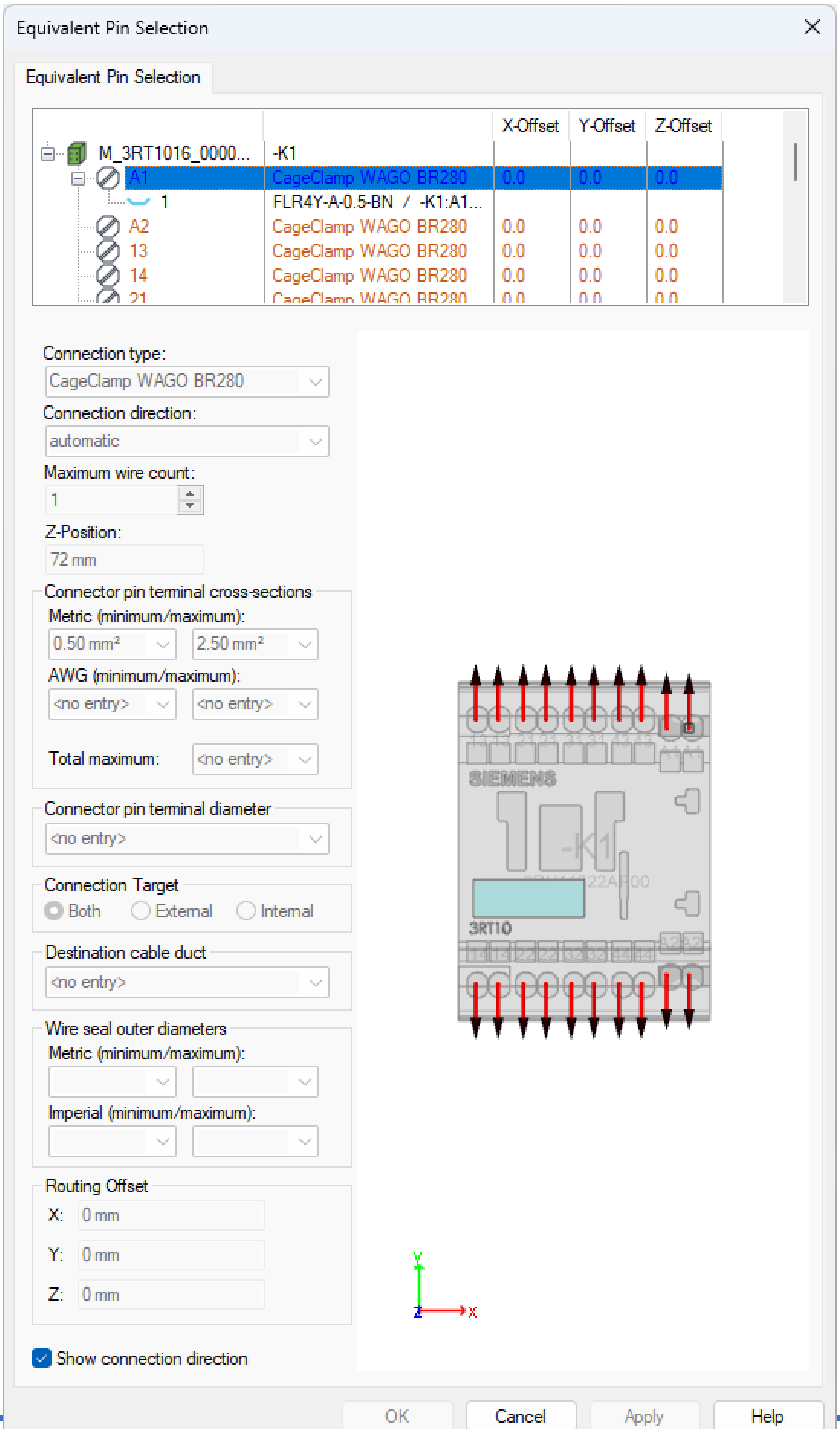
When the setting is active, conductors/cores can only be connected to pins if the pins of the respective connection are directly compatible with the conductor or wire.
In this case, condcutors/wires are not connected to equivalent pins.

Distribution of conductors/wires to equivalent pins is prevented if connections are created or modified using the following methods:

- The connection method ***Reconnect Cores*** is used.
- The connection method ***Use default wire when connection*** is used.

- The connection method ***Reconnect connection*** is used.
- The connection method ***Swap pins of connector(s)*** is used.
- The connection method ***Reorder Conductors*** is used.
- Wires from the device tree or database are connected.
- The assignment of conductor/wire ends to pins is changed using the dialog ***Wire/Core Connections***.
- The function **e3Pin.SetEndPinId ()** of the programming interface is used for the assignment of conductor/wire ends to pins.
- The command ***Repair conductor/wire and pin*** is used.
- The assignment of conductor/wire ends to pins is changed using the connection table.
- The symbol of a device is changed.
- Conductors/wires are connected using the formboard table.
- Conductors/wires are connected using the terminal plan.

If the distribution is prevented by the setting ***Don't distribute wire/core over equivalent pins*** and the conductor or wire can potentially be connected to another pin, the ***Equivalent Pin Selection*** opens:



Equivalent Pin Selection

Equivalent Pin Selection

| | | X-Offset | Y-Offset | Z-Offset |
|---|---|---|---|---|
| M_3RT1016_0000... | -K1 | | | |
| A1 | CageClamp WAGO BR280 | 0.0 | 0.0 | 0.0 |
| 1 | FLR4Y-A-0.5-BN / -K1:A1... | | | |
| A2 | CageClamp WAGO BR280 | 0.0 | 0.0 | 0.0 |
| 13 | CageClamp WAGO BR280 | 0.0 | 0.0 | 0.0 |
| 14 | CageClamp WAGO BR280 | 0.0 | 0.0 | 0.0 |
| 21 | CageClamp WAGO BR280 | 0.0 | 0.0 | 0.0 |

Connection type: CageClamp WAGO BR280

Connection direction: automatic

Maximum wire count: 1

Z-Position: 72 mm

Connector pin terminal cross-sections

Metric (minimum/maximum): 0.50 mm² 2.50 mm²

AWG (minimum/maximum): <no entry> <no entry>

Total maximum: <no entry>

Connector pin terminal diameter

<no entry>

Connection Target

[● Both] ○ External ○ Internal

Destination cable duct

<no entry>

Wire seal outer diameters

Metric (minimum/maximum):

Imperial (minimum/maximum):

Routing Offset

X: 0 mm

Y: 0 mm

Z: 0 mm

[✓] Show connection direction

**Note:** If the setting ***Always open the "Equivalent Pin Selection" dialog*** is active, the dialog always opens when a conductor or wire is connected to a pin.
If condcutors/wires are used for a connection, the terminal plan or a schema pin, the dialog does not open.

The dialog shows all pins with equivalent signals.
The pins are displayed in various colors to indicate whether the pin can be used for a connection:

- Pin name **blue**: the selected pin.
- Pin name **black**: pins to which the wire can be connected.
- Pin name **red orange**: pins that are not equivalent.
  Pin ends locked against distribution are also displayed in **red orange** in dialogs containing the tab ***Wire/Conductor Connection***.
- Pin name **gray**: equivalent pins to which the wire cannot be connected.

## North America

### USA

Zuken USA Inc.
Westford, MA 01886, USA
Tel: +1 978 692 4900

## Asia

### Japan

Zuken (Worldwide Head Office)
Yokohama, Kanagawa 224-8585, Japan
Tel: +81 45 942 1511

### China

Zuken (Shanghai) Technical Center Co., Ltd.
Room 301, No.555 Nanjing West Road,
Shanghai, 200041, People's Republic of China
Tel: +86-21-3218-1784

### Korea

Zuken Korea Inc.
Seoul 135-283, Korea
Tel: +82 2 5648031

### Singapore

Zuken Singapore Pte Ltd.
#22-05 Gateway East, Singapore 189721
Tel: +65 6392 5855

### Taiwan

Zuken Taiwan Inc.
Taipei 110, Taiwan
Tel: +886 2 7718 1116

## Europe

### Germany

Zuken GmbH (European HQ)
D-85399 Hallbergmoos, Germany
Tel: +49 89 7104059 00

Zuken E3 GmbH
D-89079 Ulm, Germany
Tel: +49 7305 9309 0

Zuken E3 GmbH
D-30659 Hannover, Germany
Tel: +49 511 8595 9489

### Switzerland

Zuken E3 GmbH
CH-5504 Othmarsingen, Switzerland
Tel: +41 62 561 08 00

### United Kingdom

Zuken UK Ltd.
Bristol, BS32 4RF, UK
Tel: +44 1454 207 801

### France

Zuken S.A.
#91974 Les Ulis Cédex, France
Tel: +33 1 69 29 48 00

### Italy

Zuken S.r.l.
20090 Milanofiori Assago, Milan, Italy
Tel: +39 02 575 921

### Netherlands

Zuken GmbH
NL-6075 HA Herkenbosch, The Netherlands
Tel: +31 475 520 998

**https://www.zuken.com/e3**

ZUKEN®

All trademarks mentioned are the property of their respective owners, Copyright © Zuken E3 GmbH