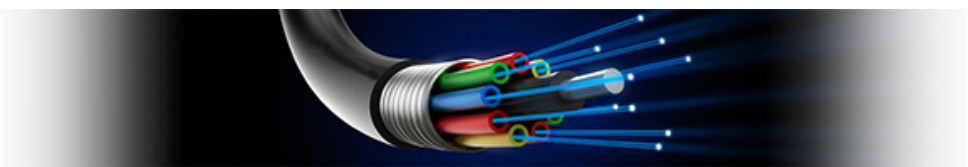
ZUKEN®

E3 series

New Features 2026-26.00



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

Table of Contents

*E³.*series Release Notes

Date: 5/8/2025

This manual describes the following functions of version:

2026 Build 26.00

!! 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.

Update Information of Version 2026

Windows Support

The following Windows systems are supported with E3.series 2026:

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 2026 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 2026, so copies should be used forE3.series 2026.

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 2026, a project can no longer be opened with an earlier version.

Multi-User Installation

Existing multi-user projects from Version 2025 can directly be opened with Version 2026. They are automatically converted upon opening. However, this cannot be undone, and the projects can no longer be opened with Version 2025.

That’s why we recommend installing Version 2026 multi-user projects in parallel to the existing version, so that it’s also possible to still access these projects with E3.series Version 2025.

To transfer the projects to Version 2026, these projects must be saved in Version 2025 as Single-User projects. They can then be stored in the multi-user environment of Version 2026.

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 2026 Build 26.00

The following functionality is new in Version 2026 Build 26.00:

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 2026 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 in *e3DbeApplication* and *e3DbeModel* - Creation of views with *CreateDXFfromSTEP* can be aborted after timeout

New values have been added to the parameter **flags** of the function **CreateDXFfromSTEP** in the ***e3DbeApplication*** interface for defining a timeout after which the entire import will be aborted or the creation of individual views skipped.
Return values have been added accordingly.

The following values were added:

Parameter:

- **flags**`[in][optional]`: The parameter supports the following new values:
  - **Timeout for aborting the entire import**
    - **0x000100** (Decimal value **256**): 1 minute
    - **0x000200** (Decimal value **512**): 2 minutes
    - **0x000400** (Decimal value **1024**): 4 minutes

- **0x000800** (Decimal value **2048**): 8 minutes
- **0x001000** (Decimal value **4096**): 16 minutes
- **0x002000** (Decimal value **8192**): 32 minutes
- **0x004000** (Decimal value **16384**): 64 minutes
- **0x008000** (Decimal value **32768**): 128 minutes

- **Timeout for skipping the creation of individual views**
  - **0x010000** (Decimal value **65536**): 1 minute
  - **0x020000** (Decimal value **131072**): 2 minutes
  - **0x040000** (Decimal value **262144**): 4 minutes
  - **0x080000** (Decimal value **524288**): 8 minutes
  - **0x100000** (Decimal value **1048576**): 16 minutes
  - **0x200000** (Decimal value **2097152**): 32 minutes
  - **0x400000** (Decimal value **4194304**): 64 minutes
  - **0x800000** (Decimal value **8388608**): 128 minutes

Return values:

- **return -9**: Failed calling the function - The import of the STEP file was aborted by the user.
- **return -10**: Failed calling the function - The import of the STEP file was aborted due to timeout.
- **return -11**: Failed calling the function - The creation of at least one view was skipped by the user.
- **return -12**: Failed calling the function - The creation of at least one view was skipped due to timeout.

New values have been added to the parameter **flags** of the function **ImportStep** in the ***e3DbeModel*** interface for defining a timeout after which the entire import will be aborted or the creation of individual views skipped.
Return values have been added accordingly.

The following values were added:

Parameter:

- **flags**`[in][optional]`: The parameter supports the following new values:

  - **Timeout for aborting the entire import**
    - **0x0001** (Decimal value **1**): 1 minute
    - **0x0002** (Decimal value **2**): 2 minutes
    - **0x0004** (Decimal value **4**): 4 minutes

- **0x0008** (Decimal value **8**): 8 minutes
- **0x0010** (Decimal value **16**): 16 minutes
- **0x0020** (Decimal value **32**): 32 minutes
- **0x0040** (Decimal value **64**): 64 minutes
- **0x0080** (Decimal value **128**): 128 minutes

- **Timeout for skipping the creation of individual views**
  - **0x0100** (Decimal value **256**): 1 minutes
  - **0x0200** (Decimal value **512**): 2 minutes
  - **0x0400** (Decimal value **1024**): 4 minutes
  - **0x0800** (Decimal value **2048**): 8 minutes
  - **0x1000** (Decimal value **4096**): 16 minutes
  - **0x2000** (Decimal value **8192**): 32 minutes
  - **0x4000** (Decimal value **16384**): 64 minutes
  - **0x8000** (Decimal value **32768**): 128 minutes

Return values:

- **return -8**: Failed calling the function - The import of the STEP file was aborted by the user.
- **return -9**: Failed calling the function - The import of the STEP file was aborted due to timeout.
- **return -10**: Failed calling the function - The creation of at least one view was skipped by the user.
- **return -11**: Failed calling the function - The creation of at least one view was skipped due to timeout.

References: `Designer-42596` und `E3-2541`

Customer request: Enhancement in *e3Job* - Highlighting Hyperlinks when jumping to them for exporting with *ExportPDF* and *ExportPDFEx* ,

A new value has been added to the **options** parameter of the **ExportPDF** function in the ***e3Job*** interface for highlighting the targets of hyperlinks in the PDF when jumping to them.

The following value has been added:

- **options** `[in]`: The parameter supports the following new values:
  - **0x100000** (Decimal value **1048576**): With this value the targets of hyperlinks are highlighted in the PDF using color and flashing when they are jumped to.
    The highlight flashes five times and then disappears.
    The option cannot be used when **options**also has the value ***0x00800*** (Decimal value **2048**) (export as PDF/A standard).

A new value has been added to the **options** parameter of the **ExportPDFEx** function in the ***e3Job*** interface for highlighting the targets of hyperlinks in the PDF when jumping to them.
Return values have been added accordingly.

The following value has been added:

Parameter:

- **options** `[in]`: The parameter supports the following new values:
  - **0x100000** (Decimal values **1048576**): With this value the targets of hyperlinks are highlighted in the PDF using color and flashing when they are jumped to.
    The highlight flashes five times and then disappears.
    The option cannot be used when **options**also has the value ***0x00800*** (Decimal value **2048**) (export as PDF/A standard).

Return values:

- **return -19**: Failed calling the function - The parameter **options** contains the incompatible values for *Export as PDF/A Standard* (**0x00800**) and *Highlight jump targets* (**0x100000**).

References: `Designer-15372` und `E3-161`

Enhancement in *e3AttributeDefinition* - New value for the *Owner* property

The ***e3AttributeDefinition*** interface has been enhanced with new values for the *Owner* property for specifying the attribute owners *Busbar* and *Busbar Type*.

The following values were added:

- **attributeDefinition = 48** corresponds to the property *Owner* with the value *Busbar type*
- **attributeDefinition = 49** corresponds to the property *Owner* with the value *Busbar*

Enhancement in *e3CavityPart* and *e3Pin* - New return value in *SetValue* and *SetFitting* when object name contains too many characters

The following function in the ***e3CavityPart*** interface supports a new return value:

return = e3CavityPart.SetValue (value)

- **return -7**: Failed calling the function - the maximum permitted number of 64 characters has been exceeded.

---

return = e3Pin.SetFitting (name)

- **return -2**: Failed calling the function - the maximum permitted number of 64 characters has been exceeded.

Enhancement in *e3Component* and *e3DbeComponent* - New return values in *GetComponentType*

The behavior of the following functions in the ***e3Component*** and ***e3DbeComponent*** interfaces has been enhanced:

return = e3Component.GetComponentType ( )

return = e3DbeComponent.GetComponentType ( )

- The functions support a return value for dynamic blocks and block connectors:
  - **Return value40**: The component is a **dynamic block**.
  - **Return value41**: The component is a **block connector**.

Enhancement in *e3DbeModel* - New value for *flags* in *ImportStep*

A new value has been added to the parameter **flags** of the **ImportStep** function in the ***e3DbeModel*** interface for using the origin of STEP models as the origin of the model in E3.series:

return = e3DbeModel.ImportStep (filename, flags)

- **flags**`[in][optional]`: The parameter supports the following new values:
  - **0x080** (Decimal values **128**): The origin of the STEP model is used as the origin of the model in E3.series.

Enhancement in *e3Job* - *ExportForeign*

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

return = e3Job.ExportForeign (format, file)

- The structure of the XML files, with which functions are exported, contain a version number of the structure used.
  The version specifies which information is contained in the XML export and how they are structured syntactically. The version number is increased if there are changes in future relases.
  With the enhancement, the version number `2` is introduced.
  The information is specified in the root element **`<E3CM>`** with the attribute **`ContentVersion`**.
  Example: **`<E3CM ContentVersion="2">`**

Enhancement in *e3Outline* - New return value in *SetUseCutOutGraphic* and *SetUseInE3CutOut* if no compatible license is used

The following functions in the ***e3AttributeDefinition*** interface support a new return value:

return = e3Outline.SetUseCutOutGraphic (set)

- **return -7**: Failed calling the function - no compatible license used. The function cannot be executed using E3.viewPlus or E3.redliner.

return = e3Outline.SetUseInE3CutOut (set)

- **return -6**: Failed calling the function - no compatible license used. The function cannot be executed using E3.viewPlus or E3.redliner.

Enhancements in *e3Sheet* - New return value in *DisplayEx*

The following function in the ***e3Sheet*** interface supports a new return value:

return = e3Sheet.DisplayEx (flags)

- **return -5**: Failed calling the function -Displaying the sheet with 2D views in the ***3D Panel World*** mode not possible.

Enhancements in *e3Symbol* - New return values in *GetSymbolType* for MIL symbols

The following function in the***e3Symbol*** interface supports new return values:

return = e3Symbol.GetSymbolType ( )

- **return 45**: Successfully called the function - The selected symbol is of the type *MIL block connector*.
- **return 46**: Successfully called the function - The selected symbol is of the type *MIL connector*.
- **return 47**: Successfully called the function - The selected symbol is of the type *MIL feed-through connector*.

COM-Interface - New Interfaces and Functions

This chapter provides an overview of which interfaces and functions that are new in E3.series 2026.

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 function in in *e3Pin* - *IsUnmeasured*

The following function was added to the ***e3Pin*** interface for querying whether wires, conductors or hoses/pipes are unmeasured:

return = e3Pin.IsUnmeasured (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 selected object is unmeasured.
- **return 0**: Successfully called the function - The selected object is not unmeasured.

- **return -1**: Failed calling the function - no function is open or no **e3Pin** object is selected.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the selected object type is invalid.
- **return -4**: Failed calling the function - the selected object has differing routing paths that are in part measured and in part unmeasured.

References: `Designer-32473` and `E3-2817`

Customer request: New function in *e3Slot* - *SetMountType*

The following function was added to the ***e3Slot*** interface for changing slot descriptions the project:

return = e3Slot.SetMountType (newval, flags)

Parameters:

- **newval** `[in]`: The slot description that is assigned.
- **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 - slot description was changed.
- **return -1**: Failed calling the function - no slot selected or no project opened.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the selected slot ID is invalid.
- **return -4**: Failed calling the function - the selected device is locked.
- **return -5**: Failed calling the function - no slot description specified.
- **return -6**: Failed calling the function - slot description is invalid.

References: `Designer-37804` and `E3-3599`

Customer request: new functions in *e3DbeApplication* - *Save* and *SaveAs*

The following functions has been added to the ***e3DbeApplication*** interface for saving existing projects to the database editor:

return = e3DbeApplication.Save (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 - project was saved.
- **return -1**: Failed calling the function - no project open.
- **return -2**: Failed calling the function - the value defined for **flags** is invalid.
- **return -3**: Failed calling the function - project cannot be saved, as it is read-only.
- **return -4**: Failed calling the function - an internal error occurred.

References: `Designer-34436`

The following functions has been added to the ***e3DbeApplication*** interface for saving new projects to the database editor:

return = e3DbeApplication.SaveAs (name, compressed, flags)

Parameters:

- **name** `[in]`: File name including format and path that is used to save the file.
- **compressed** `[in][optional]`: An optional parameter with which the function can be enhanced. The parameter defines whether the project is saved compressed.
  The parameter supports the following values:
  - **false**: The project is saved uncompressed.
    If **compressed** is not used, the project is saved uncompressed by default.
  - **true**: The project is saved compressed.
- **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 - project was saved.
- **return -1**: Failed calling the function - no project opened.
- **return -2**: Failed calling the function - the value defined for **flags** is invalid.
- **return -3**: Failed calling the function - project cannot be saved as a template file (***.e3t** format) because a component, a symbol or a model is being edited.
- **return -4**: Failed calling the function - an internal error occurred.

References: `Designer-34436`

Customer request: New functions in in *e3Job* - *ExportRedlinerInformation* and *ImportRedlinerInformation*

The following functions were added to the ***e3Job*** interface for exporting information from Redliner files in the ***.e3r**format.
The command is only available with a valid E3.redliner license:

return = e3Job.ExportRedlinerInformation (fileName, flags)

Parameters:

- **fileName** `[in]`: The parameter defines the path and file name of the Redliner file in the ***.e3r**format.
- **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 information was exported successfully from the Redliner file.
- **return -1**: Failed calling the function - no project open.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the function is only available with a valid E3.redliner license.
- **return -4**: Failed calling the function - the format of the selected file is invalid and has to be in the t ***.e3r** format.
- **return -5**: Failed calling the function - an error occurred while accessing file **fileName**.
- **return -6**: Failed calling the function - an error occurred while exporting.

References: `Designer-13422` and `E3-3654`

The following function was added to the ***e3Job*** interface for importing information from Redliner files in the ***.e3r**format.
The information of the Redliner files cannot be imported when the module E3.redliner, E3.view or E3.viewPlus is used:

return = e3Job.ImportRedlinerInformation (fileName, flags)

Parameters:

- **fileName** `[in]`: The parameter defines the path and file name of the Redliner file in the ***.e3r**format.
- **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 information was imported successfully from the Redliner file.
- **return -1**: Failed calling the function - no project open.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.

- **return -3**: Failed calling the function - Redliner data cannot be imported when the module E3.redliner, E3.view or E3.viewPlus is used.
- **return -4**: Failed calling the function - the format of the selected file is invalid and has to be in the t ***.e3r** format.
- **return -5**: Failed calling the function - an error occurred while accessing file **fileName**.
- **return -6**: Failed calling the function - the file **fileName** is corrupt.
- **return -7**: Failed calling the function - an error occurred while importing.

References: `Designer-13422` and `E3-3654`

New function in *e3Outline - SetCreatesThreadedHole*

The following functions have been added to the ***e3Outline*** interface for assigning the property ***Creates threaded hole*** to contours of the type ***Drill-hole***. This causes a threaded hole to be created on the reference object of the model upon manufacture:

return = e3Outline.SetCreatesThreadedHole (set, flags)

Parameters:

- **set** `[in]`: The parameter defines whether the property ***Creates threaded hole*** is assigned to the contour.
  The properties ***Creates threaded hole*** and ***Use alternative graphic***, which is assigned using the **e3Outline.SetUseCutOutGraphic (set)** function, cannot be active at the same time.

  The property ***Creates threaded hole*** can be queried with the function **e3Outline.IsThreadedHole ()**.

  The parameter supports the following values:

  - **true**: The contour causes a threaded hole on the reference object of the model upon manufacture.
    The property ***Use alternative graphic*** is automatically deactivated.
  - **false**:The contour **does not** cause a threaded hole on the reference object of the model upon manufacture.
- **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 property was successfully activated or deactivated.
- **return -1**: Failed calling the function - no project open.
- **return -2**: Failed calling the function - no object selected.
- **return -3**: Failed calling the function - the selected object is not a contour.
- **return -4**: Failed calling the function - the selected contour is not of the type ***Drill-hole***.

- **return -5**: Failed calling the function - the selected device is locked and can therefore not be edited.
- **return -6**: Failed calling the function - the drill hole was created via the device properties in the ***Drill Hole Definition*** tab and cannot be changed using this function.
- **return -7**: Failed calling the function - the selected object is a contour of the component and cannot be edited using this function.
- **return -8**: Failed calling the function - used license is incompatible. The function cannot be executed using E3.viewPlus or E3.redliner.
- **return -9**: Failed calling the function - the value defined for **flags** is invalid.

New functions in *e3DbeModel* - *GetJustificationArea*, *GetJustificationLine*, *GetJustificationPoint*, *GetMountingDescriptions*, *SetJustificationArea*, *SetJustificationLine*, *SetJustificationPoint* and *SetMountingDescriptions*

The following functions were added to the ***e3DbeModel*** interface for querying the justification defined for models on slots of the type ***Area***:

return = e3DbeModel.GetJustificationArea (z, flags)

Parameters:

- **z** `[out]`: Defines the justification on slots in direction of the X-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 - the parameter **z** specifies the justification on slot of the type ***Area***.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - no model selected.
- **return -3**: Failed calling the function - the value specified for **flags** is invalid.

The following functions were added to the ***e3DbeModel*** interface for querying the justification defined for models on slots of the type ***Line***:

return = e3DbeModel.GetJustificationLine (y, z, flags)

Parameters:

- **y** `[out]`: Defines the justification on slots in direction of the Y-axis.
- **z** `[out]`: Defines the justification on slots in 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 - the parameters **y** and **z** specify the justification in slots of the type ***Line***.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - no model selected.
- **return -3**: Failed calling the function - the value specified for **flags** is invalid.

The following functions were added to the ***e3DbeModel*** interface for querying the justification defined for models on slots of the type ***Point***:

return = e3DbeModel.GetJustificationPoint (x, y, z, flags)

Parameters:

- **x** `[out]`: Defines the justification on slots in direction of the X-axis.
- **y** `[out]`: Defines the justification on slots in direction of the Y-axis.
- **z** `[out]`: Defines the justification on slots in 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 - the parameters **x**, **y** and **z** specify the justification on slots of the type ***Point***.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - no model selected.
- **return -3**: Failed calling the function - the value specified for the **flags** is invalid.

The following function was added to the ***e3DbeModel*** interface for querying the mounting description of models:

return = e3DbeModel.GetMountingDescriptions (mountingdescriptions, flags)

Parameters:

- **mountingdescriptions** `[out]`: An array with the mounting description of 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 amount of mounting descriptions defined for the model.
- **return -1**: Failed calling the function - no project open.
- **return -2**: Failed calling the function - no model selected.
- **return -3**: Failed calling the function - the value specified for **flags** is invalid.

The following function was added to the ***e3DbeModel*** interface for defining the justification of models on slots of the type ***Area***:

return = e3DbeModel.SetJustificationArea (z, flags)

Parameters:

- **z** `[in]`: Sets the justification on slots in 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 - the value for the justification has been set.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - no model selected.
- **return -3**: Failed calling the function - the value specified for **flags** is invalid.
- **return -4**: Failed calling the function - the value for the justification is invalid, because it exceeds the minimal or maximal range.

The following function was added to the ***e3DbeModel*** interface for defining the justification of models on slots of the type ***Line***:

return = e3DbeModel.SetJustificationLine (y, z, flags)

Parameters:

- **y** `[in]`: Sets the justification on slots in direction of the Y-axis.
- **z** `[in]`: Sets the justification on slots in 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 - the values for the justification has been set.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no open project.
- **return -2**: Failed calling the function - no model selected.

- **return -3**: Failed calling the function - the value specified for **flags** is invalid.
- **return -4**: Failed calling the function - at least one of the values for the justification is invalid, because it exceeds the minimal or maximal range.

The following function was added to the ***e3DbeModel*** interface for defining the justification of models on slots of the type ***Point***:

return = e3DbeModel.SetJustificationPoint (x, y, z, flags)

Parameters:

- **x** `[in]`:Sets the justification on slots in direction of the X-axis.
- **y** `[in]`: Sets the justification on slots in direction of the Y-axis.
- **z** `[in]`: Sets the justification on slots in 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 - the values for the justification has been set.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no open project.
- **return -2**: Failed calling the function - no model selected.
- **return -3**: Failed calling the function - the value specified for **flags** is invalid.
- **return -4**: Failed calling the function - at least one of the values for the justification is invalid, because it exceeds the minimal or maximal range.

The following function was added to the ***e3DbeModel*** interface for defining the mounting description of models:

return = e3DbeModel.SetMountingDescriptions (mountingdescriptions, flags)

Parameters:

- **mountingdescriptions** `[in]`: An array with the mounting descriptions that are assigned. If the array is empty or an empty string (`""`) is given the parameter, the mounting descriptions are deleted.
- **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 mounting descriptions were assigned.
- **return -1**: Failed calling the function - no project open.

- **return -2**: Failed calling the function - no model selected.
- **return -3**: Failed calling the function - the value specified for **flags** is invalid.

New functions in *e3DbeSlot* - *CreateAreaPolygon*, *CreateAreaRectangle*, *CreateLine*, *CreatePoint*, *Delete*, *GetDirectionRotation*, *SetDirectionRotation* and *SetRotation*

The following functions were added to the ***e3DbeSlot*** interface for creating polygonal slots of the type ***Area*** in the Database Editor:

return = e3DbeSlot.CreateAreaPolygon (modelid, xarr, yarr, zarr, xdir, ydir, zdir, description, name, flags)

Parameters:

- **modelid** `[in]`: The ID of the model for which the slot is created.
- The parameters **xarr**, **yarr** and **zarr** define the coordinates of the individual points of the polygon.
  Each array needs to have the same number of coordinates and describe at least four points.
  The first and last value of each array need to be identical in order to close the polygon.
- **xarr** `[in]`: An array with the x coordinates of the individual points of the slot.
- **yarr** `[in]`: An array with the y coordinates of the individual points of the slot.
- **zarr** `[in]`:An array with the z coordinates of the individual points of the slot.
  Note: The coordinates of the first point of the polygon are created using the first value of **xarr**, **yarr** and **zarr**.
  The coordinates of the second point are created using the second value of **xarr**, **yarr**and **zarr** and so on.
- The direction of the slot is described using the following parameters. The direction can only be defined by one parameter at a time:
  - **xdir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Right*** or ***Left***.
    With the value **1** the slot is created in with the direction ***Right***.
    With the value **-1** the slot is created in with the direction ***Left***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **ydir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Up*** or ***Down***.
    With the value **1** the slot is created in with the direction ***Up***.
    With the value **-1** the slot is created in with the direction ***Down***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **zdir** `[in]`:This parameter can be used to control whether the slot is placed with the direction ***Front*** or ***Back***.
    With the value **1** the slot is created in with the direction ***Front***.
    With the value **-1** the slot is created in with the direction ***Back***.
    With the value **0**the slot is created in with the direction of a different parameter.

- **Example:**
  A slot with the direction ***Right*** is created using **xdir = 1**, **ydir = 0** and **zdir = 0**.
  No slot is created using **xdir = 1**, **ydir = 1** and **zdir = 0** because both **xdir** and **ydir** have a value for a direction. Only one parameter can define the direction at a time.

- **description** `[in][optional]`:Description of the slot.
- **name** `[in][optional]`: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 ≥ 0**: Successfully called the function - the ID of the slot, that was created, is output.
- **return -1**: Failed calling the function - no valid model selected.
- **return -2**: Failed calling the function - the value specified for **flags**is invalid.
- **return -3**: Failed calling the function - the definition of the direction is invalid.
- **return -4**: Failed calling the function - the parameters **xarr**, **yarr** and **zarr** have different amounts of values.
- **return -5**: Failed calling the function - the polygon has less then four points.
- **return -6**: Failed calling the function - the points of the polygon do not describe a closed area.
- **return -7**: Failed calling the function - the points of the polygon do not lie on one level.
- **return -8**: Failed calling the function - the form of the polygon is invalid.
- **return -9**: Failed calling the function - the slot could not be created.

The following functions were added to the ***e3DbeSlot*** interface for creating rectangular slots of the type ***Area*** in the Database Editor:

return = e3DbeSlot.CreateAreaRectangle (modelid, xpos, ypos, zpos, xdir, ydir, zdir, length, width, description, name, flags)

Parameters:

- **modelid** `[in]`: ID of the model for which the slot is created.
- **xpos** `[in]`: The x coordinate that is used as the origin for the creation of the slot.
- **ypos** `[in]`: The z coordinate that is used as the origin for the creation of the slot.
- **zpos** `[in]`: The y coordinate that is used as the origin for the creation of the slot.
- The direction of the slot is described using the following parameters. The direction can only be defined by one parameter at a time:

- **xdir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Right*** or ***Left***.
  With the value **1**the slot is created in with the direction ***Right***.
  With the value **-1**the slot is created in with the direction***Left***.
  With the value **0**the slot is created in with the direction of a different parameter.
- **ydir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Up*** or ***Down***.
  With the value **1** the slot is created in with the direction ***Up***.
  With the value **-1** the slot is created in with the direction ***Down***.
  With the value **0**the slot is created in with the direction of a different parameter.
- **zdir** `[in]`:This parameter can be used to control whether the slot is placed with the direction ***Front*** or ***Back***.
  With the value **1** the slot is created in with the direction ***Front***.
  With the value **-1** the slot is created in with the direction ***Back***.
  With the value **0**the slot is created in with the direction of a different parameter.
- **Example:**
  A slot with the direction ***Right*** is created using **xdir = 1**, **ydir = 0** and **zdir = 0**.
  No slot is created using **xdir = 1**, **ydir = 1** and **zdir = 0** because both **xdir** and **ydir** have a value for a direction. Only one parameter can define the direction at a time.

- **length** `[in]`: Length of the slot.
- **width** `[in]`: Width of the slot.
- **description** `[in]`: Description of the slot.
- **name** `[in][optional]`: 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 ≥ 0**: Successfully called the function - ID of the slot, that was created, is output.
- **return -1**: Failed calling the function - no valid model selected.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the definition of the direction is invalid.
- **return -4**: Failed calling the function - no slot description given.
- **return -5**: Failed calling the function - the value for **length** is invalid.
- **return -6**: Failed calling the function - the value for **width** is invalid.
- **return -7**: Failed calling the function - the slot could not be created.

The following functions were added to the ***e3DbeSlot*** interface for creating slots of the type ***Line*** in the Database Editor:

return = e3DbeSlot.CreateLine (modelid, xpos, ypos, zpos, xdir, ydir, zdir, length, width, rotation, description, name, flags)

Parameters:

- **modelid** `[in]`: ID of the model for which the slot is created.
- **xpos** `[in]`: The x coordinate that is used as the origin for the creation of the slot.
- **ypos** `[in]`: The z coordinate that is used as the origin for the creation of the slot.
- **zpos** `[in]`: The y coordinate that is used as the origin for the creation of the slot.
- The direction of the slot is described using the following parameters. The direction can only be defined by one parameter at a time:
  - **xdir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Right*** or ***Left***.
    With the value **1**the slot is created in with the direction ***Right***.
    With the value **-1**the slot is created in with the direction***Left***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **ydir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Up*** or ***Down***.
    With the value **1** the slot is created in with the direction ***Up***.
    With the value **-1** the slot is created in with the direction ***Down***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **zdir** `[in]`:This parameter can be used to control whether the slot is placed with the direction ***Front*** or ***Back***.
    With the value **1** the slot is created in with the direction ***Front***.
    With the value **-1** the slot is created in with the direction ***Back***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **Example:**
    A slot with the direction ***Right*** is created using **xdir = 1**, **ydir = 0** and **zdir = 0**.
    No slot is created using **xdir = 1**, **ydir = 1** and **zdir = 0** because both **xdir** and **ydir** have a value for a direction. Only one parameter can define the direction at a time.
- **length** `[in]`: Length of the slot.
- **width** `[in]`: Width of the slot.
- **rotation** `[in]`: Rotation of the slot.
- **description** `[in]`: Description of the slot.
- **name** `[in][optional]`: 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 ≥ 0**: Successfully called the function - ID of the slot, that was created, is output.
- **return -1**: Failed calling the function - no valid model selected.

- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the definition of the direction is invalid.
- **return -4**: Failed calling the function - no slot description given.
- **return -5**: Failed calling the function - the value for **length** is invalid.
- **return -6**: Failed calling the function - the slot could not be created.

The following functions were added to the ***e3DbeSlot*** interface for creating slots of the type ***Point*** in the Database Editor:

return = e3DbeSlot.CreatePoint (modelid, xpos, ypos, zpos, xdir, ydir, zdir, rotation, description, name, flags)

Parameters:

- **modelid** `[in]`: ID of the model for which the slot is created.
- **xpos** `[in]`: The x coordinate that is used as the origin for the creation of the slot.
- **ypos** `[in]`: The z coordinate that is used as the origin for the creation of the slot.
- **zpos** `[in]`: The y coordinate that is used as the origin for the creation of the slot.
- The direction of the slot is described using the following parameters. The direction can only be defined by one parameter at a time:
  - **xdir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Right*** or ***Left***.
    With the value **1**the slot is created in with the direction ***Right***.
    With the value **-1**the slot is created in with the direction***Left***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **ydir** `[in]`: This parameter can be used to control whether the slot is placed with the direction ***Up*** or ***Down***.
    With the value **1** the slot is created in with the direction ***Up***.
    With the value **-1** the slot is created in with the direction ***Down***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **zdir** `[in]`:This parameter can be used to control whether the slot is placed with the direction ***Front*** or ***Back***.
    With the value **1** the slot is created in with the direction ***Front***.
    With the value **-1** the slot is created in with the direction ***Back***.
    With the value **0**the slot is created in with the direction of a different parameter.
  - **Example:**
    A slot with the direction ***Right*** is created using **xdir = 1**, **ydir = 0** and **zdir = 0**.
    No slot is created using **xdir = 1**, **ydir = 1** and **zdir = 0** because both **xdir** and **ydir** have a value for a direction. Only one parameter can define the direction at a time.
- **rotation** `[in]`: Rotation of the slot.
- **description** `[in]`: Description of the slot.

- **name** `[in][optional]`: 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 ≥ 0**: Successfully called the function - ID of the slot, that was created, is output.
- **return -1**: Failed calling the function - no valid model selected.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the definition of the direction is invalid.
- **return -4**: Failed calling the function - no slot description given.
- **return -5**: Failed calling the function - the slot could not be created.

The following functions were added to the ***e3DbeSlot*** interface for deleting slots in the Database Editor:

return = e3DbeSlot.Delete (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 selected slot was deleted.
- **return -1**: Failed calling the function - no valid slot selected.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the selected object is the slot of a mounting rail or cable duct.
- **return -4**: Failed calling the function - an internal error occurred.

The following functions were added to the ***e3DbeSlot*** interface for querying the rotation of slots in the Database Editor:

return = e3DbeSlot.GetDirectionRotation (xAxisRotation, yAxisRotation, zAxisRotation, flags)

Parameters:

- **xAxisRotation** `[out]`: Rotation of the slot around the X-axis.
- **yAxisRotation** `[out]`: Rotation of the slot around the Y-axis.
- **zAxisRotation** `[out]`: Rotation of the slot around 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 - the rotations are output with **xAxisRotation**, **yAxisRotation** and **zAxisRotation**.
- **return -1**: Failed calling the function - no slot selected or no project open.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.

The following functions were added to the ***e3DbeSlot*** interface for defining the rotation of slots in the Database Editor:

return = e3DbeSlot.SetDirectionRotation (xAxisRotation, yAxisRotation, zAxisRotation, flags)

Parameter:

- **xAxisRotation** `[in]`: Rotation of the slot around the X-axis.
  Slots of the direction ***Right*** or ***Left*** cannot be rotated around the X-axis.
- **yAxisRotation** `[in]`: Rotation of the slot around the Y-axis.
  Slots of the direction ***Top*** or ***Bottom*** cannot be rotated around the Y-axis.
- **zAxisRotation** `[in]`: of the slot around the Z-axis.
  Slots of the direction ***Front*** or ***Back*** cannot be rotated around 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 - the rotation values of the slot were set.
- **return -1**: Failed calling the function - no slot selected or no project open.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the slot has the direction ***Front*** or ***Back*** and cannot be rotated around the Z-axis.
- **return -4**: Failed calling the function - the slot has the direction ***Right*** or ***Left*** and cannot be rotated around the X-axis.
- **return -5**: Failed calling the function - the slot has the direction ***Top*** or ***Bottom*** and cannot be rotated around the Y-axis.

The following functions were added to the ***e3DbeSlot*** interface for defining the rotation of plugged devices in slots of the type ***Line*** or ***Point*** in the Database Editor:

return = e3DbeSlot.SetRotation (rotation, flags)

Parameter:

- **rotation** `[in]`: The rotation plugged devices will get in the slot. For slots of the type ***Line*** only the rotation values **-180**, **0** and**+180** are supported.
- **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 - devices, that are plugged on the selected slot, are rotated by the following value.
- **return -1**: Failed calling the function - no valid slot selected.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the selected slot is of the type ***Area***.
- **return -4**: Failed calling the function - the selected slot is of the type ***Line*** and only supports the rotation values **-180°**, **0°** and **+180°**.
- **return -5**: Failed calling the function - an internal error occurred.

New function in *e3Device* - *GetCableDuctWireAndCoreIds*, *GetConnectedCableDuctIds*, *SetJustificationArea*, *SetJustificationLine* and *SetJustificationPoint*

The following function has been added to the ***e3Device*** interface for querying the IDs of conductors and wires that are routed through a selected cable duct:

return = e3Device.GetCableDuctWireAndCoreIds (coreIds)

Parameter:

- **coreIds** `[out]`: The IDs of the conductors and wires that are routed through the selected cable duct.

Return values:

- **return ≥ 0**: Successfully called the function - the number of conductors and wires routed through the cable duct is displayed.
- **return -1**: Failed calling the function - no project is open.
- **return -2**: Failed calling the function - the selected object is not a device.
- **return -3**: Failed calling the function - the selected device is not a cable duct.
- **return -4**: Failed calling the function - the selected cable duct is not placed.
- **return -5**: Failed calling the function - an internal error occurred.

The following function has been added to the ***e3Device*** interface for querying the IDs of cable ducts connected to a selected cable duct:

return = e3Device.GetConnectedCableDuctIds (Ids, flags)

Parameter:

- **Ids** `[out]`: The IDs of the cable ducts that are connected to the selected cable duct.
  Note: If the selected or one of the connected cable ducts has variant positions and all variants/options are active at the same time, the output can contain faulty information.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether the query for connected cable ducts is done recursively and therefore also finds those cable ducts that are not directly connected to the selected cable duct but indirectly via another cable duct.
  Example: The cable ducts **-U1**, **-U2** and **-U3** are connected in a straight line (**-U1** with **-U2** and **-U2** with **-U3**). The function queries for cable ducts connected to **-U1** .
  If the query is recursive, the IDs of the cable ducts **-U2** and **-U3** are displayed.
  If the query is **not** recursive, only the ID of the cable duct **-U2** is displayed.

  Additionally, the parameter can be used to define what is considered when displaying connected cable ducts that are defined as cable duct inlets/outlets.

  The parameter can be comprised of the following values:

  - **0x0001** (Decimal value**1**): The query for connected cable ducts is done recursively and displays those cable ducts that are connected **directly or indirectly** to the selected cable duct.
    If the cable ducts form a closed circle, the ID of each cable duct is displayed only once.
    If **flags** is not used, the search is not done recursively.
  - **0x0002** (Decimal value **2**): When querying for cable ducts that are connected to the selected cable duct and are defined as inlets or outlets, the cable duct inlet and outlet are treated as two connected cable ducts.
    If **flags** is not used or does not contain the value **0x0002**, cable duct inlets and outlets are treated as a single cable ducts.

Return values:

- **return ≥ 0**: Successfully called the function - the number of cable ducts connected to the selected cable duct is displayed.
- **return -1**: Failed calling the function - no project is open.
- **return -2**: Failed calling the function - the selected object is not a device.
- **return -3**: Failed calling the function - the selected device is not a cable duct.
- **return -4**: Failed calling the function - the selected cable duct is not placed.
- **return -5**: Failed calling the function - the value selected for **flags** is invalid.
- **return -6**: Failed calling the function - an internal error occurred.

The following functions have been added to the ***e3Device*** interface for changing the justification on slots of the type ***Area***:

return = e3Device.SetJustificationArea (z, flags)

Parameter:

- **z** `[in]`: Sets the justification on slots in 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 - the value for the justification was set.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - the selected device has no model.
- **return -3**: Failed calling the function - the device is locked or the project was opened with E3.view.
- **return -4**: Failed calling the function - the model of the device is locked.
- **return -5**: Failed calling the function - the justification cannot be changed if all variants in the project are active and the device is placed or has variant positions.
- **return -6**: Failed calling the function - the value for the justification is invalid because it does not lie within the maximum or minimum range of the area.
- **return -7**: Failed calling the function - the values could not be changed.
- **return -8**: Failed calling the function - devices placed on the model are locked.

The following function was added to the ***e3Device*** interface for changing the justification on slots of the type ***Line***:

return = e3Device.SetJustificationLine (y, z, flags)

- **y** `[in]`: Sets the justification on slots in direction of the Y-axis.
- **z** `[in]`: Sets the justification on slots in 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 - the values for the justification were set.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - the selected device has no model.
- **return -3**: Failed calling the function - the device is locked or the project was opened with E3.view.
- **return -4**: Failed calling the function - the model of the device is locked.

- **return -5**: Failed calling the function - the justification cannot be changed if all variants in the project are active and the device is placed or has variant positions.
- **return -6**: Failed calling the function - at least one of the values for the justification is invalid because it does not lie within the maximum or minimum range of the area.
- **return -7**: Failed calling the function - the values could not be changed.
- **return -8**: Failed calling the function - devices placed on the model are locked.

The following function was added to the ***e3Device*** interface for changing the justification on slots of the type ***Point***:

return = e3Device.SetJustificationPoint (x, y, z, flags)

- **x** `[in]`: Sets the justification on slots in direction of the X-axis.
- **y** `[in]`: Sets the justification on slots in direction of the Y-axis.
- **z** `[in]`: Sets the justification on slots in 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 - the values for the justification were set.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - the selected device has no model.
- **return -3**: Failed calling the function - the device is locked or the project was opened with E3.view.
- **return -4**: Failed calling the function - the model of the device is locked.
- **return -5**: Failed calling the function - the justification cannot be changed if all variants in the project are active and the device is placed or has variant positions.
- **return -6**: Failed calling the function - at least one of the values for the justification is invalid because it does not lie within the maximum or minimum range of the area.
- **return -7**: Failed calling the function - the values could not be changed.
- **return -8**: Failed calling the function - devices placed on the model are locked.

New function in *e3Group* - *GetGUID* and *SetGUID*

The following function has been added to the ***e3Group*** interface for querying the GUID (*Globally Unique Identifier*) of group objects:

return = e3Group.GetGUID ( )

Return values:

- **return "<GUID>"**: Successfully called the function - the function specifies the GUID of the selected group.
- **return "Empty"**: Failed calling the function - the selected object is not a group.

The following function has been added to the ***e3Group*** interface for assigning a GUID (*Globally Unique Identifier*) to group objects:

return = e3Group.SetGUID (guid)

Parameter:

- **guid** `[in]`: Specifies the GUID that is assigned to a group.

Return values:

- **return "<GUID>"**: Successfully called the function - the GUID was assigned to the selected group.
- **return "Empty"**: Failed calling the function - the selected object is not a group.

New functions in *e3Job - FocusOnIDs*, *GetLockedObjects*, *HasLockedObjects* and *UpdateAllComponentsAttributes*

The following function has been added to the ***e3Job*** interface for selecting objects based on their ID and jumping to the position of the object on the sheets:

return = e3Job.FocusOnIDs (focusIds, highlight, highlightColour)

Parameter:

- **focusIds** `[in]`: The IDs of objects that are selected.
- **highlight** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether the selected objects are highlighted.
  The parameter supports the following values:
  - **false**: The selected objects are not highlighted.
    If **highlight** is not used, objects are not highlighted by default.
  - **true**: The selected objects are highlighted.
    If the setting ***Keep existing highlights when jumping*** is not active and objects were previously highlighted in the user interface, for example due to an object search, the existing highlights are removed.
    If no color is specified in the **highlightColour** parameter, the color set under ***General → Highlight*** is used.
- **highlightColour** `[in] [optional]`: An optional parameter with which the function can be enhanced.
  The parameter specifies the color in which the objects are highlighted.

The parameter supports values between **-1** (automatic) and **255** and correspond to the colors defined in the color table.

Return values:

- **return True**: Inconclusive - the function was executed for the selected object IDs.
  It is not possible to conclude from the return value whether the selection via the ID was successful for all objects.
- **return False**: Failed calling the function - no project open.

The following function was added to the ***e3Job*** interface for querying the number of locked objects in the project:

return = e3Job.GetLockedObjects (Ids, flags)

Parameters:

- **Ids** `[out]`: Outputs the IDs of the locked objects.
- **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 number of locked objects in the project.
- **return -1**: Failed calling the function - no project open.

The following function was added to the ***e3Job*** interface for querying whether there are locked objects in the project:

return = e3Job.HasLockedObjects (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 - there are no locked objects in the project.
- **return -1**: Successfully called the function - there is at least one locked object in the project.

The following function has been added to the ***e3Job*** interface for updating attributes of components used in the project:

return = e3Job.UpdateAllComponentsAttributes ( )

Return values:

- **return > 0**: Successfully called the function - the number of components that were updated is displayed.
- **return 0**: Inconclusive - no components were updated or no project is open.
- **return -1**: Failed calling the function - an internal error occurred.

New functions in *e3Pin* - *GetComponentPinId* and *GetOriginalCoreName*

The following function was added to the ***e3Pin*** interface for querying which identifier the component pin of a given pin of a device, connection or net segment has:

return = e3Pin.GetComponentPinId (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 identifier of the component pin is output.
- **return 0**: Successfully called the function - the given device pin has no component pin.
- **return -1**: Failed calling the function - no valid ID of a slot selected or no project open.
- **return -2**: Failed calling the function - the value specified for **flags** is invalid.
- **return -3**: Failed calling the function - the pin does not belong to a device, connection or net segment.

The following function has been added to the ***e3Pin*** interface for querying the name defined for conductors of cables in the database:

return = e3Pin.GetOriginalCoreName (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 "conductor name"**: Successfully called the function - the return value specifies the name defined for the conductor of the cable in the database.
- **return <EmptyString>**: Failed calling the function - an error occurred.

New interface *e3DbeJob* for editing projects in the Database editor and new function in *e3DbeApplication*

The following functions have been added to the ***e3DbeJob*** interface and exist in ***e3DbeApplication*** with identical functionality:

return = e3DbeJob.CreateDbeAttributeObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeAttributeObject ( )

return = e3DbeJob.CreateDbeComponentObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeComponentObject ( )

return = e3DbeJob.CreateDbeGraphObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeGraphObject ( )

return = e3DbeJob.CreateDbeModelObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeModelObject ( )

return = e3DbeJob.CreateDbeModelPinObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeModelPinObject ( )

return = e3DbeJob.CreateDbeNodeObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeNodeObject ( )

return = e3DbeJob.CreateDbeSlotObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeSlotObject ( )

return = e3DbeJob.CreateDbeSymbolObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeSymbolObject ( )

return = e3DbeJob.CreateDbeTextObject ( ) with identical functionality to
return = e3DbeApplication.CreateDbeTextObject ( )

return = e3DbeJob.GetActiveComponentId ( ) with identical functionality to
return = e3DbeApplication.GetActiveComponentId ( )

return = e3DbeJob.GetActiveModelId ( ) with identical functionality to
return = e3DbeApplication.GetActiveModelId ( )

return = e3DbeJob.GetActiveSymbolId ( ) with identical functionality to
return = e3DbeApplication.GetActiveSymbolId ( )

return = e3DbeJob.GetId ( ) with identical functionality to
return = e3DbeApplication.GetId ( )

return = e3DbeJob.GetName ( )with identical functionality to
return = e3Job.GetName ( )

return = e3DbeJob.GetPath ( )with identical functionality to
return = e3Job.GetPath ( )

return = e3DbeJob.GetSettingValue (name) with identical functionality to
return = e3Job.GetSettingValue (name)

return = e3DbeJob.GetType ( ) with identical functionality to
return = e3Job.GetType ( )

return = e3DbeJob.Save (flags) with identical functionality to
return = e3DbeApplication.Save (flags)

return = e3DbeJob.SaveAs (name, compressed, flags) with identical functionality to
return = e3DbeApplication.SaveAs (name, compressed, flags)

return = e3DbeJob.SetId (id)with identical functionality to
return = e3Job.SetId (id)

return = e3DbeJob.SetSettingValue (name) with identical functionality to
return = e3Job.SetSettingValue (name, value)

The following functions have been added to the ***e3DbeJob*** interface and exist in ***e3DbeApplication*** with slightly modified functionality:

return = e3DbeJob.Close (flags) with similar functionality to
return = e3Job.Close ( ) but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.Create (name, flags) with similar functionality to
return = e3Job.Create (name) but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.
- The return value **-2** has been added.
  The return value signifies that an error occurred when calling the function because a project is already open.

return = e3DbeJob.DeleteComponent (name, version, flags) with similar functionality to
return = e3DbeApplication.DeleteComponent (name, version) but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **1** has been added.
  The return value signifies that the component has been deleted successfully.
- The return value **-6** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-7** has been added.

The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.DeleteModel (name, flags) with similar functionality to **return = e3DbeApplication.DeleteModel (name)** but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **1** has been added.
  The return value signifies that the model has been deleted successfully.
- The return value **-5** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-6** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.DeleteSymbol (name, version, flags) with similar functionality to **return = e3DbeApplication.DeleteSymbol (name, version)** but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **1** has been added.
  The return value signifies that the symbol has been deleted successfully.
- The return value **-6** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-7** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.EditComponent (name, version, flags) with similar functionality to **return = e3DbeApplication.EditComponent (name, version)** but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **1** has been added.
  The return value signifies that the component has been edited successfully.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-6** has been added.

The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.EditModel (name, flags) with similar functionality to **return = e3DbeApplication.EditModel (name)** but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **1** has been added.
  The return value signifies that the model has been edited successfully.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-4** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.EditSymbol (name, version, flags) with similar functionality to **return = e3DbeApplication.EditSymbol (name, version)** but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **1** has been added.
  The return value signifies that the symbol has been edited successfully.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-6** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.GetComponentIds (ids, flags) with similar functionality to **return = e3DbeApplication.GetComponentIds (ids)** but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **-2** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.GetModelIds (ids, flags) with similar functionality to **return = e3DbeApplication.GetModelIds (ids)** but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **-2** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.GetSymbolIds (ids, flags) with similar functionality to
return = e3DbeApplication.GetSymbolIds (ids) but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **-2** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.New (name, flags) with similar functionality to
return = e3Job.New (name) but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.
- The return value **-2** has been added.
  The return value signifies that an error occurred when calling the function because a project is already open.

return = e3DbeJob.NewComponent (name, version, baseName, baseVersion, flags) with similar functionality to
return = e3DbeApplication.NewComponent (name, version, baseName, baseVersion, flags) but with the following changes:

- The return value **1** has been added.
  The return value signifies that the component has been created successfully.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-8** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.NewModel (name, baseName, flags) with similar functionality to
return = e3DbeApplication.NewModel (name, baseName, flags) but with the following changes:

- The return value **1** has been added.
  The return value signifies that the model has been created successfully.
- The return value **-9** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.

return = e3DbeJob.NewSymbol (name, version, baseName, baseVersion, flags) with similar functionality to
return = e3DbeApplication.NewSymbol (name, version, baseName, baseVersion, flags) but with the following changes:

- The return value **1** has been added.
  The return value signifies that the symbol has been created successfully.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because no project is open.
- The return value **-8** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

return = e3DbeJob.Open (name, flags) with similar functionality to
return = e3Job.Open (name) but with the following changes:

- The optional parameter **flags** has been added with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.
- The return value **-1** has been added.
  The return value signifies that an error occurred when calling the function because the value defined for **flags** is not supported.

The following functions have been added to the ***e3DbeApplication*** interface for querying whether projects are open in the Database Editor:

Note: Currently, only one project can be open in each instance of the Database Editor.

return = e3DbeApplication.GetJobCount ( )

Return values:

- **return 1**: Successfully called the function - a project is open in the Database Editor.
- **return 0**: Successfully called the function - no project is open in the Database Editor.

The following functions have been added to the ***e3DbeApplication*** interface for querying IDs of projects that are open in the Database Editor:

Note: Currently, only one project can be open in each instance of the Database Editor.

return = e3DbeApplication.GetJobIds (ids)

Parameter:

- **ids** `[out]`: Contains the ID of the project that is open in the Database Editor.

Return values:

- **return 1**: Successfully called the function - a project in the Database Editor is open and the ID is available in the array **ids**.
- **return 0**: Successfully called the function - no project is open in the Database Editor.

General Changes

Enhanced setting for placing sheet graphics behind the sheet format

Since E3.series Version 2023 it has not been possible, by default, to move sheet graphics to a level behind the sheet format, as this could cause faulty displays (For instance when using transparent graphics). From E3.series Version 2026 it will again be possible to move graphics behind the sheet format. To prevent sheet graphics from being moved behind the sheet format accidentally, the functionality is connected to a setting that will have to be activated in the windows registry.

This allows for more flexibility of the display order of sheet graphics in the new version. Faulty displays in connection with sheet graphics and the sheet format can also be better avoided by having to activate the functionality actively in the registry of the operating system.

In order to be able to move sheet graphics behind the sheet format, activate the functionality either for the *user-specific settings (CURRENT_USER)* or for *all users at the workstation (LOCAL_MACHINE)*.

To do this change the respective registries.

- `HKEY_CURRENT_USER\SOFTWARE\Zuken\E3.series\<version>\Settings` (*HKCU*, for current user)
- `HKEY_LOCAL_MACHINE\SOFTWARE\Zuken\E3.series\<version>\Settings` (*HKLM*, for all users of local machine)

Create the new *DWORD*`AllowGraphicsBehindSheetFormat` in the registry editor.

For this, right-click on the folder `Settings`and select the command ***New→ DWORD-Value (32-Bit)***in the displayed context menu.
Next, enter `AllowGraphicsBehindSheetFormat`as the name for the new string.

Assign the value 1 to `AllowGraphicsBehindSheetFormat` to be able to move sheet graphics behind the sheet format.
If `AllowGraphicsBehindSheetFormat` has the value 0 or is missing, sheet graphics cannot be moved behind the sheet format.

In order to check, whether sheet graphics can be moved behind the sheet format, first the registry is searched. If the function is not activated there, *HKCU* is subsequently searched , and then, if necessary, *HKLM*.
If the *DWORD* is missing or has the value 0, sheet graphics cannot be moved behind the sheet format.

References: `E3-3639`

Start parameter for showing loading times of program components of E3.series

With a start parameter the loading times of important program components can be output in the Output window of E3.series.
With an additional parameter this information can be written directly to a file.

This allows for better examination of irregularities in the starting behavior of E3.series, for example by the Zuken support.

In order to display the loading times of important program components when starting, use the respective start parameter.

When the parameter **/profiling=0** or the parameter **/profiling**without a value is used, the amount of time, that was needed for the start of the main function, will be displayed.

When the parameter **/profiling=1** is used, a more detailed list will be displayed that also shows, for instance, the amount of time that was needed to load the individual licenses.

In order to write the information output by **/profiling** into a file, use the start parameter **/profilingfile** and enter a path and file name as the value of the parameter.

Example: `/profilingfile="C:\Temp\Profiling.txt"`

Please ensure that you have write permission for the specified folder.

If a file of the same name already exists in the target folder, it will be overwritten.

References: `E3-2902`

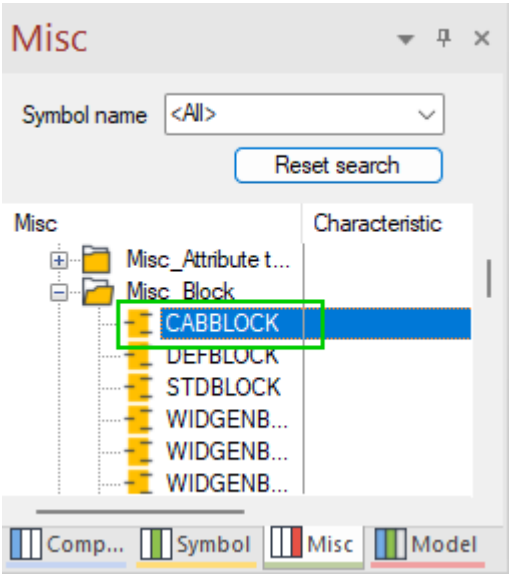
Text type for displaying individual block numbers and the the total of split blocks

For split blocks, it is possible to display the amount of blocks a given block is split into and what number the respective blocks have using a text type.
The text type can be assigned to block symbols that are used for the display of split blocks.
The number of the split block is determined using the name of the sheet on which the split block is placed and the position of the block on the sheet.

This makes it easier to view how often the block has been split and on which sheet the split block can be found.

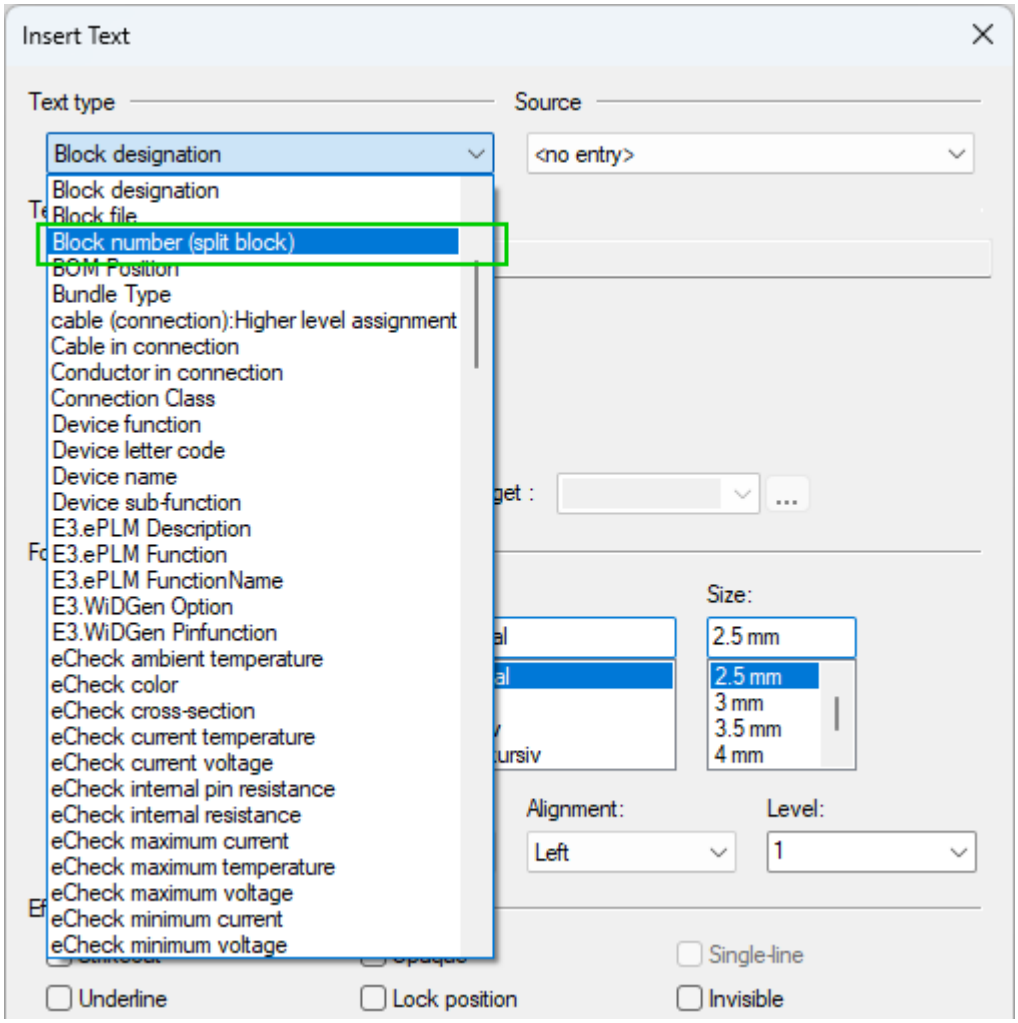
In order to display the individual block number and the total number of split off blocks, select ***Tools → Start Database Editor*** in the main menu.
The database editor opens.

Edit the block symbols that are to be used in the project and the information for split blocks to be displayed, for example ***CABBLOCK*** under ***Misc***:



In order to add the text type, select ***Insert → Text...*** in the main menu.
The ***Insert Text*** dialog opens.

Under ***Text type*** select the text type ***Block number (split block)*** and place it on the block:



The text type has the number **1187** and is only displayed when at least one split block of the block object is placed in the project.
For blocks, that are not split, the text type **1187** show no content.

The order of the numbers of split blocks is assigned according to the position of the block in the project. First the alphanumeric sorting of the sheets from 1 to 9 and from A to Z is taken into account, then the X-coordinate on the sheet from left to right and then the Y-coordinate on the sheet from top to bottom.

References: E3-1373

Database Editor

General improvements in handling of the Database Editor

The Database Editor has been enhanced with improvements for handling.

This makes working with the Database Editor more intuitive and efficient.

The following improvements have been made:

1. **Deleting contours via the tree view from the Database Editor**

In order to delete contours in the Database Editor, search for a model with contours that is to be deleted and edit it.

Then select all contours that are to be deleted in the workspace or tree view.

Open the context menu and select the ***Delete*** command.

Note: The command is only available if all selected objects can be deleted.

2. **Saving objects to the database via the tree view**

In order to save components, models and symbols to the Database Editor, create a new object or edit an existing one.

Then open the context menu of the objects that are to be saved in the tree view and select ***Save To Database***.

To save all components, models and symbols, that are being edited, open the context menu of the respective folder and select ***Save To Database***.

Note: When multiple objects are selected, the command is only available if all selected objects are of the same type such as ***component***.

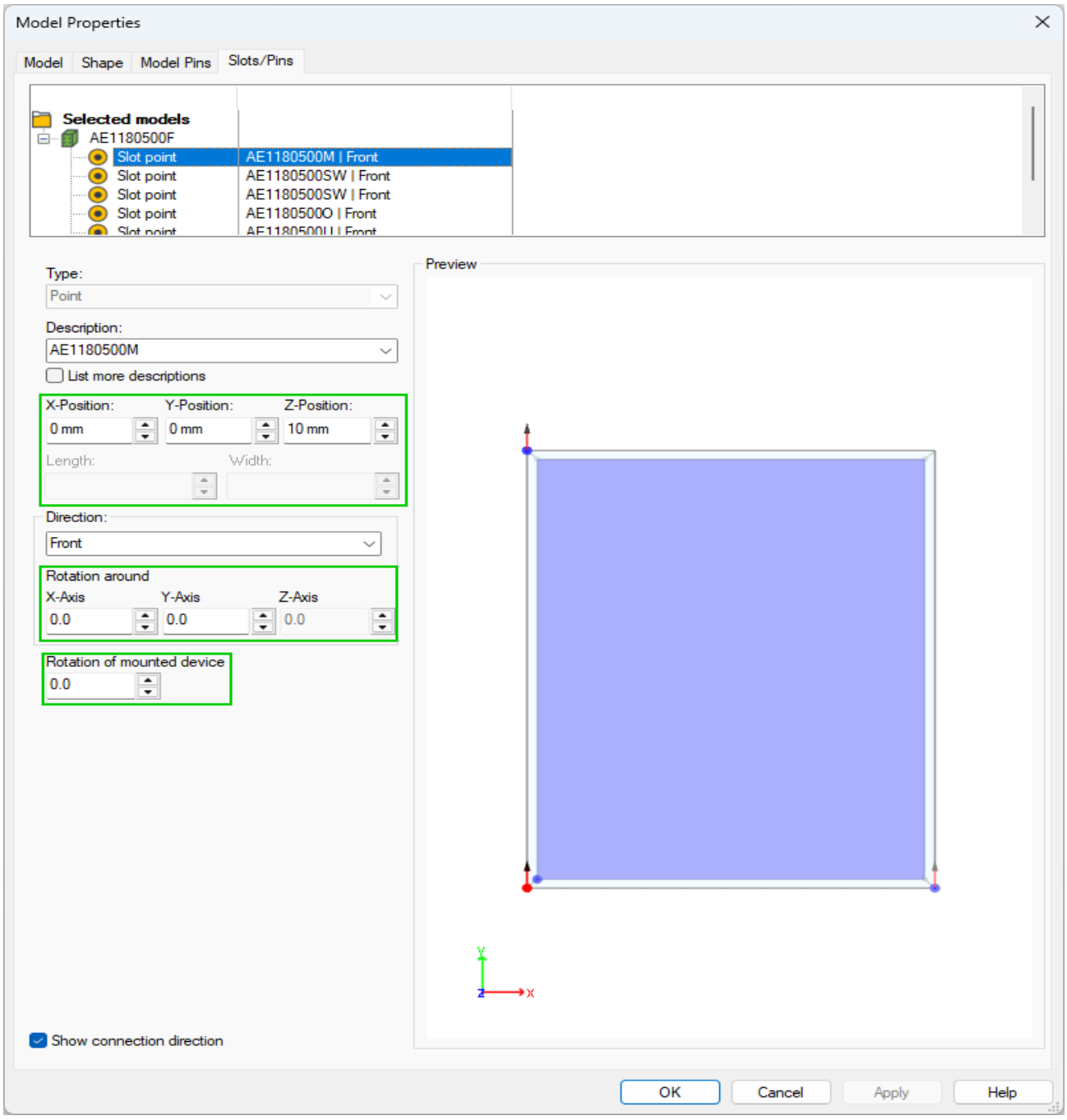
Defining properties of slots and the rotation of devices placed on slots

The definition of slots in the database has been enhances so that, among other things, the rotation values can be defined.
In addition, it is possible to define how devices are rotated when they are placed on slots.
The rotation of placed devices can only be defined for slots of the types ***Line*** and ***Point***.

This allows for better control and optimization of processes around the planning and montage of panels.

In order to define properties of slots and the rotation values of devices, that are to be placed on slots, in the Database Editor, create a new model or edit an existing one.

Then open the ***Model Properties*** and open the ***Slots/Pins*** tab:



Define the position of the slot with x, z and y coordinates under ***X-Position:***, ***Y-Position:*** and ***Z-Position:*** .
The coordinates refer to the origin of the model.

Define the length of the slot under***Length:***.
For slot of type ***Line*** the value corresponds to the length of the line.
For slot of type ***Area*** the value corresponds to the extent of the length.
The length cannot be defined for slots of the type ***Point***.
For unrotated slots in the front view, the length increases in the direction of the X-axis, starting from the origin of the design.

Define the length of the slot under ***Width:***.
For slot of type ***Line*** the value corresponds to the width of the capture range.
For slot of type ***Area*** the value corresponds to the extent of the length.
The width cannot be defined for slots of the type ***Point***.
For unrotated slots in the front view, the length increases in the direction of the Y-axis, starting from the origin of the design.

For slots, that are rotated or have a different direction, a different axis may define the length or width.

Under ***Rotation around*** it is possible to define the rotation value of the slot around the X-, Y- and Z-axis.

Under ***Rotation of mounted device*** it is possible to define the rotation value of the device to be placed on the slot.
The rotation can only be defined for slots of the types ***Line*** and ***Point***.
The rotation takes place in the top view of the slot.

Grouping graphic and text elements of models and symbols in the Database Editor

Graphic and text elements of models and symbols can be grouped in the Database Editor.
The objects in the resulting object groups can then be edited together, for example by moving or copying them, and therefore behave like object groups in the project mode.

This allows for instance for simpler and faster changes of a symbol with multiple graphic and text elements in the Database Editor.

In order to group graphic and text elements of models and symbols in the Database Editor, edit am existing object or create a new one.

Then place the graphic and text elements to be grouped and open the context menu.

Next select the ***Group Objects*** command. Graphics, that mark the space requirements of objects or the drawing area of sheets, cannot be grouped.
The selected objects will be grouped.

References: `Designer-4284` und `E3-160`

Defining signals for connectors, devices and terminals in the Database Editor

The Database Editor has be enhanced to allow for the definition of signals for connectors, devices and terminals.

This allows for further objects with predefined signals to be created for specific uses in the database, making working with signals altogether more user friendly.

In the Database Editor signals can be defined for components of type **Connector**, **Device**, **Terminal** and **Block**.

First create or edit a component of this type in the Database Editor.

Then open the component properties, go to the ***Signal*** tab and define signals for the required pins.

Note the following when working with components for which signals are defined:

- When switching components of devices in the project, the signals of the pins are not changed.
- When updating the component in the project, the signals are overwritten according to ***General → Update in Project → Overwrite signals of block connectors***.
  The following rules apply:
  - System generated signals are adopted from the database only if no user defined signal is used in the project.
  - The signal ****NC**** is only adopted if the pin is not plugged.
  - A missing signal value ***<no entry>*** is not adopted.
  - Translated texts are supported.

References: `Designer-07017` and `E3-1116`

Importing and Exporting Data

Importing Data

- Enhancements of import of STEP files
- Defining character encoding for the export of DWG files in the configuration file

Exporting Data

- Highlighting hyperlink targets when exporting PDF files on jump
- Defining character encoding for the export of DWG files in the configuration file

---

Enhancements of import of STEP files

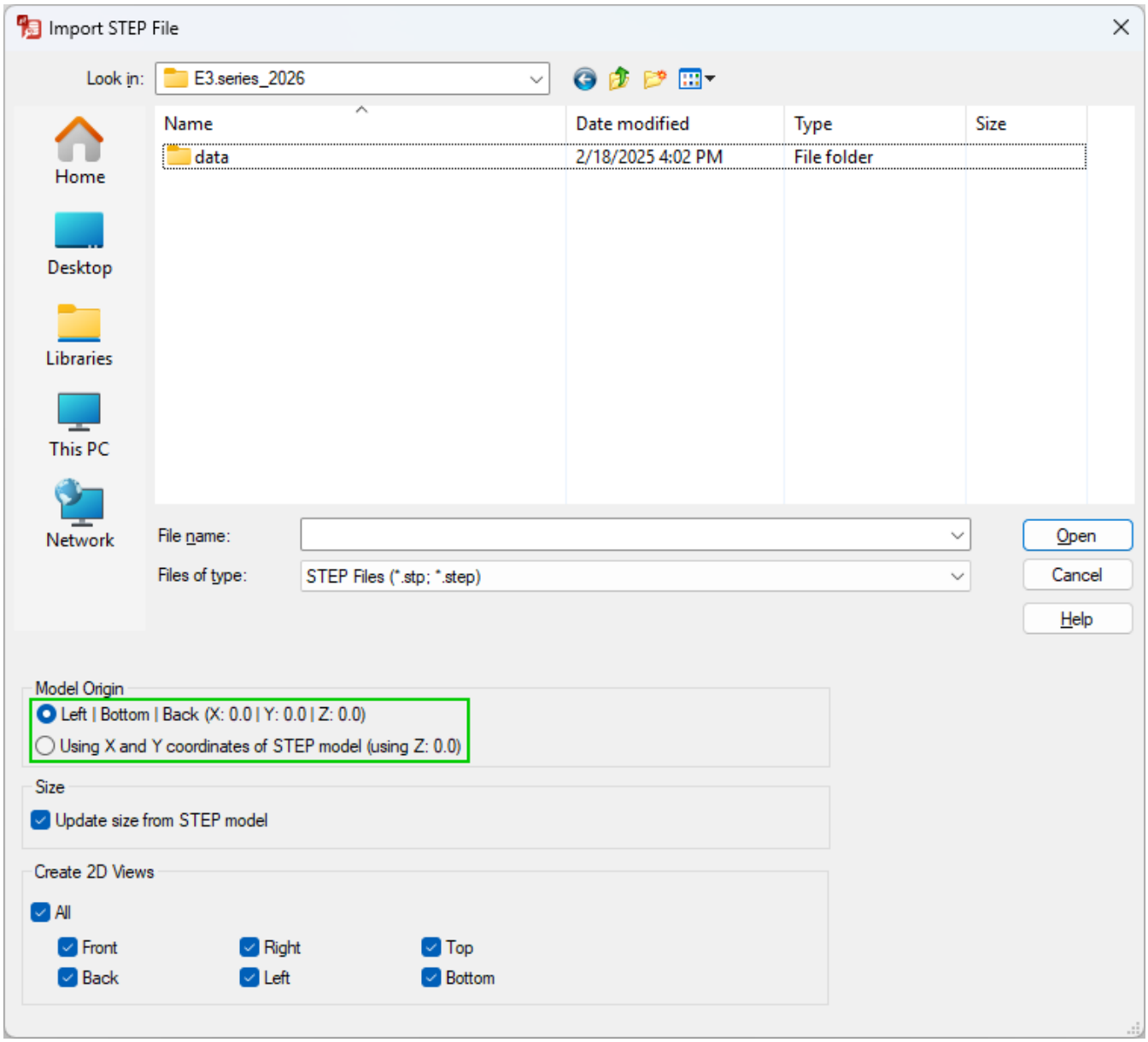
When importing STEP files it is possible to configure which origin coordinates are to be used for the model in E3.series.

This allows for more flexibility of the use of STEP files in E3.series.

In order to configure the placement of the STEP model on the model in E3.series, select ***Tools*** → ***Start Database Editor*** in the main menu.
The database editor will open.

Next select a model to which a STEP model is to be added and select ***Insert*** → ***Import STEP file*** in the main menu or select the ***STEP model...*** command in the ***Database Model*** toolbar.
The ***Import STEP file*** dialog opens.

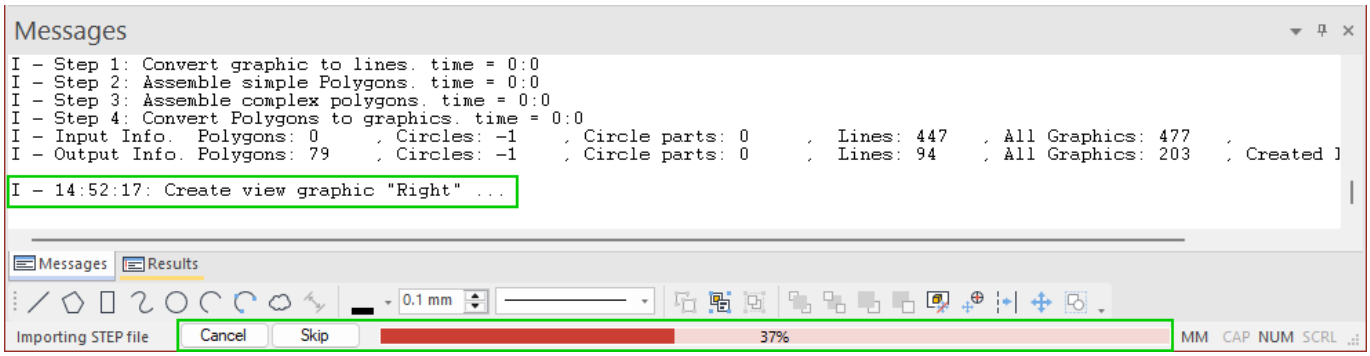
Under ***Model Origin*** you can configure the placement of the STEP model:



Select ***Left | Bottom | Back (X: 0.0 | Y: 0.0 | Z: 0.0)*** in order to ignore the origin of the STEP model and use the origin of the model in E3.series (0|0|0).
Or select ***Using X and Y coordinates of STEP model (using Z: 0.0)*** in order to use the origin of the STEP model as the origin of the model in E3.series.

The progess of the import of STEP files is displayed in the Messages window.
In addition, the current stage of the import process is displayed.

This allows for better understanding of how far the import has progressed, especially with larger STEP files.
The individual steps or the entire import can also be canceled:



In order to skip individual steps of the STEP import click on ***Skip*** in the status bar.
The import will then be continued with the next view.

In order to cancel the entire import, click on ***Abort*** in the status bar.
The import is canceled and no information from the STEP model is added to the model.

Highlighting hyperlink targets when exporting PDF files on jump

Cross-references that are clickable in exported PDF files and lead to the target can be highlighted when jumping.

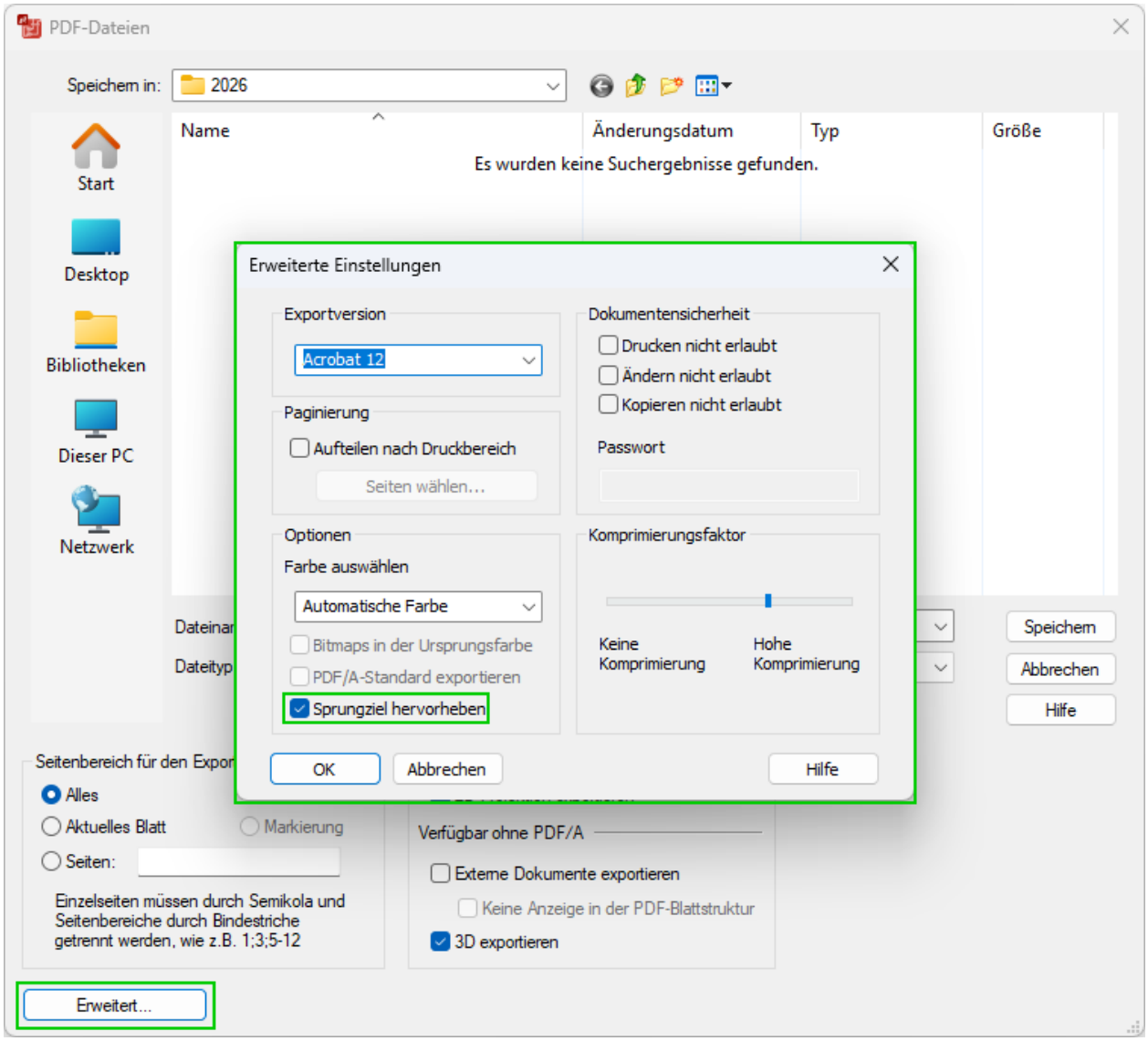
This allows for easier viewing of the targets the link refers to, particularly on pages with a large number of hyperlinks, which improves navigation in PDF files overall.

To highlight the destinations of hyperlinks when jumping, open a project with hyperlinks.

Then either select ***File → Export → Export PDF...*** in the main menu window to export all sheets of the project or open the context menu of a sheet in the sheet view and select the ***Export PDF...*** command.
The ***PDF Files*** dialog opens.

Then click on the ***Advanced...*** button in the dialog.
The ***Advanced settings*** dialog opens.

Finally, activate the setting ***Highlight jump*** and start the export of the PDF file. The setting cannot be activated if the ***Export PDF/A standard*** setting is active:



If you click on a hyperlink in the PDF file, you will jump to the target of the hyperlink. The jump target is highlighted in color and flashes.
The highlighting flashes five times and then disappears.

The color defined in the settings under ***General → Highlight → Color:*** is used for the highlighting.

References: `Designer-15372` und `E3-161`

Defining character encoding for the export of DWG files in the configuration file

The character encoding used for text when exporting DWG files can be defined in the configuration file and in the registry.
Note: The character encoding for DXF files is defined via the encoding of the file itself.

Encoding that causes problems when the content of the DWG file is checked and processed with another program can thereby be avoided.

In order to define the character encoding in the configuration file, open `DXFExport.cfg` in the installation folder of E3.series and create the configuration key `[DWG Code Page]` if it does not already exist. Alternatively, you can change the character encoding in the registry under `HKCU\Software\Zuken\E3.series\<version number>\Settings` with the entry `DXFDWGCodepage`.

The following values are possible:

- `ASCII`; in the registry corresponds to the value `1`
- `8859_1` or `iso8859-1`; corresponds to the value `2` in the registry
- `8859_2` or `iso8859-2`; corresponds to the value `3` in the registry
- `8859_3` or `iso8859-3`; corresponds to the value `4` in the registry
- `8859_4` or `iso8859-4`; corresponds to the value `5` in the registry
- `8859_5` or `iso8859-5`; corresponds to the value `6` in the registry
- `8859_6` or `iso8859-6`; corresponds to the value `7` in the registry
- `8859_7` or `iso8859-7`; corresponds to the value `8` in the registry
- `8859_8` or `iso8859-8`; corresponds to the value `9` in the registry
- `8859_9` or `iso8859-9`; corresponds to the value `10` in the registry
- `DOS437`; corresponds to the value `11` in the registry
- `DOS850`; corresponds to the value `12` in the registry
- `DOS852`; corresponds to the value `13` in the registry
- `DOS855`; corresponds to the value `14` in the registry
- `DOS857`; corresponds to the value `15` in the registry
- `DOS860`; corresponds to the value `16` in the registry
- `DOS861`; corresponds to the value `17` in the registry
- `DOS863`; corresponds to the value `18` in the registry
- `DOS864`; corresponds to the value `19` in the registry
- `DOS865`; corresponds to the value `20` in the registry
- `DOS869`; corresponds to the value `21` in the registry
- `DOS932`; corresponds to the value `22` in the registry
- `MACINTOSH` or `mac-roman`; corresponds to the value `23` in the registry
- `big5`; corresponds to the value `24` in the registry
- `ksc5601`; corresponds to the value `25` in the registry
- `johab`; corresponds to the value `26` in the registry
- `DOS866`; corresponds to the value `27` in the registry
- `ansi1250`; corresponds to the value `28` in the registry

- ansi1251; corresponds to the value 29 in the registry
- ansi1252; corresponds to the value 30 in the registry
- gb2312; corresponds to the value 31 in the registry
- ansi1253; corresponds to the value 32 in the registry
- ansi1254; corresponds to the value 33 in the registry
- ansi1255; corresponds to the value 34 in the registry
- ansi1256; corresponds to the value 35 in the registry
- ansi1257; corresponds to the value 36 in the registry
- ansi874; corresponds to the value 37 in the registry
- ansi932; corresponds to the value 38 in the registry
- ansi936; corresponds to the value 39 in the registry
- ansi949; corresponds to the value 40 in the registry
- ansi950; corresponds to the value 41 in the registry
- ansi1361; corresponds to the value 42 in the registry
- ansi1200; corresponds to the value 43 in the registry
- ansi1258; corresponds to the value 44 in the registry
- utf8; corresponds to the value 45 in the registry
- cnt; corresponds to the value 46 in the registry

Which encoding is used for the export is decided using the following prioritization:

1. When an encoding is defined in the configuration file DXFExport.cfg, this encoding is used.
2. When an encoding is defined for the value DXFDWGCodepage in the registry under HKCU\Software\Zuken\E3.series\<Versionsnummer>\Settings, this encoding is used.
3. When no encoding is defined, DOS850 is used.

References: Designer-44212 and E3-503

Multiuser

Configure size of data package to be written into database

Actions, that are executed in multiuser projects, are collected in data packages first. Once the data package has reached a certaib size, they are written to the database.
The size of the data package, that is to be saved, can be configured.

This make it possible to adjust how often data is written to the database on the server, so either fewer big data packages or many small ones are synchronized with the database.
The bigger the data package, the more storage is needed on the device of the user.

In order to configure the size to be written to the database, adjust the configuration file `configure.xml` on the multiuser server.

Add the element **<UpdateBatch>** to the element **<Server>** in the configuration file.

Select a value between ***2*** and ***4294967295*** as the content. The value represents the size of the data package in kilobytes.
For most use cases the value **65536** is suitable.

When a valid value is defined for **<UpdateBatch>**, **<UsePreparedStatements>** in the configuration file is ignored.

Example:

XML Configuration

```
<Server>
                <!-- ... -->
                <UpdateBatch>65536</UpdateBatch>
            </Server>
```

Note: When using an Oracle database system, it is recommended to use **<UsePreparedStatements>**.

References: `E3-3485`

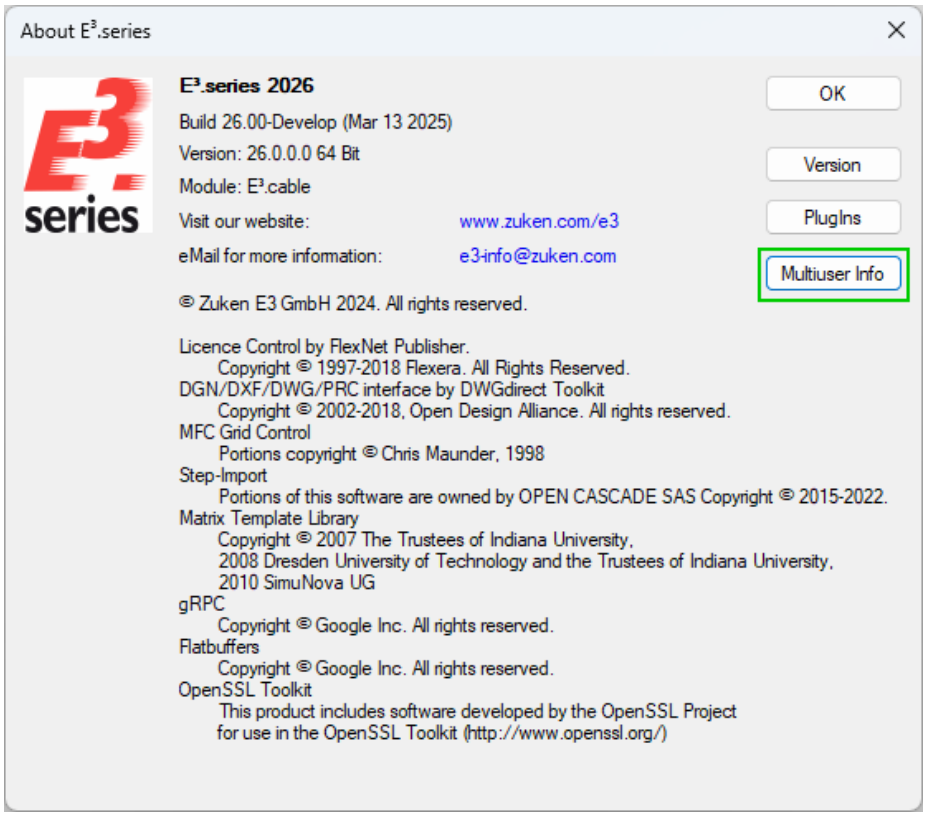
Output information about the opened project

For open projects, it is possible to output which users are connected to the project and which users have the project open.

This makes it easier to track whether other users currently connected to the project are making further changes to the project.

In order to output the information of the open multiuser project, select ***Help → About E³.series...*** in the main menu.
The ***About E³.series*** dialog will open.

Next, click ***Multiuser Info***:



The information about the open multiuser project will be output in the output window.

References: `Designer-13112` and `E3-4147`

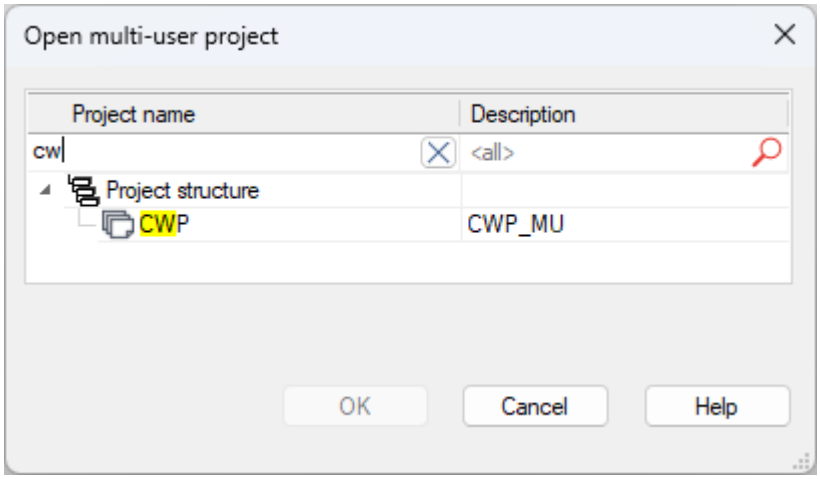
Filtering projects by name and description when opening, deleting or renaming

Multi-user projects can be filtered by name and description.
This is available when opening, deleting or renaming projects.

This improves the project organization on multi-user servers.

In order to filter multi-user projects by name or description, select ***Tools*** in the main menu. Then select either ***Open Multi-user Project...***, ***Delete Multi-user Project...*** or ***Rename Multi-user Project...***
The respective dialog will open.

Enter the filter terms in the (missing or bad snippet) and the ***Description*** column:



Results, that do not correspond to the filter terms, are automatically hidden.

References: `Designer-18156` and `E3-2002`

Updating Objects

Show only objects whose time stamp has changed when updating components and symbols in the project

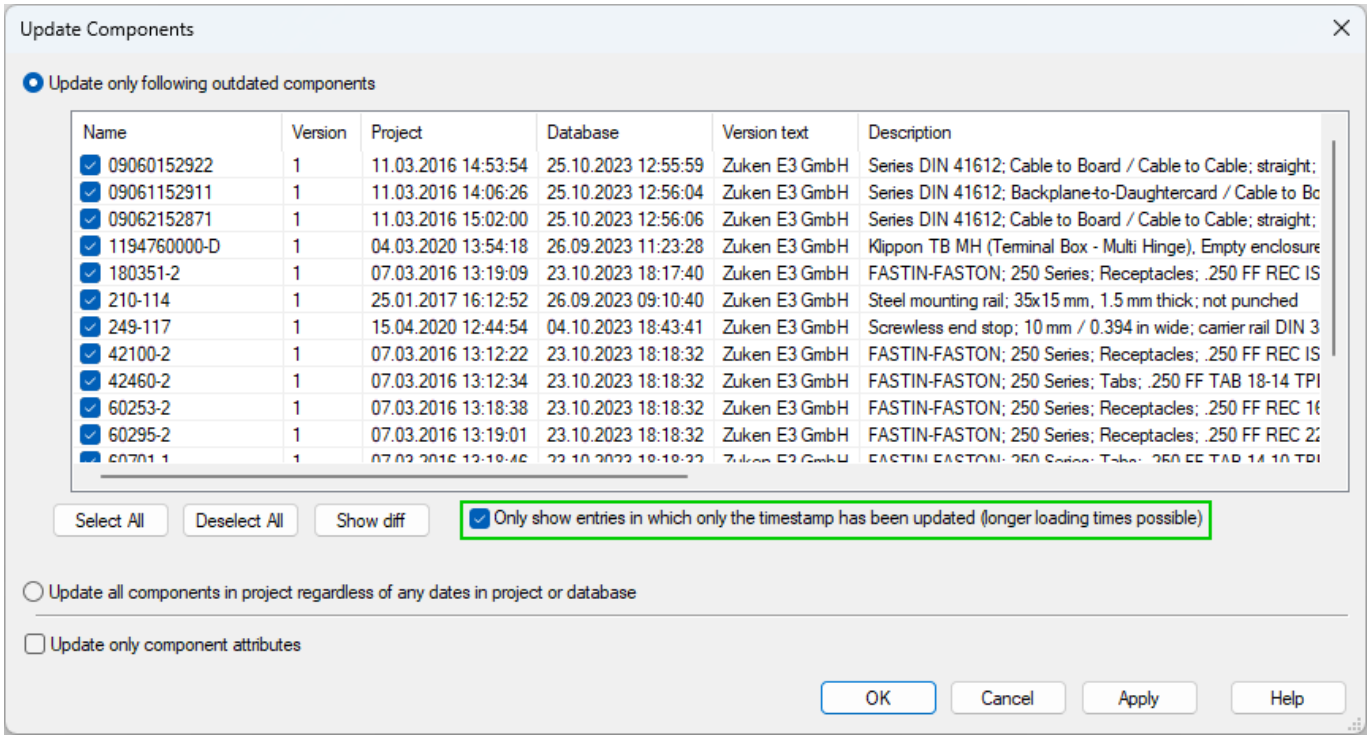
The function for updating components and symbols in the project has been enhanced to show only objects where no attributes or values have changed except the time stamp.
That only the time stamp and no other information differs for an object can occur, for example, if objects are updated using E3.DBUpdate and the configuration key `AutoRefreshLASTUPDT=1` is set.

This allows for identification of objects where only the time stamp has changed but no attributes or values that influence productive projects.
The time stamp of these objects can then be updated selectively, which significantly reduces the number of 'outdated' components and symbols.

To display and update only components or symbols where the time stamp has changed, open a project in E3.series.

Then open the context menu on a folder or the upper most element of the database window and select either Update all components in project... or ***Update all symbols in project...*** depending on the active tab.

The ***Update Components*** or ***Update Symbols*** dialog opens:



Then activate the setting ***Only show entries where nothing has changed except the timestamp (can take up to several minutes!)*** .

The view is updated automatically and not shows only objects where there is no difference to the version saved in the database except the time stamp.

Note: The filtering of the objects where only the time stamp has changed might take some time. The progress will be displayed in the status bar of E3.series.

Finally, select all objects whose time stamps are to be updated and save the changes to the database with ***Apply***.

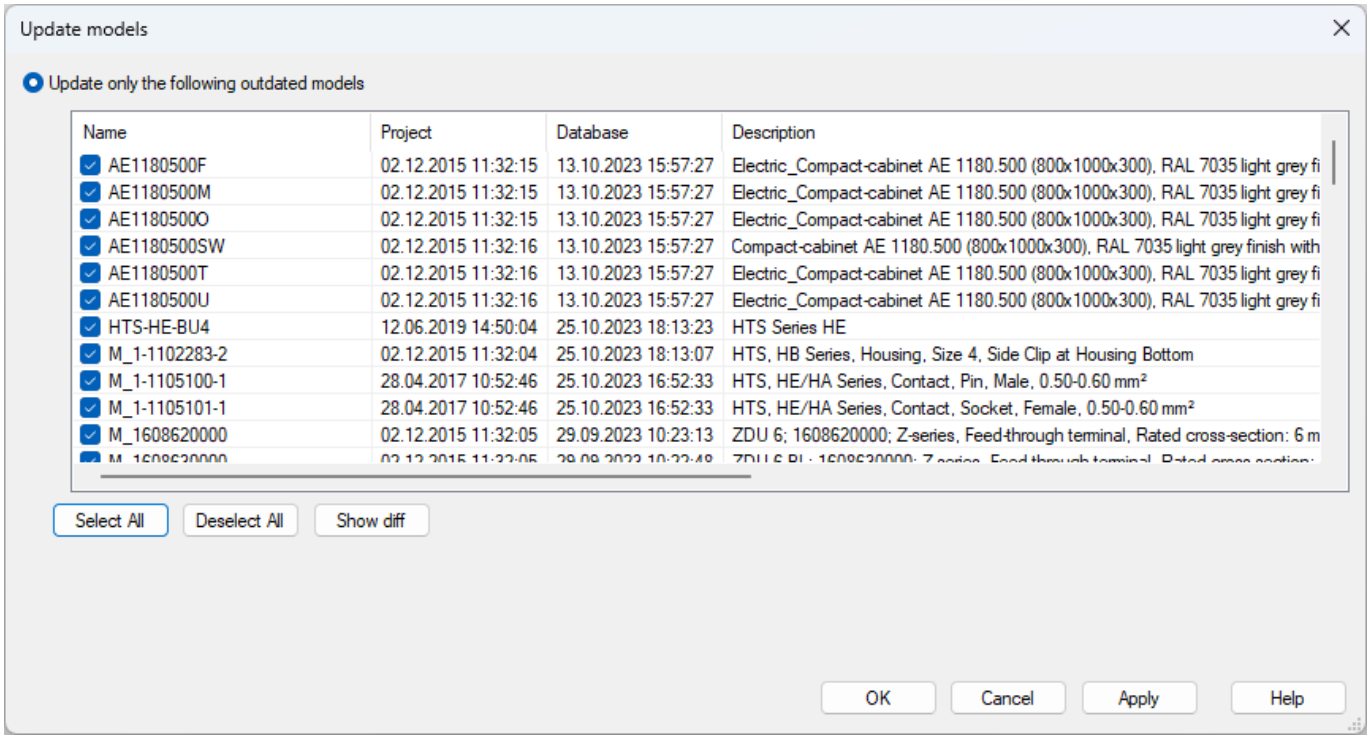
Only update models whose slots are unchanged

When updating model used in the project, it is possible to update only those models whose slot number and properties are unchanged.
The model components remain unchanged.

Thus it is no longer necessary to update all devices, that use the model component, when updating models. When updating this way it is possible to change the position of slots, contours and graphical views.
This makes the updating of objects more efficient.

In order to update only models with unchanged slots, open a project in E3.series.

Next, open the context menu on a folder or the uppermost element of the database window and select ***Update all models in project...***
The ***Update models*** dialog will open:



Select all models to be updated and confirm the selection with ***Ok*** or ***Apply***.
The selected models are updated.

It is possible that placed models are moved when updating, because the properties of models change, such as the description of the model or the restriction.

Project Handling

Keeping changes of dimension texts and the referenced objects synchronized

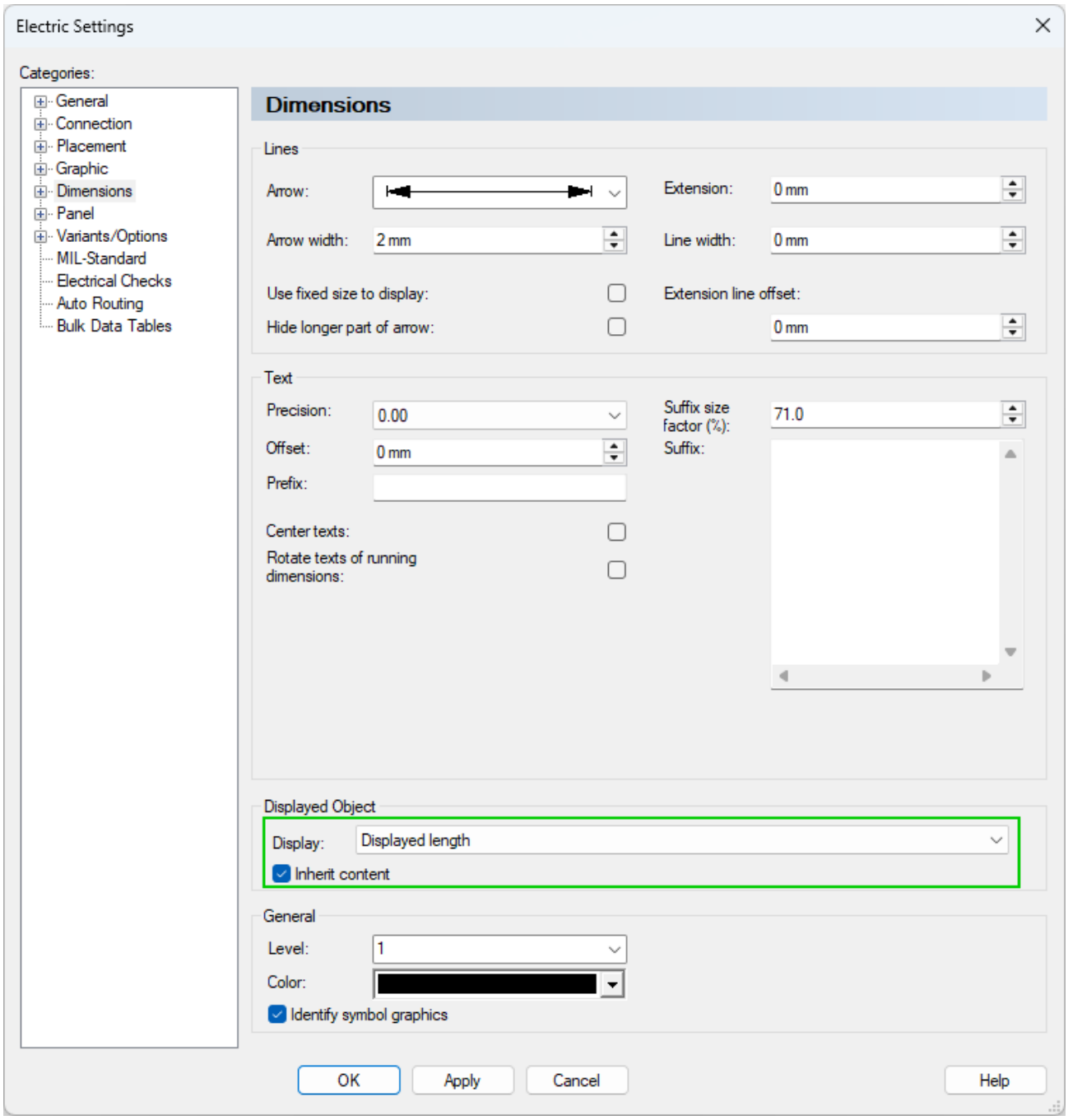
It is possible to keep changes of dimension texts or the information of the connected objects synchronized across both objects.

This allows for changes of the dimension text to be transferred to the referenced object and vice versa, improving the overall data consistency.

In order to keep changes of dimension texts and the referenced objects synchronized, select ***Tools → Settings...*** in the main menu.
The ***Electric Settings*** dialog opens.

Note: The setting can be adjusted via the ***Electric Settings*** and the ***Fluid Settings*** dialog.

Next, activate the ***Inherit content*** setting under ***Dimensions***.
Optionally. select the attribute or the information to be synchronized between the dimension text and reference object under ***Display:***
The setting is user specific:



Please take into account the following restrictions:

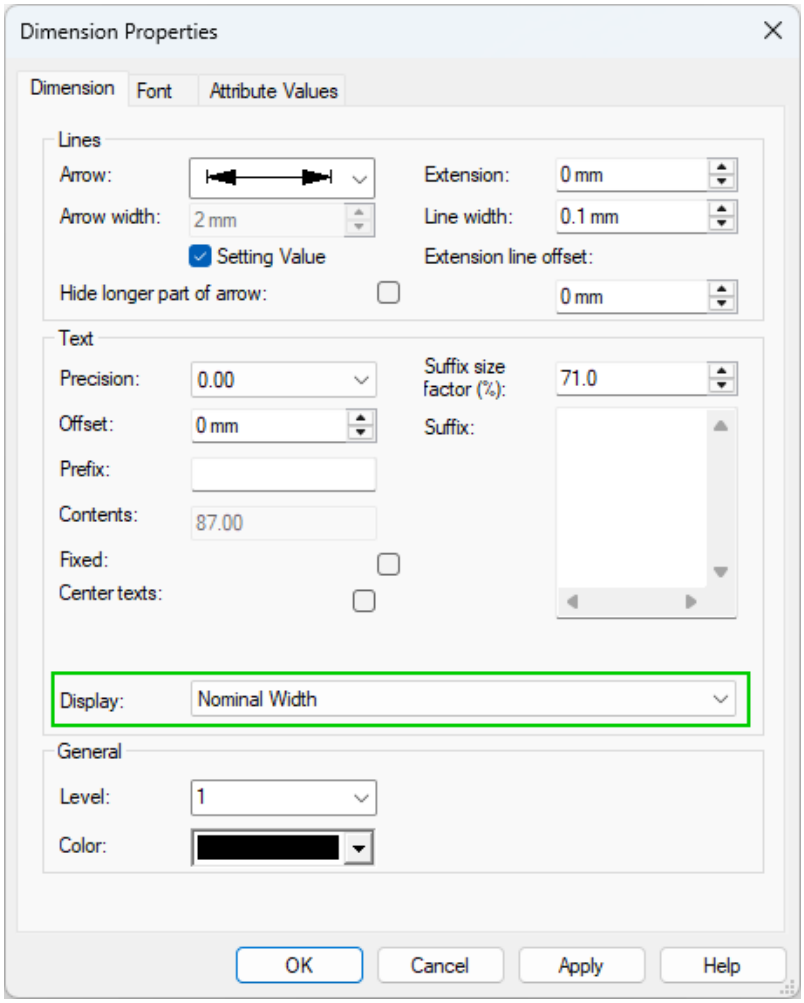
- Attribute lists, that have a ***Range for numeric values:*** or are ***Read only***, cannot be kept synchronized.
- Attributes, for which the option ***Changeable by script only*** is defined, cannot be kept synchronized.
- The ***Prefix:*** and ***Suffix:*** of the dimension text are not applied to the reference object when changing.

- When a dimension is set across multiple net segments the sum of the individual net segment length is used for the dimension text.
- When a dimension is set across multiple reference objects and an attribute with a string is used for the display of the dimension text, the content of the dimension text depends on whether the attribute value of the reference objects is identical:
  - When the values are identical, the value is displayed.
  - Otherwise ***<multiple values>*** is displayed.
- When the option ***Fixed:*** is set for dimension texts, no comparison is made between the reference object and the dimension text.
- When an attribute is displayed for the dimension text, that is not available for the respective reference object, the attribute is only added when the dimension text changes and the option ***Fixed:*** is not set.

Changes to the text of the dimension object or the respective reference object are also automatically made at the other location.

It is possible to set for each individual dimension object which attribute or information is to be synchronized.

To do this open the respective dimension property and select which information of the object is to be displayed and kept synchronized under ***Display:***



References: E3-1139

Controlling display of unchanged sheet frames during graphical comparison

The configuration file for the graphical comparison of projects has been enhanced for controlling whether unchanged sheet frames of sheets are displayed when other contents on the sheet have changed.

This allows unchanged sheet frames to be displayed during graphical comparison that were always hidden in the previous version of E3.series 2026.
In previous versions, when comparing sheets in projects, unchanged sheet frames always displayed empty content, which could cause issues with checking the projects.

In order to display the content of unchanged sheet frames, modify the configuration file that is used for the project comparison.

In the configuration section `[Result Options]`, add the configuration key `showUnchangedRegions`with the value `true`.
In addition, the configuration key `ResultFile` must have the value `ChangedOnly`.

When `showUnchangedRegions` has the value `false`, unchanged sheet frames are not displayed during graphic comparison.
This value is specified by default.
The result with this value correspond to the behavior of previous versions of E3.series 2026.

References: Designer-43560

Opening properties of contours and slots via the tree view

In the project mode and the Database Editor, properties of contours and slots can be opened directly via the tree view.

This allows for more efficient displaying and changing of properties of contours and slots.

In order to open the properties of contours and slots, select the contours or slots whose properties are to be displayed, in the tree view.

Open the context menu and select ***Model Properties...*** in the Project Mode or ***Properties...*** in the Database Editor.
The ***Model Properties*** are opened directly on the active tab ***Contour*** or ***Slots/Pins*** respectively.

The selected object is highlighted in the table and its properties are displayed.
When multiple objects are selected, all objects are highlighted in the table and only those properties are displayed that all objects have in common.

Note: The properties can only be opened via the context menu if only contours or slots are selected.

References: Designer-14960

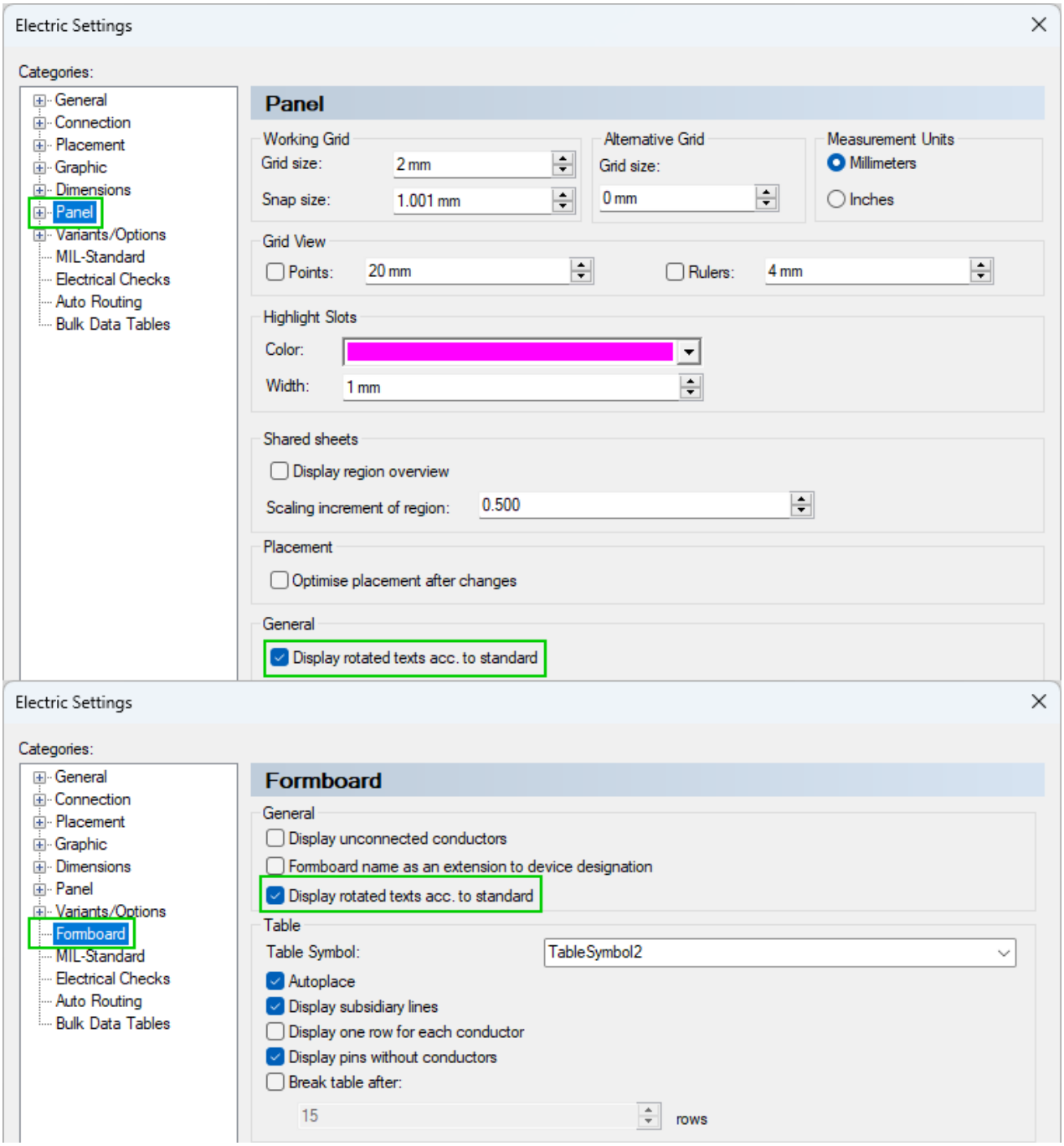
Displaying rotated texts on panel and formboard sheets according to standard

For rotated texts that are placed on frames of panel or formboard sheets, it is possible to individually control whether they confirm to the DIN norm.

This allows for better individualization of the behavior for displaying rotated texts in special use cases.

In order to control the display of rotated texts that are placed on frames of panel or formboard sheets, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.
Note: For the panel and formboard functionality, corresponding licenses are required.

Then, activate the setting ***Display rotated texts acc. to standard*** under ***Panel*** or ***Formboard***.
The display of content outside the area of the panel or formboard sheet, such as the sheet border, are controlled using ***General → Display → Display rotated texts acc. to standard***.
The setting is user specific:



When the setting is active, texts are always displayed independent of the symbol, such that they can be read from the bottom or from the right.

References: `Designer-05288`

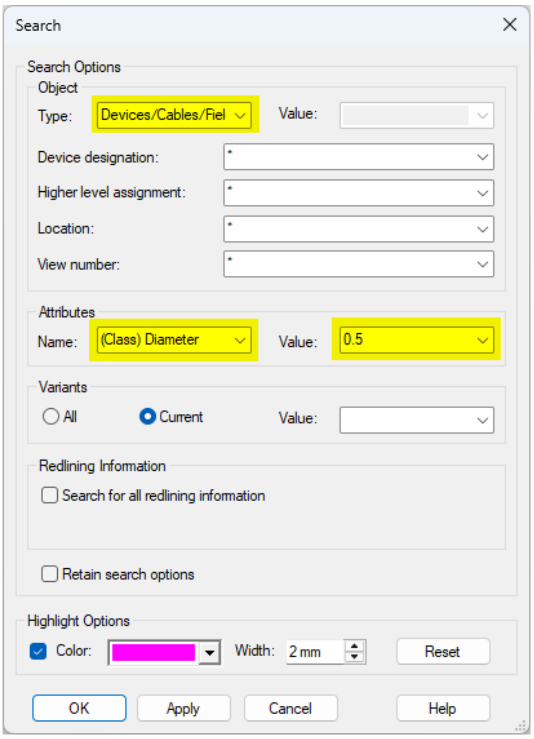
Search for model attributes finds devices as well

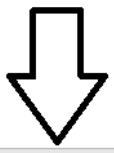
When searching for attributes with the owner *Component*, *Cable type* or *Model*, all devices, that have the attribute, will also be found.

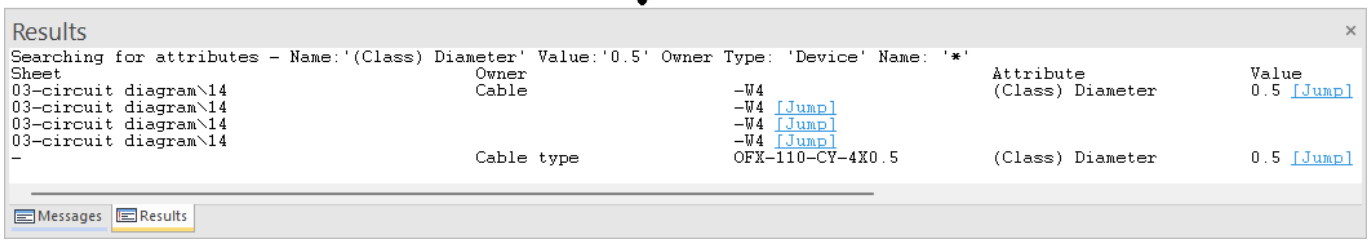
This makes it easier to view whether the attribute value of the device deviates from the value defined for the component.

To do this select ***Edit → Search...*** in the main menu and look for the required attribute in the ***Search*** dialog.

In the output window all devices with the searched for attribute, are displayed under ***Results***.
Click on ***Jump*** in order to highlight the object in the structure or on the schematic.







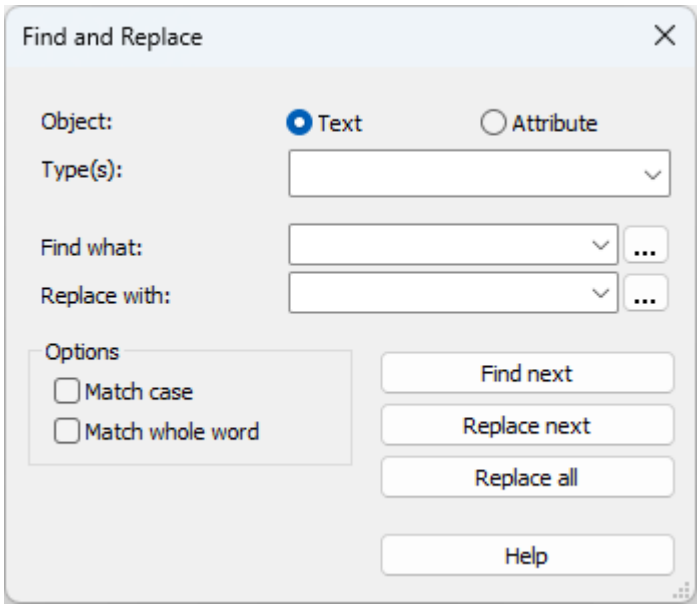
Search and replace texts and attribute values in the project

In addition to the existing functionality of searching projects for texts and attribute values, the search results can also be replaced with other texts or attribute values from E3.series 2026.

This allows for editing many objects at once.

In order to search for and replace texts and attributes, open a project in which changes are to be made.

Select the ***Replace...*** command in the ***Edit*** main menu.
The ***Find and Replace*** dialog opens:



In this dialog, it is possible to configure the search with the following settings:

Find and Replace	
Object:	Defines whether the content of text objects or the values of attributes are searched for.
Type(s):	Displays a list of text types or attributes that can be taken into account for finding and replacing. Depending on the selection, only text types or attributes are displayed in the list. Note the following on the availability of text types and attributes: **Text types** • User texts are available • Comment texts are available • Texts that are assigned to attributes are **not** available **Attributes** • Only attributes of the type ***Text*** are available • Attributes with owners that are changeable are available • Attributes that are only changeable via script are **not** available • Internal attributes are **not** available
Find what:	Defines the search term.

	When searching, hidden or inactive texts and attributes are displayed. In order to use terms from the translation table, click on ... and select the required term or enter the translation code directly.
Replace with:	Defines what the found results are to be replaced with. In order to use terms from the translation table, click on ... and select the required term or enter the translation code directly. Attribute values that can be changed are displayed blue in the output window. Attribute values that cannot be changed are displayed black in the output window. **Note:** Locked objects and objects of the following types cannot be changed: • Components and component pins • Cables • Symbols and symbol pins • Wires • Slots • Contours • Functional units and functional ports • State • Bundles
Options	
Match case	If this option is active, only terms that exactly match the specified case are found. If this option is **not** active, terms are displayed irrespective of case.
Match whole word	If this option is active, the search term is only found if it is not part of another word. If this option is **not** active, the search term is found even if it is part of another word. **Example:** When this option is active, only results containing the exact word **engine** will be found for the selected objects and types with the search term **engine**. When the option is **not** active, values such as **engineer** are also found with the search term **engine** for the selected objects and types.
Find next	Displays the next result of the selected search term.
Replace next	The next search result of the selected search term is replaced with the required text. Each individual replacement is a single server transaction when using multi user projects.
Replace all	Replaces all search results in the selected text. Regardless of the number of texts to be replaced, the command counts as a single

server transaction when used in multi-user projects.

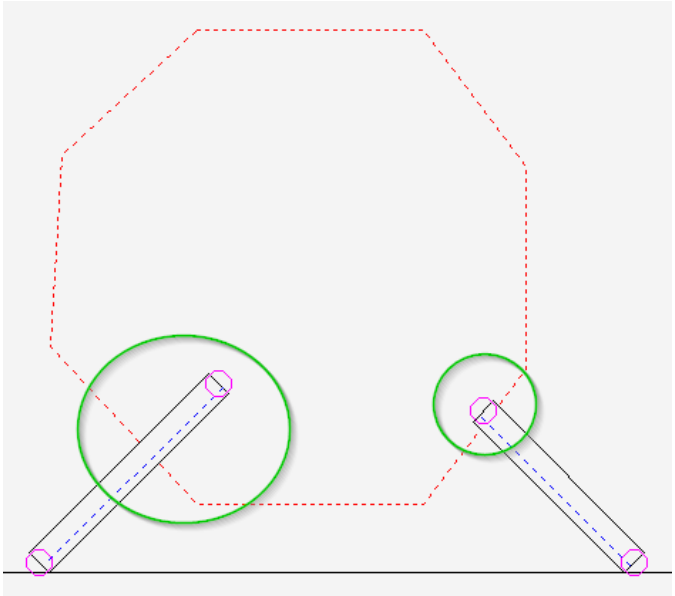
Panel (also: 3D)

Using cutout areas for automatic connections

It is possible to create automatic connections through cutout areas in order to connect devices.

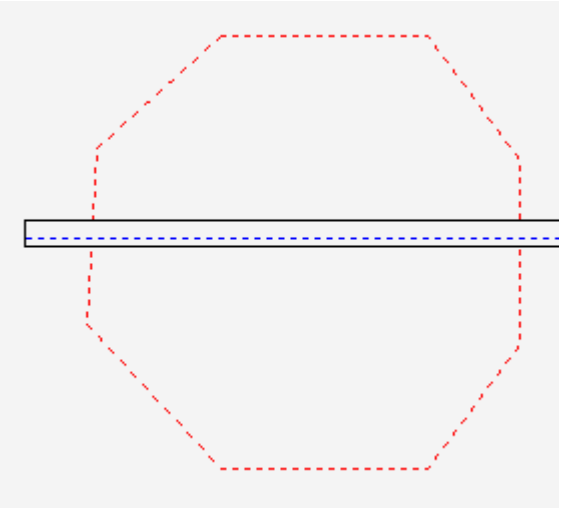
This makes it possible to lay connections on models, i.e. mounting plates, from the front to the back.

In order to lay conductors/wires through a cutout area when connecting automatically, place one cable duct on each slot of a device, near the cutout area and align it in such a way that one of the ends touches the cut out area or lies on it:

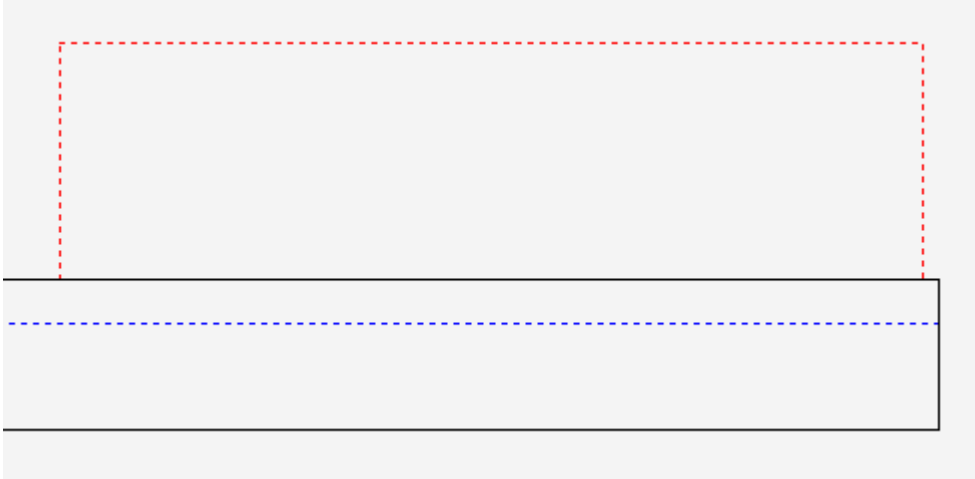


Please take note of the following conditions for the placement of the cable ducts so that automatically created connections can be laid through cutout areas:

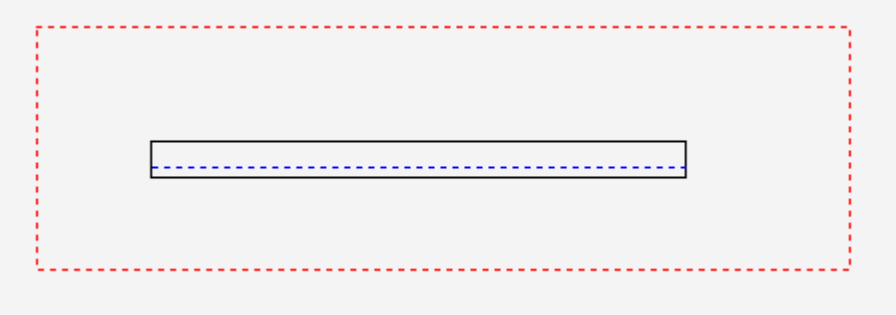
- Cable ducts either have to be placed in different devices or, if they are placed in the same device, have to be placed on slots with different directions.
- Cable ducts cannot lie across the cutout area:



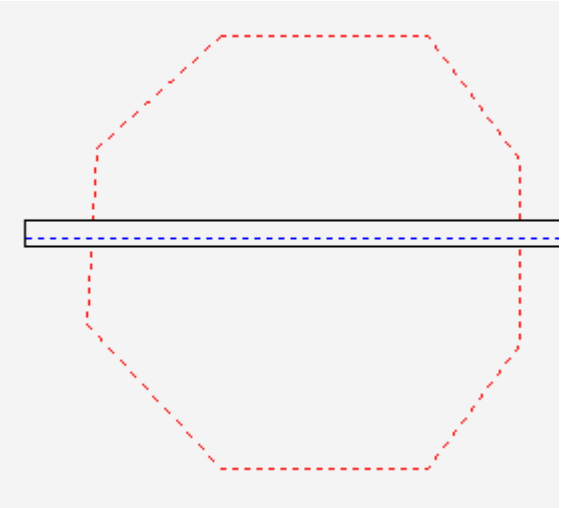
- Cable ducts cannot lie parallel to the cutout area:



- Cable ducts cannot lie completely on the cutout area:



- For each cutout area, through which a connection is to be created, at least one cable duct for the entry and at least one for the exit must be placed.



Note: In the 2D mode it is possible that segments of wires and docking points are displayed, even though they are hidden by other objects.
For a clearer view, please use the 2D projection or the 3D mode.

References: `Designer-17295` and `E3-302`

Placing components without mounting rail description on slots

Components can be optionally placed on slots even if they do not have mounting rail descriptions.

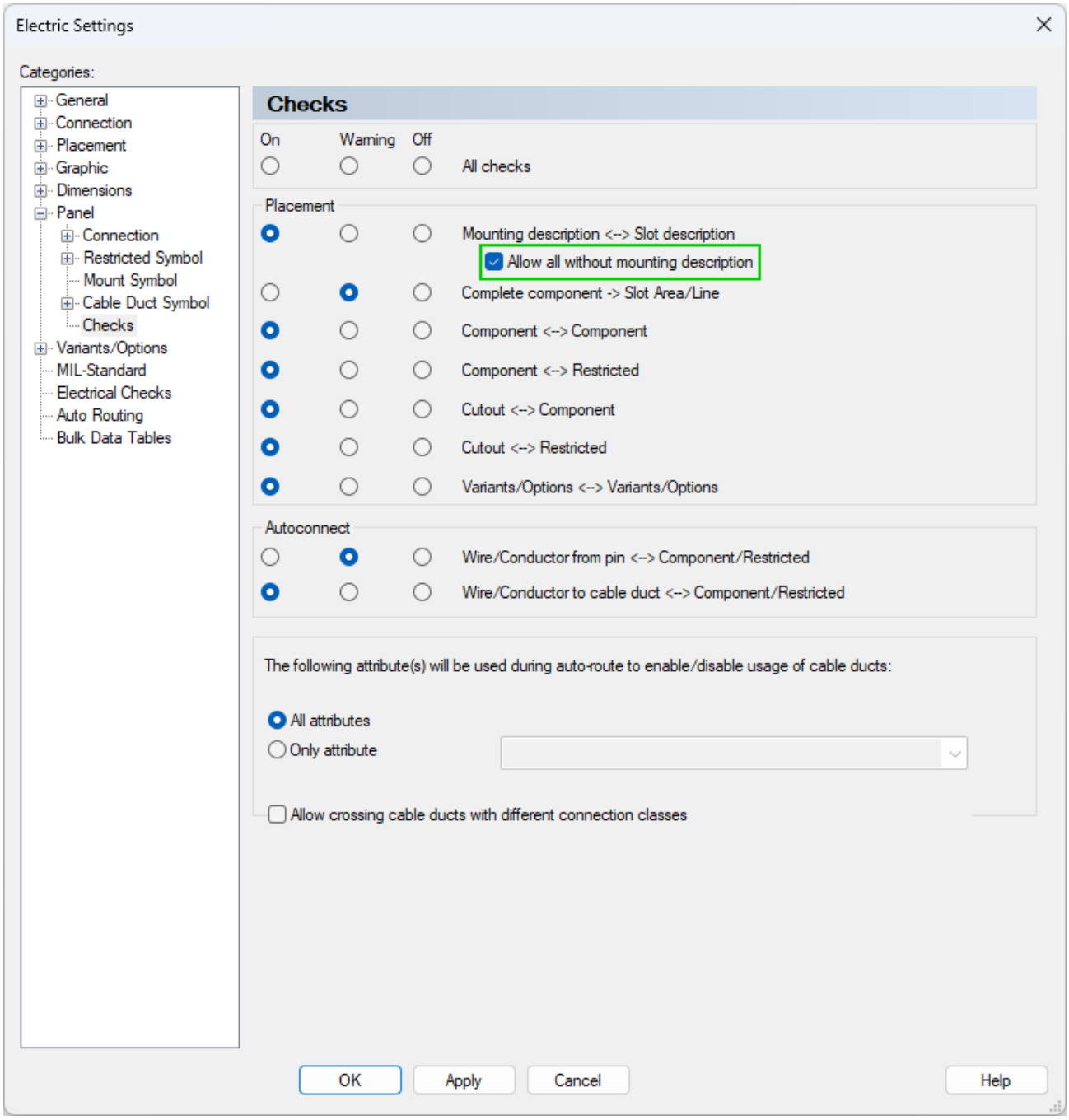
This extends the check for matching mounting rail descriptions so that components without a mounting rail description can also be placed.
As a result, it is possible to use components that have been imported from third-party systems into E3.series and do not contain mounting rail descriptions.

Select ***Tools → Settings...*** in the main menu window in order to be able to place components without mounting rail descriptions on slots.

The ***Electric Settings*** opens.

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

Under ***Panel → Checks → Checks → Mounting description <--> Slot description***, set the check on either ***Warning*** or ***Off*** and, in addition, select the ***Allow all without mounting description*** setting.
The setting is user specific:



Afterwards, components without mounting rail description can be placed on slots.

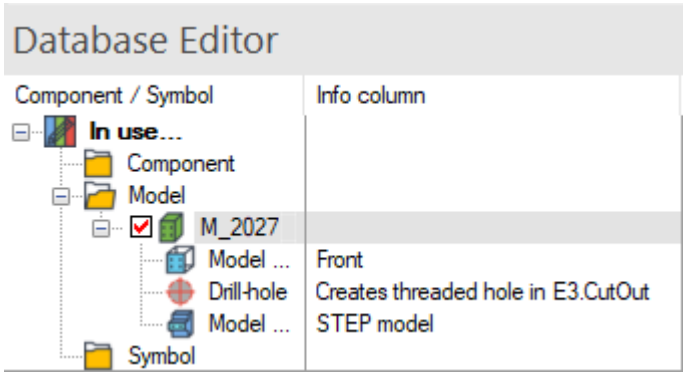
Enhancement of handling of threaded holes

Contours of the type ***Drill-hole*** and the property of creating threaded holes in E3.CutOut are realized as threaded holes on the reference object of the model upon manufacture.

From E3.series Version 2026, this property can be added directly to contours of the type ***Drill-hole***. In connection with this enhancement, further optimizations have been added to improve the handling of such contours.

This allows for easier identification and modification of drill holes that create threaded holes in E3.CutOut.

In tree views these contours are displayed with the icon for drill holes and an additional text in the info column.



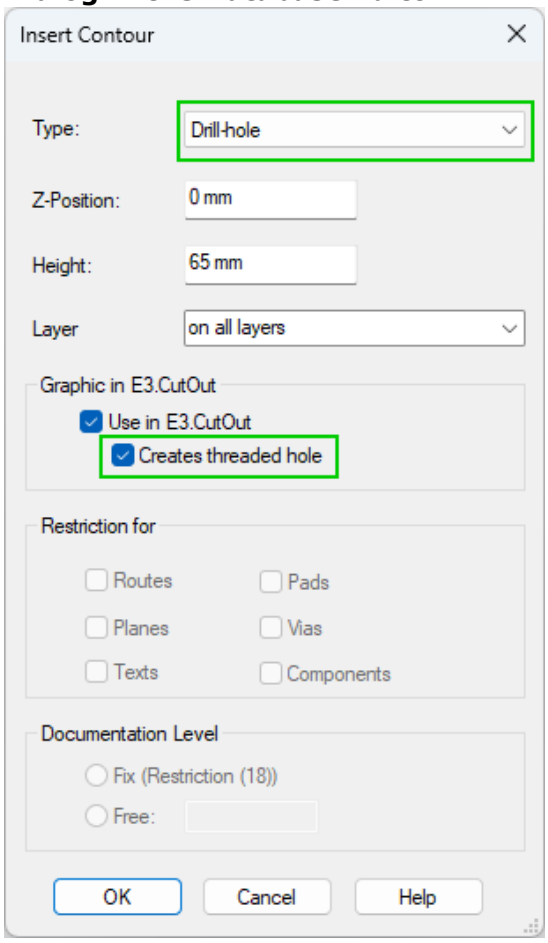
The following options exist for defining contours as threaded holes:

1. **Adding contours that create threaded holes on reference objects in the project or the database**

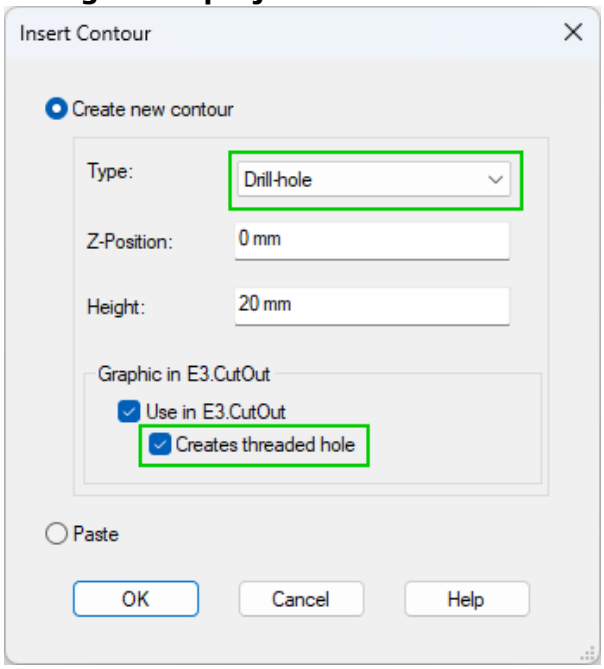
To add drill holes to a model in the project or the database, that create threaded holes on the reference object of the model, select the model in the work space of E3.series. Then select the ***Contour...*** command on the toolbar, the context menu or the main menu.

The ***Insert Contour*** dialog opens:

Dialog in the Database Editor



Dialog in the project



Select the contour type ***Drill-hole***, activate the ***Creates threaded hole*** checkbox and define the other properties of the contour.
Then, place the contour on the model.

Note: ***Creates threaded hole*** can only be activated if the checkbox ***Use in E3.CutOut*** is also active.

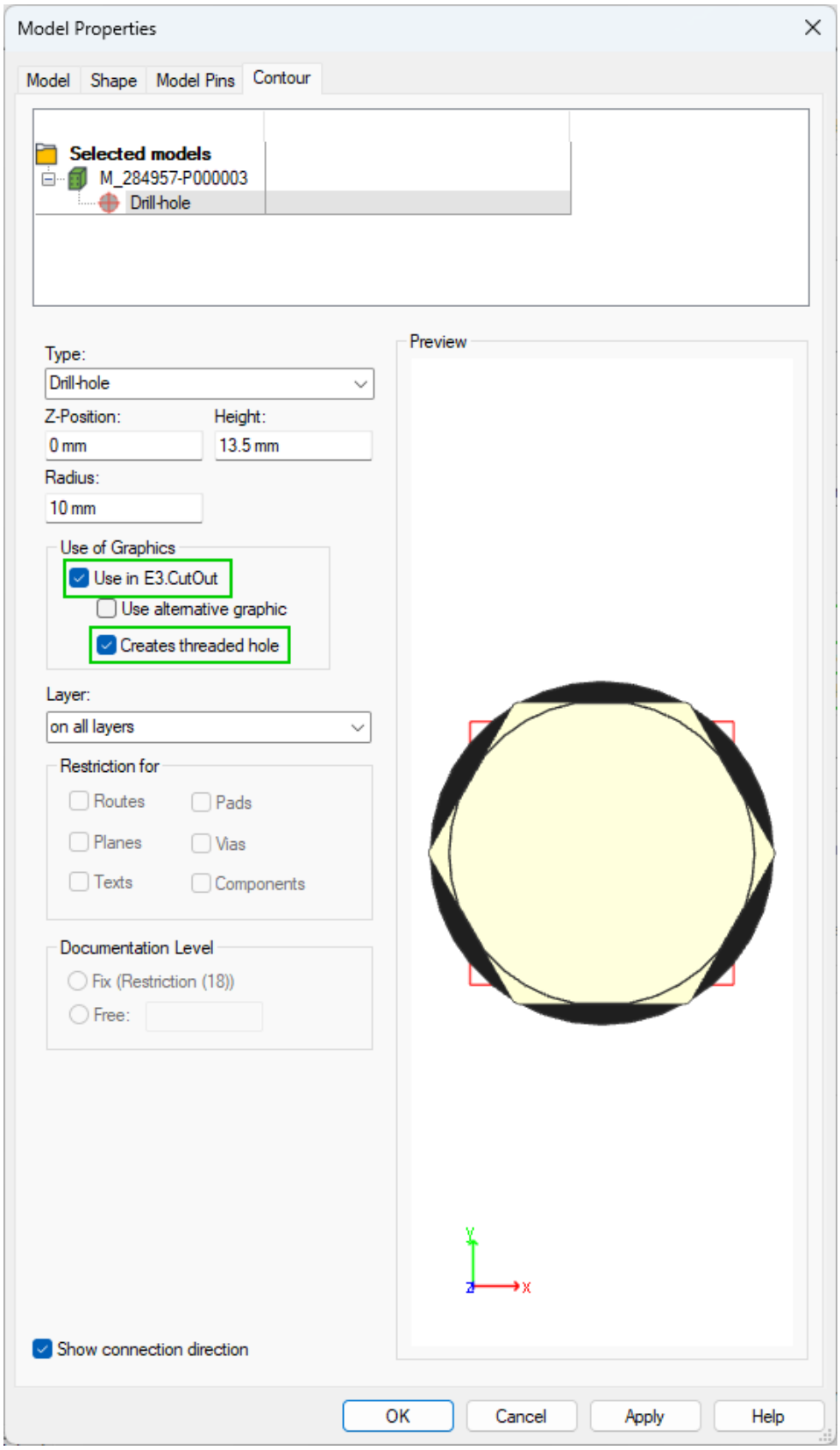
---

2. **Converting existing drill holes to contours that create threaded holes on reference objects**

In order to convert drill holes already defined to contours that create threaded holes on reference objects, edit the model with the drill hole in the Database Editor.

Next, open the model properties in the work space or the tree view via the context menu and select the ***Contour*** tab.
Activate the ***Use in E3.CutOut*** checkbox if it is not already activated.

Finally, select all required drill holes and activate the ***Creates threaded hole*** checkbox:



Note: The ***Use alternative graphic*** checkbox cannot be active at the same time.

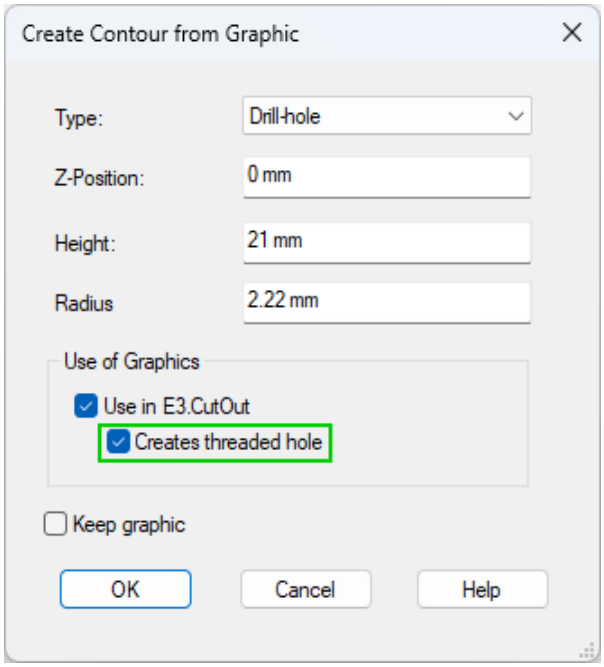
3. **Creating contours that create threaded holes on reference objects from existing graphics**

In order to create contours that create threaded holes on reference objects from circle graphics and circular polygons, select a model with a circular graphic in the Database Editor and select all graphics that are to be converted to drill holes.

Note: Circular polygons are polygons with at least 15 points, where all 15 points have the same distance to the center of the polygon. In addition, the distance between the first and last point of the polygon must not be greater than the distance between all other points.

Select circle graphics or circular polygons and select the command ***Insert → Contour...*** from the main menu bar, or click on the ***Insert Contour*** button in the ***Database Model*** toolbar.

The ***Create Contour from Graphic*** dialog opens:



Activate the ***Creates threaded hole*** check box.

All other values in the dialog are pre-populated:

- ***Z-Position:*** and ***Height:*** correspond to the values of the model to which the selected objects belong.
- ***Radius*** specifies the radius with which the threaded hole is created. The value is based on the radius of the circular graphic.
  When multiple graphic objects are selected, ***<multiple values>*** is displayed. All threaded holes are created with the radius of the graphic object on which it is based.

References: `Designer-43297`

Changing slot justification and description in the project

The values of the slot justification as well as the slot and mounting description can be changed within the project and can thus differ from the component on which it is based.

This allows for dynamic modification of placement of objects on slots based on the specific requirements in the project without having to modify the component in the Database Editor.

In order to change slot justifications within the project, select the device, whose alignment is to be changed, on a sheet or in the tree view.

Note: The justification can only be changed if neither the device nor the objects placed on it are locked.
The justification can also not be changed if all variants are activated in the project and the device is placed or has variant positions.

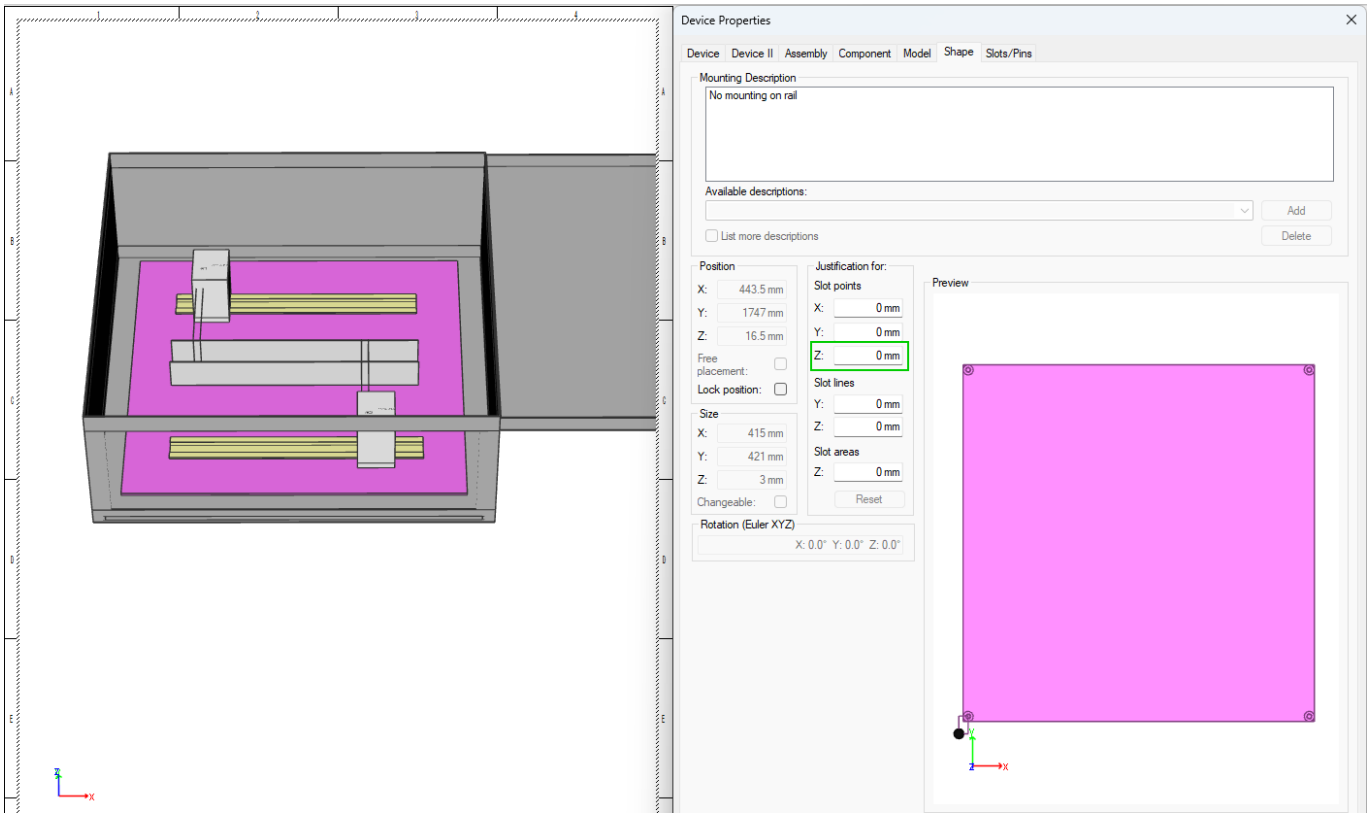
Then, open the context menu and select ***Device Properties...***
The device properties will open.

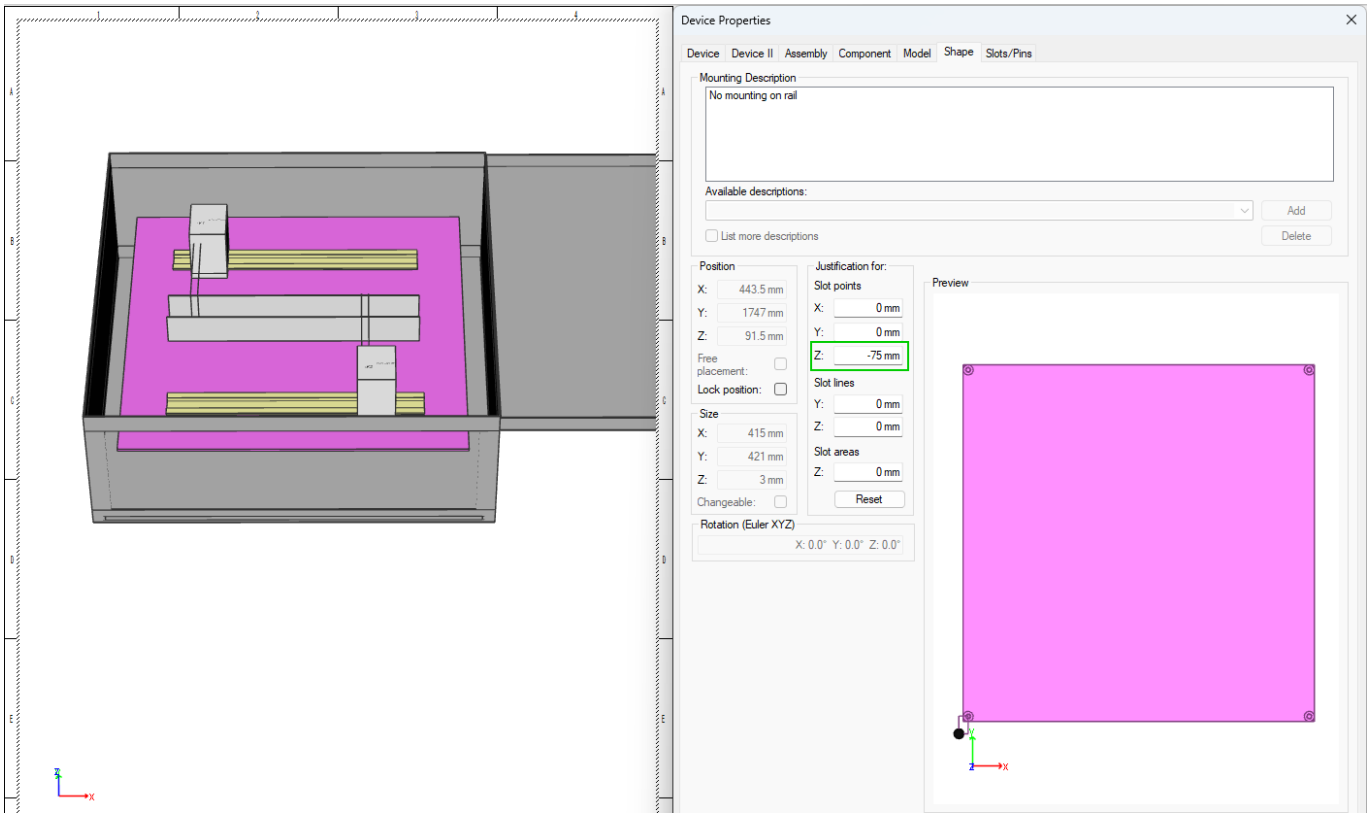
Next, go to the ***Shape*** tab, change the justification values and confirm the modification with ***Apply*** or ***Ok***.
The values are changed, objects already placed on the component are moved and existing connections are rerouted.

In order to reset the slot justification to the values that are defined for the component in the Database, click on ***Reset***.

Values that have been changed in the project are not reset when using the ***Update in Project*** function or changing the component in the Database Editor.

Example:





References: `Designer-17711` and `E3-1270`

In order to change slot or mounting rail descriptions in the project, select the device, whose description is to be changed, on a sheet or in the tree view.

Note: The slot and mounting rail descriptions can only be changed if neither the device nor the objects placed on it are locked.

Open the context menu and select ***Device Properties...***
The device properties will open.

To change the slot descriptions, select ***Slots/Pins*** tab.
To change the mounting rail descriptions, select the ***Shape*** tab.

If the setting ***Panel → Checks → Mounting description <--> Slot description*** is ***On*** and a device is placed on the slot, only the descriptions of those devices, that are placed on the slot, are available.

For the change of the slot description, select the required description in the selection list or enter a new description.
For the change of the mounting rail description, select the required description in the selection list or enter a new description.
Expand the description with ***Add***.
Confirm the changes with ***Apply*** or ***Ok***.

Values, that are changed in the project, are not reset when using the ***Update in Project*** function or when changing the component in the database editor.

References: `Designer-37804` and `E3-3599`

Connections/Busses, Signals/Logic Lines, Supplies

Setting for connectors for using higher level assignment and location of placed devices

With settings it is possible to control whether connectors take on the higher level assignment and location the devices connected to them. It is possible to define whether just the higher level assignment, just the location or both pieces of information are assigned.
In previous versions of E3.series it was only possible to assign either both designations or neither.

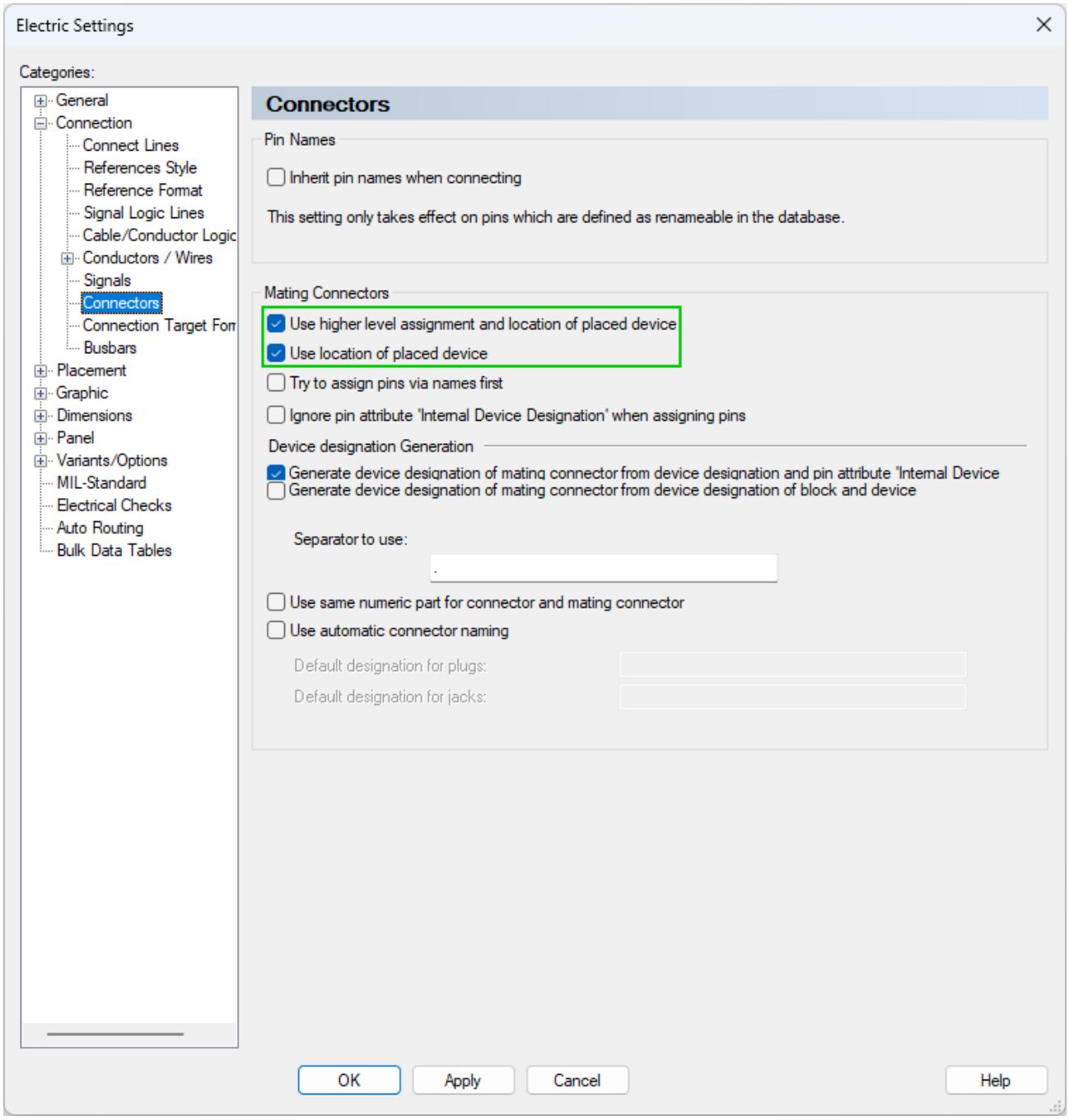
This allows for a more flexible automatic assignment of higher level assignment and location and makes adaption to the project requirements more easy.

In order to configure the automatic assignment of higher level assignment and location for connectors, select ***Tools → Settings...*** in the main menu.

The ***Electric Settings*** dialog opens.

Note: The setting can be adjusted via the ***Electric Settings*** and the ***Fluid Settings*** dialog.

Next activate the setting ***Use higher level assignment of placed device*** or ***Use location of placed device*** under ***Connection → Connectors***.
In the***Fluid Settings*** dialog the setting can be found in the ***Fitting*** tab.
The settings are user-specific:



With the settings active, the designation is assigned to the connectors of the connected device in the following cases:

- When a connection is created using the ***Connection*** command and connectors are generated automatically.
- When a connector and the connected device already have the same higher level assignment or location and the designation of the device is changed.
- When connector devices are created that are plugged onto a placed device.

Note: For projects, that are opened using E3.series 2026, were created in a previous version and have an automatic assignment of higher level assignment and location for connectors, the settings ***Use higher level assignment of placed device*** and ***Use location of placed device*** are activated.

References: E3-1344

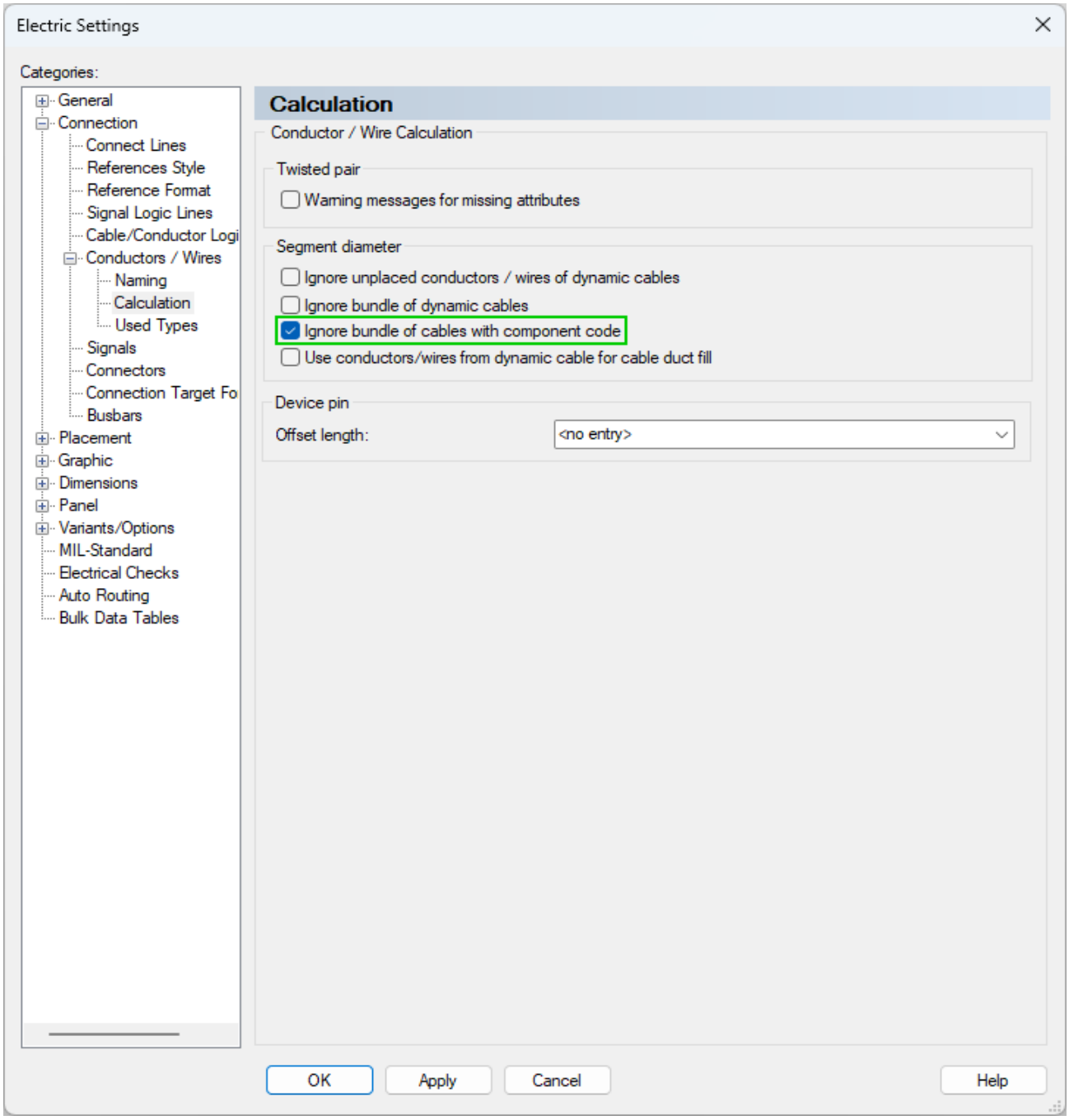
Ignoring cable bundles from the database when calculating the diameter of segments

Bundles, that are made up of cables and are created as devices in the database, can be ignored for the calculation of segment diameters.
In addition the settings relevant for the calculation of segment diameters, can be temporarily activated or deactivated in the dialog with the information on the segment diameter.

On the one hand, it is thus possible to completely exclude cable bundles form the calculation of segment diameters, for instance because they have to be processed further in the manufacture and would therefore distort the result of the calculation.
On the other hand, it is possible to simulate the calculation in the dialog depending on the settings without changing the project settings beforehand. Thus it is possible to calculate and compare the different results more easily.

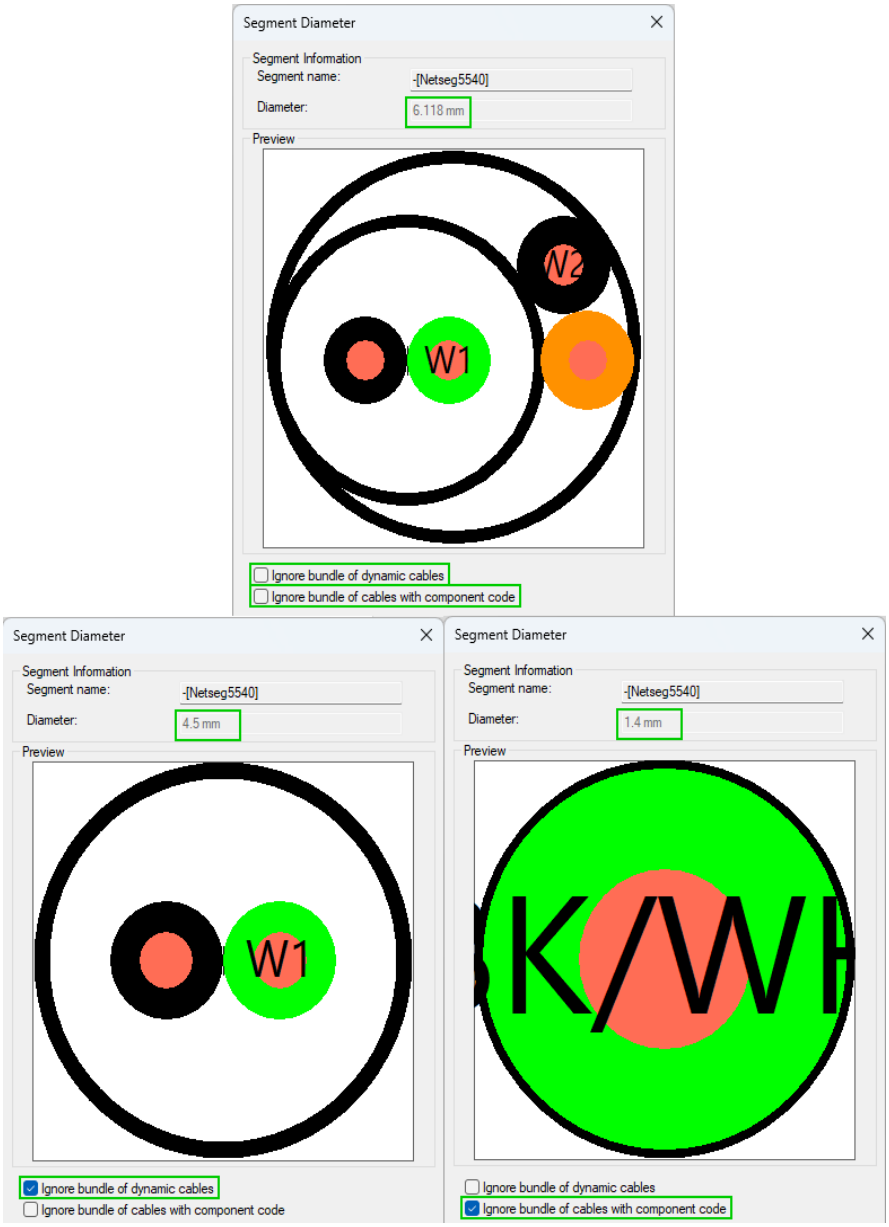
In order to exclude bundles, that are made up of cables form the database, from the calculation of segment diameters, select ***Tools → Settings...*** in the main menu.
The ***Electric Settings*** dialog opens.

Deactivate the setting ***Ignore bundle of cables with component code*** under ***Connection → (missing or bad snippet) → Calculation***.
The setting is project specific:



When the setting is active, cable bundles that are saved as components in the database, are ignored for the calculation of segment diameters.

In order to see the different results of the segment diameter calculation in direct comparison, open the context menu of a segment and select the ***Segment Diameter*** command in the context menu.
The dialog ***Segment Diameter*** opens:



The settings ***Ignore bundle of dynamic cables*** and ***Ignore bundle of cables with component code*** can be activated and deactivated temporarily in order to compare the changes of segment diameters.
When the dialog is closed, the status is reset to the value defined in the ***Electric Settings***.

References: E3-1323

Hiding connections without connected conductors in the connection table

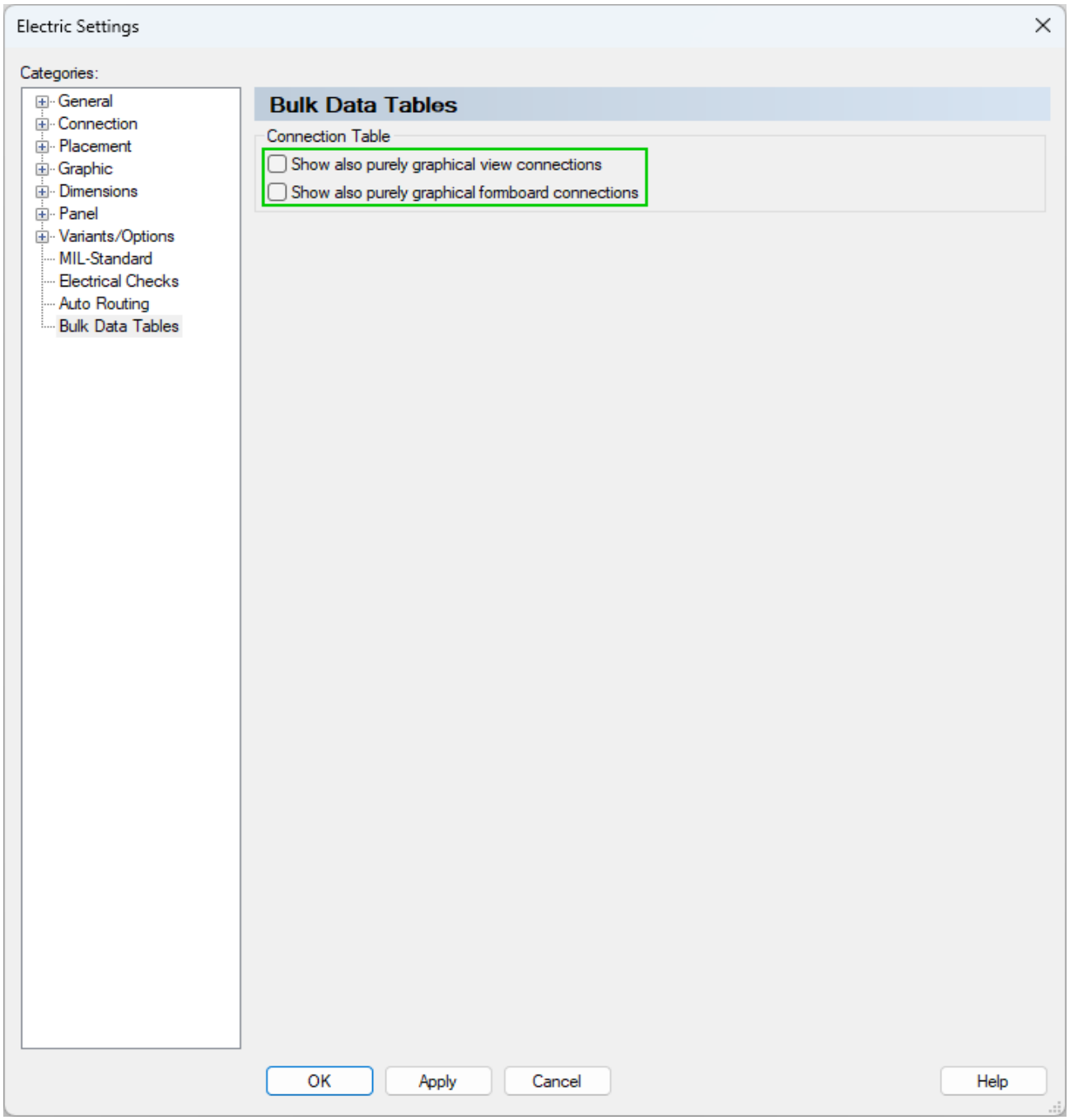
Connections, that are only displayed graphically and do not have any connected conductors/wires, can be hidden in the connection table with a setting.

This makes it possible to exclude connections without physical properties from the connection table, improving its clarity.
In bigger projects this also improves the efficiency of working with connection tables.

In order to exclude purely graphical connections from the connection table, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

Note: The setting can be adjusted via the ***Electric Settings*** and the ***Fluid Settings*** dialog.

Deactivate the required setting under ***Bulk Data Tables***.
The setting is user specific. By default it is active:



When the setting ***Show also purely graphical formboard connections*** is deactivated, graphical formboard connections are excluded from the connection table.

When the setting ***Show also purely graphical view connections*** is deactivated, graphical formboard connections are excluded from the connection table.

References: `Designer-41913` and `E3-2168`

Highlighting connected cable ducts

Using new commands all cable ducts can be highlighted that are connected to a selected cable duct. It is possible to highlight only cable ducts that are directly connected as well as those that are connected indirectly via another cable duct.

This makes it easier for instance to solve routing problems caused by cable ducts that are not connected.

It is possible to highlight connected cable ducts via panel sheets, the tree view and the device table. The color used for highlighting is the one defined under ***General → Highlight → Color:***

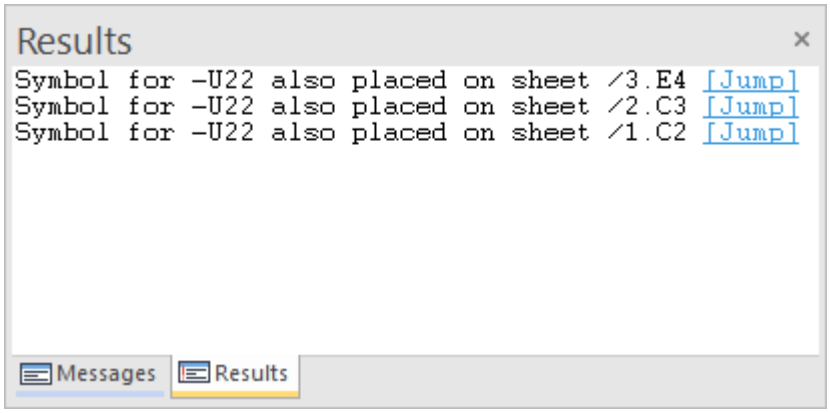
Cable duct pairs, that are defined as input and output, are treated as a single cable duct.

Note: If the selected cable duct or one of the connected cable ducts has a variant position and all variants/options are active at the same time.

In order to highlight cable ducts, that are connected to a specific cable duct, open the context menu of the required cable duct on the panel sheet , the tree view ***Panel - Placed*** or in the device table.

Then select the ***Highlight directly connected cable ducts*** ( ) command in the context menu to highlight all cable ducts, that are directly connected to the selected cable duct.

Or select the ***Highlight all connected cable ducts*** ( ) command in the context menu to also highlight all cable ducts, that are directly or recursively connected to a directly connected cable duct.
If the cable duct is placed on multiple sheets all sheets on which the selected cable duct is placed, are output in the output window under ***Results*** with a link to the respective sheet:



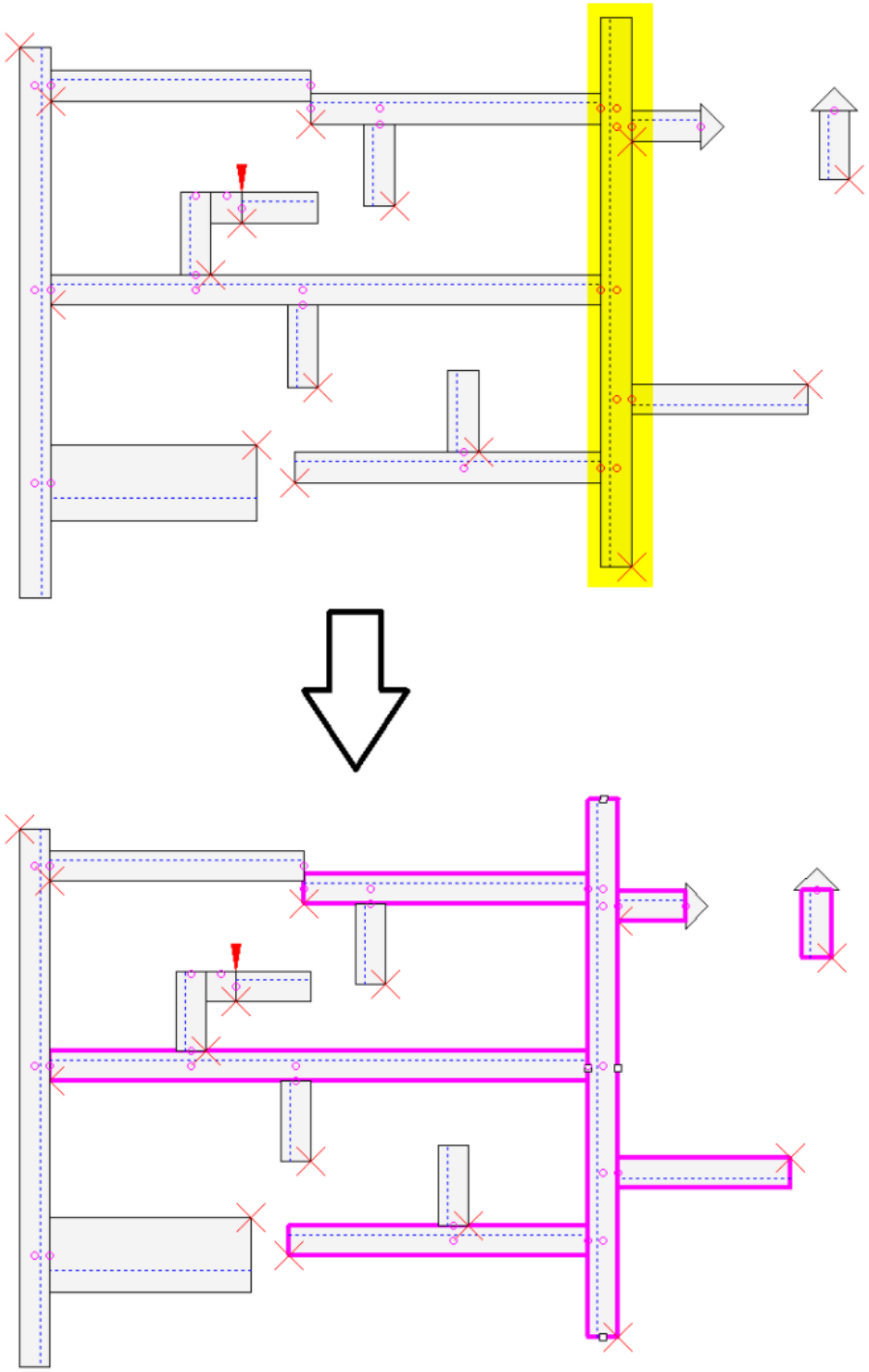
If connected cable ducts are placed in sheets, that are not open, the respective sheets are opened and the connected cable ducts highlighted and put into focus.

There are restrictions for cable ducts that are placed on shared sheets or on 2D view sheets.
Cable ducts, that are placed on shared sheets, are highlighted but not put into focus. The respective shared sheet is not opened.
Cable ducts, that are placed on 2D view sheets, are only highlighted and put into focus, if the respective sheet was created using ***Create new model view*** .

1. **Example: show all cable ducts that are connected directly to the selected cable duct**

The command ***Highlight directly connected cable ducts*** is executed starting from the cable duct marked in yellow.

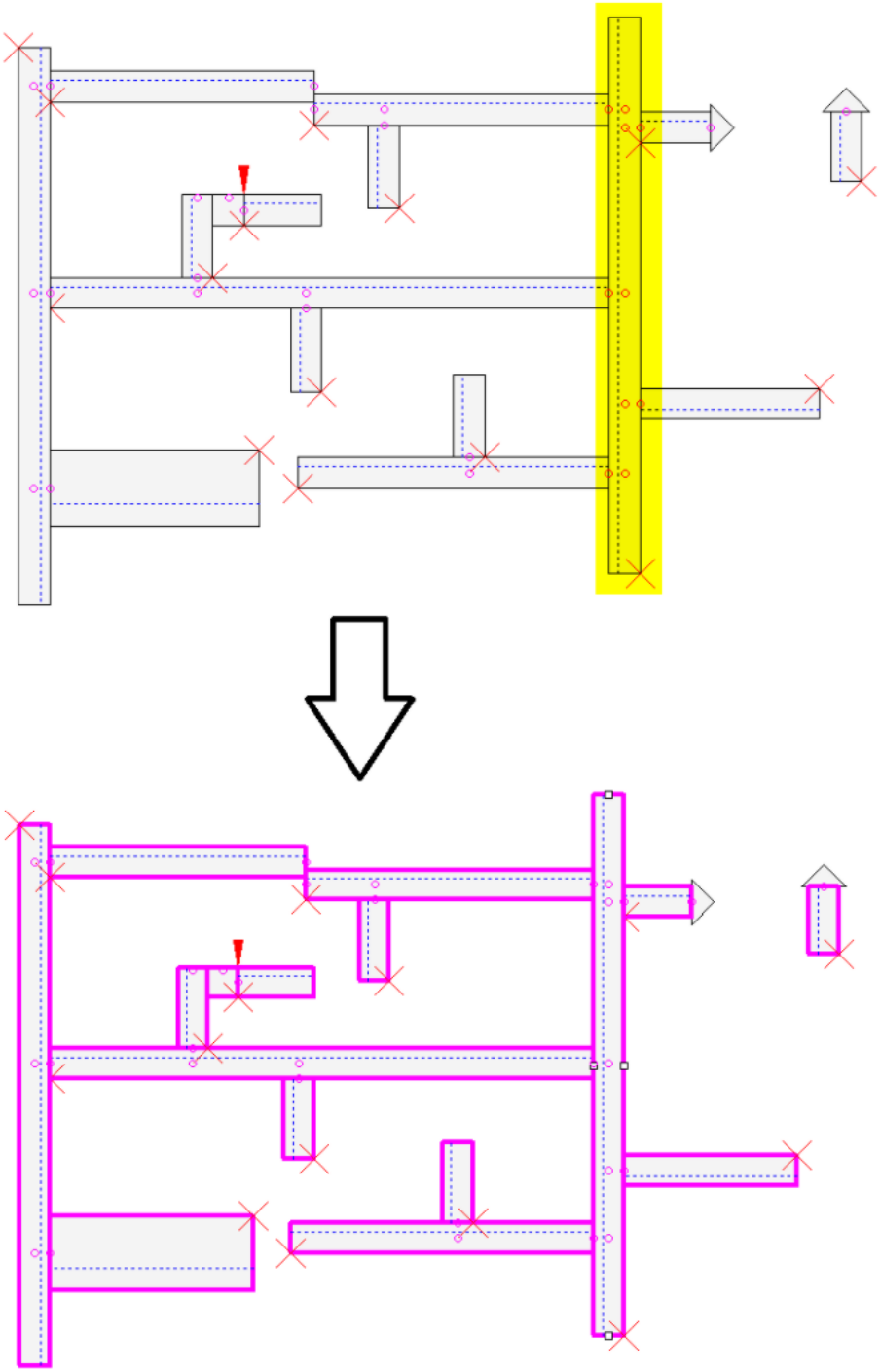
All cable ducts, that are directly connected to the cable duct, are highlighted in purple and put into focus:



2. **Example: Highlighting all cable ducts that are connected directly or recursively via a cable duct that is connected directly to the selected cable duct**

The command ***Highlight all connected cable ducts*** is executed starting from the cable duct marked in yellow.

All cable ducts, that are connected directly or recursively via a cable duct that is connected directly to the selected cable duct, are highlighted in purple and put into focus:



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