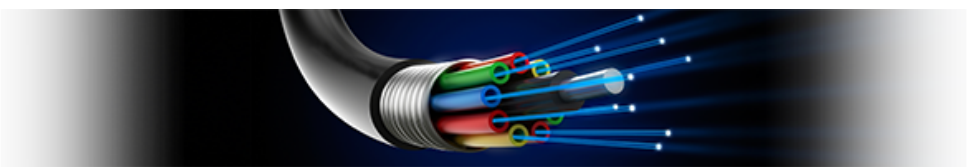
ZUKEN®

E3 series

New Features 2027-27.00



Disclaimer

Technical Inquiries
Please contact our Support Department!
E-Mail: e3-support@de.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
Meininger Allee 5
D-89231 Neu-Ulm

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

(c) Zuken E3 GmbH. All rights reserved.

Table of Contents

E3.series Release Notes 1

Update Information of Version 2027 2

Windows Support 2

Installation 2

Databases 3

Projects 3

Multi-User Installation 3

ORACLE 3

Microsoft SQL Server 3

Using an Existing Work Environment 4

New Features in Version 2027 Build 27.00 5

Overview - New Features in COM-Interface 9

COM-Interface - Enhancements and Changed Behavior of Interfaces and Functions 11

Customer requirement: Enhancement in e3DbeModel, e3DbeSymbol and e3Sheet - Set measurement units when importing DXF/DWG files (ImportDXF) 11

Enhancement in e3Component - New return value in GetFormboardSymbolIds, when no component is selected 12

Enhancement in e3DbeApplication - New return value in EditComponent, EditComponentWithPreconditions, EditModel and EditSymbol, when an internal error occurs while editing an object 12

Enhancement in e3DbeModel - New return value in ImportStep, when using the start-up parameter /noimportstep 12

Enhancement in e3DbeSlot - New supported value in CreateAreaPolygon, CreateAreaRectangle, CreateLine and CreatePoint for flags to define the direction of slots 13

Enhancement in e3Device - New return values in Create2DView, when an error occurs while calling the function 13

Enhancement in e3Field - New return values in Place, when error occurs while moving a field and its contents to a different sheet 13

Enhancement in e3Job - New supported value in ExportDrawing for options to export busbars 14

Enhancement in e3Pin - New return values in GetCCT and GetFunc, when an error occurs while calling the function 14

Enhancement in e3Symbol - New return value in GetHyperlinkTextIds, when the ID of an unplaced symbol is used 14

Enhancement in e3Tree - New supported value in GetVisibleObjectTypes and SetVisibleObjectTypes for type_arrayto query or set the display of groups 14

COM-Interface - New Interfaces and Functions 16

Customer requirement: New functions in e3Application and e3DbeApplication - Import and export configuration file with supported keyboard shortcuts (ExportKeyboardShortcuts and ImportKeyboardShortcuts) 16

Customer requirement: New functions in e3AttributeDefinition - Query information about attribute lists (GetAttributeListValues and GetValueListName) 18

Customer requirement: New functions in e3DbeJob - Display text in output window (GetOutbarText, GetOutbarTextEx, GetResultText and GetResultTextEx) 19

Customer requirement: New function in e3DbeModel, e3DbeSymbol and e3Sheet - Query measurement unit when importing DXF/DWG files (ImportDXF) 20

Customer requirement: New Interface e3SheetReferencefor working with sheet references 21

New function in the API MultiuserAdministrationTool - ArchiveProject 23

New function in e3Clipboard - SetName 23

New function in e3Group - UpdateDrawingForProjectGenerationEx 23

New function in e3Slot - GetDefinedRotation 25

New functions in e3DbeSlot - GetDirectionEx, GetLength, GetWidth, SetLength and SetWidth 25

New functions in e3Device - GetBomType, GetRouteCableInSamePath, JumpToPanel3D and SetRouteCableInSamePath 28

New functions in e3Job - Get3DAnimationEnabled and Set3DAnimationEnabled 30

New functions in e3Outline - CreateDynamicContour, CreateDynamicDrillHole and Delete 31

New functions in e3Pin - GetLogicalEquivalencePinIds, Get3DTransparency and Set3DTransparency 34

New functions in e3Sheet - GetPreferSinglePinPlacement and SetPreferSinglePinPlacement 35

General Changes 37

Enhancements to be able to use Python when working with E3.series 37

Project window for displaying grouped objects 40

Start-up parameters for opening the database editor without a project 42

Start-up parameters for starting E3.series processes without triggers ... 42
Database Editor ... 43
Changing models of components ... 43
Checking the database for errors ... 43
The command 'Place at original position' is available again in the Database editor ... 43
Enhancements and adaptions for working with block connectors and connectors with inserts ... 44
Jumping to the model tree directly from components with models ... 45
Additional area for user-defined text types ... 46
Importing and Exporting Data ... 47
Importing Data ... 47
Exporting Data ... 47
Importing and exporting configuration files with supported keyboard shortcuts ... 47
Defining measurement unit when importing DXF/DWG files ... 50
Formboard ... 52
Command in the context menu for jumping to formboard sheets ... 52
Moving nodes automatically along with any attribute text template symbols placed on them ... 52
Multiuser ... 54
Archiving multiuser projects ... 54
Project Handling ... 57
Ignoring changes of text types when comparing projects ... 57
Defining 'FONTS.DAT' file with user-specific line types via a setting ... 57
Settings and text types for defining the format of time and date ... 59
Removing empty structure nodes from the project using 'Purge unused objects' ... 61
Placing multiple symbols of components and devices together ... 62
Placing connectors on sheets using drag & drop preferably as single pins ... 63
Text type for displaying bundle names ... 66
Text types for wires, conductors and cables support more characters ... 66
Enhancements for working with clipboards created by Paste Extended ... 67
Panel (also: 3D) ... 68

Creating 2D manufacturing views of devices, models and slots 68
Routing conductors of cables through cable ducts in the same path 71
Pins with different signals can be connected automatically 73
Prefer entry to cable ducts with same HLA and LOC as respective device when routing automatically 74
Enhancements for defining the visibility of devices, conductors and wires 75
Colored highlighting of conductors/wires, tubes/hoses and busbars that are not connected in the panel 77
Additional command for automatic routing and enhancement of statistic 78
Variants/Options 80
Jumping to the 'Variants/Options' tree from the context menu of objects with variants/options 80
Editing variants/options in the sheet and device table 81
Connections/Busses, Signals/Logic Lines, Supplies 82
Keeping bundle symbol types when updating or switching cables and overbraids 82
Highlighting intersecting, unconnected connection lines 82
Checking and preventing connections between pins with different pin numbers 84

E3.series Release Notes

Date: 5/7/2026

This manual describes the following functions of version:

2027 Build 27.00

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

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

- **New Features**

Update Information of Version 2027

- Windows Support
- Installation
- Databases
- Projects
- Multi-User Installation
- ORACLE
- Microsoft SQL Server
- Using an Existing Work Environment

Windows Support

The following Windows systems are supported with E3.series 2027:

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 2027 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 2027, so copies should be used forE3.series 2027.

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 2027, a project can no longer be opened with an earlier version.

Multi-User Installation

Existing multi-user projects from Version 2026 can directly be opened with Version 2027. They are automatically converted upon opening. However, this cannot be undone, and the projects can no longer be opened with Version 2026.

That’s why we recommend installing Version 2027 multi-user projects in parallel to the existing version, so that it’s also possible to still access these projects with E3.series Version 2026.

To transfer the projects to Version 2027, these projects must be saved in Version 2026 as Single-User projects. They can then be stored in the multi-user environment of Version 2027.

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

ORACLE

When setting up the E3.series-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.series 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 2027 Build 27.00

The following functionality is new in Version 2027 Build 27.00:

- Overview - New Features in COM-Interface
  - COM-Interface - Enhancements and Changed Behavior of Interfaces and Functions
    - Customer requirement: Enhancement in e3DbeModel, e3DbeSymbol and e3Sheet - Set measurement units when importing DXF/DWG files (ImportDXF)
    - Enhancement in e3Component - New return value in GetFormboardSymbolIds, when no component is selected
    - Enhancement in e3DbeApplication - New return value in EditComponent, EditComponentWithPreconditions, EditModel and EditSymbol, when an internal error occurs while editing an object
    - Enhancement in e3DbeModel - New return value in ImportStep, when using the start-up parameter /noimportstep
    - Enhancement in e3DbeSlot - New supported value in CreateAreaPolygon, CreateAreaRectangle, CreateLine and CreatePoint for flags to define the direction of slots
    - Enhancement in e3Device - New return values in Create2DView, when an error occurs while calling the function
    - Enhancement in e3Field - New return values in Place, when error occurs while moving a field and its contents to a different sheet
    - Enhancement in e3Job - New supported value in ExportDrawing for options to export busbars
    - Enhancement in e3Pin - New return values in GetCCT and GetFunc, when an error occurs while calling the function
    - Enhancement in e3Symbol - New return value in GetHyperlinkTextIds, when the ID of an unplaced symbol is used
    - Enhancement in e3Tree - New supported value in GetVisibleObjectTypes and SetVisibleObjectTypes for type_arrayto query or set the display of groups
  - COM-Interface - New Interfaces and Functions
    - Customer requirement: New functions in e3Application and e3DbeApplication - Import and export configuration file with

supported keyboard shortcuts (ExportKeyboardShortcuts and ImportKeyboardShortcuts)

- Customer requirement: New functions in e3AttributeDefinition - Query information about attribute lists (GetAttributeListValues and GetValueListName)
- Customer requirement: New functions in e3DbeJob - Display text in output window (GetOutbarText, GetOutbarTextEx, GetResultText and GetResultTextEx)
- Customer requirement: New function in e3DbeModel, e3DbeSymbol and e3Sheet - Query measurement unit when importing DXF/DWG files (ImportDXF)
- Customer requirement: New Interface e3SheetReferencefor working with sheet references
- New function in the API MultiuserAdministrationTool - ArchiveProject
- New function in e3Clipboard - SetName
- New function in e3Group - UpdateDrawingForProjectGenerationEx
- New function in e3Slot - GetDefinedRotation
- New functions in e3DbeSlot - GetDirectionEx, GetLength, GetWidth, SetLength and SetWidth
- New functions in e3Device - GetBomType, GetRouteCableInSamePath, JumpToPanel3D and SetRouteCableInSamePath
- New functions in e3Job - Get3DAnimationEnabled and Set3DAnimationEnabled
- New functions in e3Outline - CreateDynamicContour, CreateDynamicDrillHole and Delete
- New functions in e3Pin - GetLogicalEquivalencePinIds, Get3DTransparency and Set3DTransparency
- New functions in e3Sheet - GetPreferSinglePinPlacement and SetPreferSinglePinPlacement

---

- General Changes
  - Enhancements to be able to use Python when working with E3.series
  - Project window for displaying grouped objects
  - Start-up parameters for opening the database editor without a project
  - Start-up parameters for starting E3.series processes without triggers

---

- Database Editor
  - Changing models of components
  - Checking the database for errors
  - The command 'Place at original position' is available again in the Database editor
  - Enhancements and adaptions for working with block connectors and connectors with inserts
  - Jumping to the model tree directly from components with models
  - Additional area for user-defined text types
- Importing and Exporting Data
  - Importing and exporting configuration files with supported keyboard shortcuts
  - Defining measurement unit when importing DXF/DWG files
- Formboard
  - Command in the context menu for jumping to formboard sheets
  - Moving nodes automatically along with any attribute text template symbols placed on them
- Multiuser
  - Archiving multiuser projects
- Project Handling
  - Ignoring changes of text types when comparing projects
  - Defining 'FONTS.DAT' file with user-specific line types via a setting
  - Settings and text types for defining the format of time and date
  - Removing empty structure nodes from the project using 'Purge unused objects'
  - Placing multiple symbols of components and devices together
  - Placing connectors on sheets using drag & drop preferably as single pins
  - Text type for displaying bundle names
  - Text types for wires, conductors and cables support more characters
  - Enhancements for working with clipboards created by Paste Extended
- Panel (also: 3D)

- Creating 2D manufacturing views of devices, models and slots
- Routing conductors of cables through cable ducts in the same path
- Pins with different signals can be connected automatically
- Prefer entry to cable ducts with same HLA and LOC as respective device when routing automatically
- Enhancements for defining the visibility of devices, conductors and wires
- Colored highlighting of conductors/wires, tubes/hoses and busbars that are not connected in the panel
- Additional command for automatic routing and enhancement of statistic

---

- Variants/Options
  - Jumping to the 'Variants/Options' tree from the context menu of objects with variants/options
  - Editing variants/options in the sheet and device table

- Connections/Busses, Signals/Logic Lines, Supplies
  - Keeping bundle symbol types when updating or switching cables and over-braids
  - Highlighting intersecting, unconnected connection lines
  - Checking and preventing connections between pins with different pin numbers

---

Overview - New Features in COM-Interface

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

The Help integrated in E3.series offers a more detailed reference about new, modified or deprecated functionality in the chapter **New Functionality in v2027-27.00**:

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

- **Customer requirement: Enhancement in e3DbeModel, e3DbeSymbol and e3Sheet - Set measurement units when importing DXF/DWG files (ImportDXF)**
- **Enhancement in e3Component - New return value in GetFormboardSymbolIds, when no component is selected**
- **Enhancement in e3DbeApplication - New return value in EditComponent, EditComponentWithPreconditions, EditModel and EditSymbol, when an internal error occurs while editing an object**
- **Enhancement in e3DbeModel - New return value in ImportStep, when using the start-up parameter /noimportstep**
- **Enhancement in e3DbeSlot - New supported value in CreateAreaPolygon, CreateAreaRectangle, CreateLine and CreatePoint for flags to define the direction of slots**
- **Enhancement in e3Device - New return values in Create2DView, when an error occurs while calling the function**
- **Enhancement in e3Field - New return values in Place, when error occurs while moving a field and its contents to a different sheet**
- **Enhancement in e3Job - New supported value in ExportDrawing for options to export busbars**
- **Enhancement in e3Pin - New return values in GetCCT and GetFunc, when an error occurs while calling the function**
- **Enhancement in e3Symbol - New return value in GetHyperlinkTextIds, when the ID of an unplaced symbol is used**
- **Enhancement in e3Tree - New supported value in GetVisibleObjectTypes and SetVisibleObjectTypes for type_arrayto query or set the display of groups**

COM-Interface - New Interfaces and Functions

- **Customer requirement: New functions in e3Application and e3DbeApplication - Import and export configuration file with supported keyboard shortcuts (ExportKeyboardShortcuts and ImportKeyboardShortcuts)**
- **Customer requirement: New functions in e3AttributeDefinition - Query information about attribute lists (GetAttributeListValues and GetValueListName)**
- **Customer requirement: New functions in e3DbeJob - Display text in output window (GetOutbarText, GetOutbarTextEx, GetResultText and GetResultTextEx)**

- Customer requirement: New function in e3DbeModel, e3DbeSymbol and e3Sheet - Query measurement unit when importing DXF/DWG files (ImportDXF)
- Customer requirement: New Interface e3SheetReferencefor working with sheet references
- New function in the API MultiuserAdministrationTool - ArchiveProject
- New function in e3Clipboard - SetName
- New function in e3Group - UpdateDrawingForProjectGenerationEx
- New function in e3Slot - GetDefinedRotation
- New functions in e3DbeSlot - GetDirectionEx, GetLength, GetWidth, SetLength and SetWidth
- New functions in e3Device - GetBomType, GetRouteCableInSamePath, JumpToPanel3D and SetRouteCableInSamePath
- New functions in e3Job - Get3DAnimationEnabled and Set3DAnimationEnabled
- New functions in e3Outline - CreateDynamicContour, CreateDynamicDrillHole and Delete
- New functions in e3Pin - GetLogicalEquivalencePinIds, Get3DTransparency and Set3DTransparency
- New functions in e3Sheet - GetPreferSinglePinPlacement and SetPreferSinglePinPlacement

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 2027 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 requirement: Enhancement in e3DbeModel, e3DbeSymbol and e3Sheet - Set measurement units when importing DXF/DWG files (ImportDXF)**
- **Enhancement in e3Component - New return value in GetFormboardSymbolIds, when no component is selected**
- **Enhancement in e3DbeApplication - New return value in EditComponent, EditComponentWithPreconditions, EditModel and EditSymbol, when an internal error occurs while editing an object**
- **Enhancement in e3DbeModel - New return value in ImportStep, when using the start-up parameter /noimportstep**
- **Enhancement in e3DbeSlot - New supported value in CreateAreaPolygon, CreateAreaRectangle, CreateLine and CreatePoint for flags to define the direction of slots**
- **Enhancement in e3Device - New return values in Create2DView, when an error occurs while calling the function**
- **Enhancement in e3Field - New return values in Place, when error occurs while moving a field and its contents to a different sheet**
- **Enhancement in e3Job - New supported value in ExportDrawing for options to export busbars**
- **Enhancement in e3Pin - New return values in GetCCT and GetFunc, when an error occurs while calling the function**
- **Enhancement in e3Symbol - New return value in GetHyperlinkTextIds, when the ID of an unplaced symbol is used**
- **Enhancement in e3Tree - New supported value in GetVisibleObjectTypes and SetVisibleObjectTypes for type_arrayto query or set the display of groups**

Customer requirement: Enhancement in *e3DbeModel*, *e3DbeSymbol* and *e3Sheet* - Set measurement units when importing DXF/DWG files (*ImportDXF*)

The following functions were enhanced in the interfaces ***e3DbeModel***, ***e3DbeSymbol*** and ***e3Sheet*** to set the measurement unit with which values of DXF/DWG files are interpreted:

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

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

return = e3Sheet.ImportDXF (filename, scale, x, y, rot, font, flags)

- **flags** `[in][optional]`: The parameter supports the following new values:
  - **0x100** (Decimal value **256**): Interprets values in millimeters.
  - **0x200** (Decimal value **512**): Interprets values in inches.
- The functions **e3DbeModel.ImportDXF** and **e3DbeSymbol.ImportDXF** have the following new return value:
  - **return -8**: Failed calling the function - **flags** cannot contain both values 0x100 and 0x200 at the same time.

References: `Designer-41248` and `E3-2964`

Enhancement in *e3Component* - New return value in *GetFormboardSymbolIds*, when no component is selected

The following function in the interface ***e3Component*** supports a new return value, when no component is selected:

return = e3Component.GetFormboardSymbolIds (ids)

- The function has the following new return value:
  - **return -1**: Failed calling the function - No component selected.

Enhancement in *e3DbeApplication* - New return value in *EditComponent*, *EditComponentWithPreconditions*, *EditModel* and *EditSymbol*, when an internal error occurs while editing an object

The following function in the interface ***e3DbeApplication*** supports a new return value, when an internal error occurs while editing an object:

return = e3DbeApplication.EditComponent (name, version)

return = e3DbeApplication.EditComponentWithPreconditions (name, version, preconditions, flags)

return = e3DbeApplication.EditModel (name)

return = e3DbeApplication.EditSymbol (name, version)

- The functions have the following new return value:
  - **return -1**: Failed calling the function - An internal error occurred.

Enhancement in *e3DbeModel* - New return value in *ImportStep*, when using the start-up parameter `/noimportstep`

The following function in the interface ***e3DbeModel*** supports a new return value, when using the start-up parameter `/noimportstep`:

return = e3DbeModel.ImportStep (filename, flags)

- **return -13**: Failed calling the function - Cannot import, because E3.series was started with the start-up parameter `/noimportstep`.

Enhancement in *e3DbeSlot* - New supported value in *CreateAreaPolygon*, *CreateAreaRectangle*, *CreateLine* and *CreatePoint* for *flags* to define the direction of slots

The following functions in the interface ***e3DbeSlot*** support a new value **flags** for defining the direction of slots:

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

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

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

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

- **flags** `[in][optional]`: The parameter supports the following new values:
  - **0x0000** (Decimal value **0**): The direction is set based one the values that are defined for **xdir**, **ydir** and **zdir**.
  - **0x0001** (Decimal value **1**): The direction, that is most plausible, is set based one the values defined for **xpos**, **ypos** and **zpos** or **xarr**, **yarr** and **zarr**.

Enhancement in *e3Device* - New return values in *Create2DView*, when an error occurs while calling the function

The following function in the interface ***e3Device*** supports new return values, when an error occurs while calling the function:

return = e3Device.Create2DView (modi, name, symbol, position, before)

- The function has the following new return values:
  - **return -2**: Failed calling the function - The value defined for **name** is invalid.
  - **return -3**: Failed calling the function - The value defined for **symbol** is invalid.
  - **return -4**: Failed calling the function - The value defined for **position** is invalid.

Enhancement in *e3Field* - New return values in *Place*, when error occurs while moving a field and its contents to a different sheet

The following function in the interface ***e3Field*** supports a new return value, when error occurs while moving a field and its content to a different sheet:

return = e3Field.Place (shtid, x1, y1, x2, y2, moveall)

- The function has the following new return values:
  - **return -1**: Failed calling the function - The connections do not lie completely within the field, so the field and its contents cannot be moved to a different sheet.

- **return -2**: Failed calling the function - Contents of the field are grouped with objects, that lie outside the field, so the field and its contents cannot be moved to a different sheet.

Enhancement in *e3Job* - New supported value in *ExportDrawing* for *options* to export busbars

The following function in the interface ***e3Job*** supports two new values for **options** to also output busbars when exporting drawings:

return = e3Job.ExportDrawing (name, shtids, options)

- **options** `[in]`: The parameter supports the following new values:
  - **0x00080000** (Decimal value **524288**): Busbars are also output when exporting.
  - **0x00100000** (Decimal value **1048576**): Busbars are only output when exporting, if all pins and net segments were selected.

Enhancement in *e3Pin* - New return values in *GetCCT* and *GetFunc*, when an error occurs while calling the function

The following functions in the interface ***e3Pin*** supports new return values, when an error occurs while calling the function:

return = e3Pin.GetCCT ( )

return = e3Pin.GetFunc ( )

- The function has the following new return values:
  - **return -1**: Failed calling the function - No project open or no object selected.
  - **return -2**: Failed calling the function - The selected object is invalid.

Enhancement in *e3Symbol* - New return value in *GetHyperlinkTextIds*, when the ID of an unplaced symbol is used

The following function in the interface ***e3Symbol*** supports a new return value, when the ID of an unplaced symbol is used:

return = e3Symbol.GetHyperlinkTextIds (ids)

- The function has the following new return value:
  - **return -1**: Failed calling the function - No ID of a placed symbol selected.

Enhancement in *e3Tree* - New supported value in *GetVisibleObjectTypes* and *SetVisibleObjectTypes* for *type_array*to query or set the display of groups

The following functions in the interface ***e3Tree*** support new return values for the parameter **type_array** to query or set the display of groups:

return = e3Tree.GetVisibleObjectTypes (type_array)

return = e3Tree.SetVisibleObjectTypes (type_array)

- The parameter supports the following new values:
  - **13**: The value corresponds to the checkbox ***Groups*** in the ***Tree Control Properties*** dialog.
  - **14**: The value corresponds to the checkbox ***Devices/Symbols/Pins/Models*** in the ***Tree Control Properties*** dialog.
  - **15**: The value corresponds to the checkbox ***Graphics/Texts/Dimensions*** in the ***Tree Control Properties*** dialog.
  - **16**: The value corresponds to the checkbox ***Connections*** in the ***Tree Control Properties*** dialog.
  - **16**: The value corresponds to the checkbox ***Fields*** in the ***Tree Control Properties*** dialog.

COM-Interface - New Interfaces and Functions

This chapter provides an overview of which interfaces and functions that are new in E3.series 2027.

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 requirement: New functions in e3Application and e3DbeApplication - Import and export configuration file with supported keyboard shortcuts (ExportKeyboardShortcuts and ImportKeyboardShortcuts)**
- **Customer requirement: New functions in e3AttributeDefinition - Query information about attribute lists (GetAttributeListValues and GetValueListName)**
- **Customer requirement: New functions in e3DbeJob - Display text in output window (GetOutbarText, GetOutbarTextEx, GetResultText and GetResultTextEx)**
- **Customer requirement: New function in e3DbeModel, e3DbeSymbol and e3Sheet - Query measurement unit when importing DXF/DWG files (ImportDXF)**
- **Customer requirement: New Interface e3SheetReferencefor working with sheet references**
- **New function in the API MultiuserAdministrationTool - ArchiveProject**
- **New function in e3Clipboard - SetName**
- **New function in e3Group - UpdateDrawingForProjectGenerationEx**
- **New function in e3Slot - GetDefinedRotation**
- **New functions in e3DbeSlot - GetDirectionEx, GetLength, GetWidth, SetLength and SetWidth**
- **New functions in e3Device - GetBomType, GetRouteCableInSamePath, JumpToPanel3D and SetRouteCableInSamePath**
- **New functions in e3Job - Get3DAnimationEnabled and Set3DAnimationEnabled**
- **New functions in e3Outline - CreateDynamicContour, CreateDynamicDrillHole and Delete**
- **New functions in e3Pin - GetLogicalEquivalencePinIds, Get3DTransparency and Set3DTransparency**
- **New functions in e3Sheet - GetPreferSinglePinPlacement and SetPreferSinglePinPlacement**

Customer requirement: New functions in *e3Application* and *e3DbeApplication* - Import and export configuration file with supported keyboard shortcuts (*ExportKeyboardShortcuts* and *ImportKeyboardShortcuts*)

The following functions were added to the interfaces ***e3Application*** and ***e3DbeApplication*** for exporting configuration files with supported keyboard shortcuts:

return = e3Application.ExportKeyboardShortcuts (filePath, flags)

return = e3DbeApplication.ExportKeyboardShortcuts (filePath, flags)

Parameter:

- **filePath** `[in]`: Specifies the path of the configuration file, in which the keyboard shortcut is defined.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether only user-defined keyboard shortcuts are exported.
  The parameter supports the following values:
  - **0x000** (Decimal value **0**): Exports a configuration file that contains all supported keyboard shortcuts.
  - **0x001** (Decimal value **1**): Exports only user-defined shortcuts. Shortcuts, that are already supported by E3.series with default settings, are not exported.
    When **flags** is not used, only user-defined keyboard shortcuts are exported.

Return values:

- **return 0**: Successfully called the function - The configuration file was exported.
- **return -1**: Failed calling the function - The configuration file **filePath** could not be created.
- **return -3**: Failed calling the function - An internal error occurred.

The following functions were added to the interfaces ***e3Application*** and ***e3DbeApplication*** for importing configuration files with supported keyboard shortcuts:

return = e3Application.ImportKeyboardShortcuts (filePath, flags)

return = e3DbeApplication.ImportKeyboardShortcuts (filePath, flags)

Parameter:

- **filePath** `[in]`: Specifies the path of the configuration file, in which the keyboard shortcut is defined.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether all existing keyboard combinations are to be deleted before importing.
  The parameter supports the following values:
  - **0x000** (Decimal value **0**): Deletes all existing keyboard combinations before importing.
    When **flags** is not used, existing keyboard combinations are not deleted.
  - **0x001** (Decimal value **1**): Deletes all existing keyboard combinations and imports all keyboard combinations defined in the configuration file.

Return values:

- **return 0**: Successfully called the function - The configuration file was imported.
- **return -1**: Failed calling the function - The configuration file **filePath** could not be read or is not in XML format.

- **return -2**: Failed calling the function - The configuration file **filePath** could be read, but has no valid syntax that defines keyboard combinations for E3.series .
- **return -3**: Failed calling the function - An internal error occurred.

References: E3-8133

Customer requirement: New functions in *e3AttributeDefinition* - Query information about attribute lists (*GetAttributeListValues* and *GetValueListName*)

The following function was added to the interface ***e3AttributeDefinition*** for querying information about attribute lists:

return = e3AttributeDefinition.GetAttributeListValues (attributelistname, values, flags)

Parameter:

- **attributelistname** `[in]`: Defines the attribute list whose values are to be queried.
- **values** `[out]`: Outputs the values of the attribute list.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether translatable texts are to be translated to the application language.
  The parameter supports the following values:
  - **0x000** (Decimal value **0**): Translatable texts are translated to the application language. When **flags** is not used, translatable texts are translated by default.
  - **0x001** (Decimal value **1**): Translatable texts are not translated to the application language.

Return values:

- **return ≥ 0**: Successfully called the function - The amount of values contained in the attribute list, is output.
- **return -1**: Failed calling the function - No project open.
- **return -2**: Failed calling the function - The value specified for **flags** is invalid.

The following function was added to the interface ***e3AttributeDefinition*** for querying the names of attribute lists:

return = e3AttributeDefinition.GetValueListName (flags)

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

Return values:

- **return "NameAttribut-Liste"**: Successfully called the function - The name of the attribute list is available.

- **return <LeererString>**: Inconclusive - No project open, the value specified for flags is invalid or the attribute is not contained in any attribute list.

References: `Designer-27031` and `E3-5037`

Customer requirement: New functions in *e3DbeJob* - Display text in output window (*GetOutbarText*, *GetOutbarTextEx*, *GetResultText* and *GetResultTextEx*)

The following function was added to the interface ***e3DbeJob*** for querying the text in the output window ***Messages*** :

return = e3DbeJob.GetOutbarText (index, lst)

Parameter:

- **index** `[in]`: Defines the first line number from which the text in the output window is queried.
- **lst** `[out]`: Contains the text of the output window.

Return values:

- **return > 0**: Successfully called the function - The text in the output window can be output line by line with **lst**.
- **return 0**: Inconclusive - There is no text in the output window ***Messages*** or an error occurred.

The following function was added to the interface ***e3DbeJob*** for querying the text in the output window ***Messages***:

return = e3DbeJob.GetOutbarTextEx (index, lst)

Parameter:

- **index** `[in][out]`: Defines the first line number from which the text in the output window is queried.
- **lst** `[out]`: Contains the text of the output window.

Return values:

- **return > 0**: Successfully called the function - The text in the output window can be output line by line with **lst**.
- **return 0**: Inconclusive - There is no text in the output window ***Messages*** or an error occurred.

The following function was added to the interface ***e3DbeJob*** for querying the text in the output window ***Results***:

return = e3DbeJob.GetResultText (index, lst)

Parameter:

- **index** `[in]`: Defines the first line number from which the text in the output window is queried.
- **lst** `[out]`: Contains the text of the output window.

Return values:

- **return > 0**: Successfully called the function - The text in the output window can be output line by line with **lst**.
- **return 0**: Inconclusive - There is no text in the output window ***Results*** or an error occurred.

The following function was added to the interface ***e3DbeJob*** for querying the text in the output window ***Results***:

return = e3DbeJob.GetResultTextEx (index, lst)

Parameter:

- **index** `[in][out]`: Defines the first line number from which the text in the output window is queried.
- **lst** `[out]`: Contains the text of the output window.

Return values:

- **return > 0**: Successfully called the function - The text in the output window can be output line by line with **lst**.
- **return 0**: Inconclusive - There is no text in the output window ***Results*** or an error occurred.

References: `Designer-34154` and `E3-5128`

Customer requirement: New function in *e3DbeModel*, *e3DbeSymbol* and *e3Sheet* - Query measurement unit when importing DXF/DWG files (*ImportDXF*)

The following function was added to the interfaces ***e3DbeModel***, ***e3DbeSymbol*** and ***e3Sheet*** for determining with which measurement unit DXF/DWG files are to be interpreted or which measurement unit was used:

return = e3DbeModel.GetDXFSizeEx (filename, font, dx, dy, scale, flags)

return = e3DbeSymbol.GetDXFSizeEx (filename, font, dx, dy, scale, flags)

return = e3Sheet.GetDXFSizeEx (filename, font, dx, dy, scale, flags)

Parameter:

- **filename** `[in]`: Defines the name of the DXF file.
- **font** `[in]`: Defines the name of the font that is used.
- **dx** `[in][out]`: Defines the width of the DXF file.
- **dy** `[in][out]`: Defines the height of the DXF file.
- **scale** `[in][out]`: Defines the scale of the DXF file.
- **flags** `[in][out][optional]`: An optional parameter with which the function can be enhanced. The parameter defines which measurement unit the DXF files are to be interpreted in or which measurement unit was determined.
  The parameter supports the following values:

- **0x001** (Decimal value **1**)`[in]`: Interprets the values in millimeters.
- **0x002** (Decimal value **2**)`[in]`: Interprets the values in inches.
- **0x004** (Decimal value **4**)`[out]`: The dimensions of the DXF file are interpreted in millimeters because **flags** either has the value 0x001 or because the measurement unit was determined automatically.
- **0x008** (Decimal value **8**)`[out]`: The dimensions of the DXF file are interpreted in inches because **flags** either has the value 0x002 or because the measurement unit was determined automatically.

Note: When the input parameter for **flags** is 0x001, the output parameter for **flags** is 0x004. Accordingly, the output parameter 0x008 is used with the input parameter 0x002.

Return values:

- **return 1**: Successfully called the function - Size and scale of the DXF file are output.
- **return 0**: Failed calling the function - Size and scale of the DXF file could not be determined.
- **return -1**: Failed calling the function - no project open or object set.
- **return -2**: Failed calling the function - The value specified for **filename** is invalid.
- **return -3**: Failed calling the function - The value specified for **font** is invalid.
- **return -4**: Failed calling the function - An internal error occurred.
- **return -5**: Failed calling the function - The value specified for **flags** is invalid.

References: `Designer-41248` and `E3-2964`

Customer requirement: New Interface *e3SheetReference*for working with sheet references

The interface ***e3SheetReference*** was added to be able to work with sheet references using the API.

The following list summarizes the functions that are available for the Interface initial.

The following function was added to the interface ***e3Job*** for creating an **e3SheetReference** object:

return = e3Job.CreateSheetReferenceObject ( )

The following function was added to the interface ***e3SheetReference*** for querying the IDs of sheet reference and signal reference symbols:

return = e3SheetReference.GetId ( )

Return values:

- **return > 0**: Successfully called the function - Outputs the ID of sheet reference or signal reference symbols.
- **return 0**: Failed calling the function - No sheet reference or signal reference symbol selected.

The following function was added to the interface ***e3SheetReference*** for querying the IDs of target symbols of sheet references:

return = e3SheetReference.GetDestinationSymbolIds (ids, flags)

Parameter:

- **ids** `[out]`: Outputs the IDs of the target symbols of sheet references.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.

Return values:

- **return ≥ 0**: Successfully called the function - Outputs the amount of target symbols of sheet references that were found.
- **return -1**: Failed calling the function - No project open.
- **return -2**: Failed calling the function - No sheet references set.
- **return -3**: Failed calling the function - The value specified for **flags** is invalid.
- **return -4**: Failed calling the function - An internal error occurred.

The following function was added to the interface ***e3SheetReference*** for querying the IDs of source symbols of sheet references:

return = e3SheetReference.GetSourceSymbolIds (ids, flags)

Parameter:

- **ids** `[out]`: Outputs the IDs of the source symbols of sheet references.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.

Return values:

- **return ≥ 0**: Successfully called the function - Outputs the amount of source symbols of sheet references that were found.
- **return -1**: Failed calling the function - No project open.
- **return -2**: Failed calling the function - No sheet references set.
- **return -3**: Failed calling the function - The value specified for **flags** is invalid.
- **return -4**: Failed calling the function - An internal error occurred.

The following function was added to the interface ***e3SheetReference*** for setting the ID of a sheet reference:

return = e3SheetReference.SetId (id)

Parameter:

- **ids** `[in]`: Defines the ID of a sheet reference.
  The sheet reference can also be determined using a given signal reference.

Return values:

- **return > 0**: Successfully called the function - Defines the ID of the sheet reference.
- **return 0**: Failed calling the function - An internal error occurred.

References: `Designer-04780` and `E3-7169`

New function in the API *MultiuserAdministrationTool - ArchiveProject*

The following function was added to the API ***MultiuserAdministrationTool*** for archiving projects:

return = e3MultiuserClient.ArchiveProject (file)

Parameter:

- **file** `[in]`: Defines the name of the multi-user project.

Return values:

- **return 1**: Successfully called the function - The project was archived.
- **return 0**: Failed calling the function - A general error occurred.
- **return -1**: Failed calling the function - The project is already archived.
- **return -8**: Failed calling the function - The project data could not be retrieved.

New function in *e3Clipboard - SetName*

The following function was added to the interface ***e3Clipboard*** for renaming clipboards that were created with ***Paste Extended*** :

return = e3Clipboard.SetName (name)

Parameter:

- **name** `[in]`: Defines the new name of the clipboard.
  The name must contain at least one character and cannot be longer than 252 characters.

Return values:

- **return 1**: Successfully called the function - The clipboard was renamed.
- **return -1**: Failed calling the function - No clipboard selected.
- **return -7**: Failed calling the function - No project open.
- **return -8**: Failed calling the function - The length of **name** is invalid.

New function in *e3Group - UpdateDrawingForProjectGenerationEx*

The following function was added to the interface ***e3Group*** for changing information of objects in groups, e.g. subcircuits:

return = e3Group.UpdateDrawingForProjectGenerationEx (flags, substitutes, allowedTexttypes, allowedAttributenames, resultArray)

Note: Please see the API documentation for a detailed description of the parameters and return values.

Parameter:

- **flags** `[in]`: Defines the type of information of objects, that is to be changes.
- **substitutes** `[in]`: Defines the source and target strings that are used for changing the information.
- **allowedTexttypes** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines text types to be changed by the function.
- **allowedAttributenames** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines attributes to be changed by the function.
- **resultArray** `[in][out][optional]`: An optional parameter with which the function can be enhanced.
  The parameter outputs an array that contains errors of the change of grouped objects.

Return values:

- **return 6**: Successfully called the function - The changes were made but multiple objects of the group have the same identifier.
- **return 5**: Successfully called the function - A part of the changes was made but the XML structure is missing or differs from the project, so not all changes could be made. **resultArray** lists the objects where the changes failed.
- **return 4**: Successfully called the function - A part of the changes was made but the XML structure is missing or differs from the project.
- **return 2**: Successfully called the function - A part of the changes was made but not all changes could be made. **resultArray** lists the objects where the changes failed.
- **return 1**: Successfully called the function - The changes were made.
- **return -1**: Failed calling the function - No project open, no group found or incorrect version of a subcircuit.
- **return -2**: Failed calling the function - Invalid text types are defined for**allowedTexttypes**.**resultArray** lists all invalid text types.
- **return -3**: Failed calling the function - Invalid attributes are defined for**allowedAttributenames**.**resultArray** lists all invalid attributes .
- **return -4**: Failed calling the function - The array **substitutes** is invalid.
- **return -5**: Failed calling the function - The array **substitutes** contains invalid strings.
- **return -6**: Failed calling the function - An internal error occurred.

- **return -7**: Failed calling the function - The group cannot be updated.
- **return -8**: Failed calling the function - The group is defined as a subcircuit in the database.

New function in *e3Slot* - *GetDefinedRotation*

The following function was added to the interface ***e3Slot*** for querying which value a slot has for the property ***Rotation of mounted device*** :

return = e3Slot.GetDefinedRotation (angle, flags)

Parameter:

- **angle** `[out]`: Defines the value that the slot has for the property ***Rotation of mounted device***.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.

Return values:

- **return 1**: Successfully called the function - **angle** outputs the value that the slot has for the property ***Rotation of mounted device*** .
- **return -1**: Failed calling the function - No project open or slot selected.
- **return -2**: Failed calling the function - An internal error occurred.
- **return -3**: Failed calling the function - The selected slot is of the type ***area*** and has no rotation value for the devices that were placed on it.
- **return -4**: Failed calling the function - An internal error occurred.
- **return -5**: Failed calling the function - The value specified for **flags** is invalid.

New functions in *e3DbeSlot* - *GetDirectionEx*, *GetLength*, *GetWidth*, *SetLength* and *SetWidth*

The following function was added to the interface ***e3DbeSlot*** for querying the position and orientation of slots:

return = e3DbeSlot.GetDirectionEx (x, y, z, side, flags)

Parameter:

- **x** `[out]`: Outputs the position of the slot on the X coordinate.
- **y** `[out]`: Outputs the position of the slot on the Y coordinate.
- **z** `[out]`: Outputs the position of the slot on the Z coordinate.
- **side** `[out]`: Outputs the orientation of the slot.
  The parameter supports the following values:

- **0**: The slot had the orientation ***Front***.
- **1**: The slot had the orientation ***Back***.
- **2**: The slot had the orientation ***Right***.
- **3**: The slot had the orientation ***Left***.
- **4**: The slot had the orientation ***Top***.
- **5**: The slot had the orientation ***Bottom***.

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

Return values:

- **return 1**: Successfully called the function - **x**, **y** and **z** output the position of the slot and **side** outputs the orientation of the slot.
- **return -1**: Failed calling the function - No project open or slot selected.
- **return -2**: Failed calling the function - The value specified for **flags** is invalid.

The following function was added to the interface ***e3DbeSlot*** for querying the length of slots of the type ***Area*** and ***Line***:

return = e3DbeSlot.GetLength (length, flags)

Parameter:

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

Return values:

- **return 1**: Successfully called the function - **length** outputs the length of the selected slot.
- **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 - An internal error occurred.
- **return -4**: Failed calling the function - An internal error occurred.
- **return -5**: Failed calling the function - The selected slot is not of the type ***Area*** or ***Line***.
- **return -6**: Failed calling the function - An internal error occurred.

The following function was added to the interface ***e3DbeSlot*** for querying the width of slots of the type ***Area***:

return = e3DbeSlot.GetWidth (width, flags)

Parameter:

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

Return values:

- **return 1**: Successfully called the function - **width** outputs the width of the selected slot.
- **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 - An internal error occurred.
- **return -4**: Failed calling the function - An internal error occurred.
- **return -5**: Failed calling the function - The selected slot is not of the type ***Area*** or ***Line***.
- **return -6**: Failed calling the function - An internal error occurred.

The following function was added to the interface ***e3DbeSlot*** for defining the length of slots of the type ***Area***:

return = e3DbeSlot.SetLength (length, flags)

Parameter:

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

Return values:

- **return 1**: Successfully called the function - The length **length** was set for the slot.
- **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 - An internal error occurred.
- **return -4**: Failed calling the function - An internal error occurred.
- **return -5**: Failed calling the function - The selected slot is not of the type ***Area*** or ***Line***.
- **return -6**: Failed calling the function - An internal error occurred.
- **return -7**: Failed calling the function - An internal error occurred.
- **return -8**: Failed calling the function - **length** is less than or equal to 0.

The following function was added to the interface ***e3DbeSlot*** for defining the width of slots of the type ***Area***:

return = e3DbeSlot.SetWidth (width, flags)

Parameter:

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

Return values:

- **return 1**: Successfully called the function - The width **width** was set for the slot.
- **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 - An internal error occurred.
- **return -4**: Failed calling the function - An internal error occurred.
- **return -5**: Failed calling the function - The selected slot is not of the type ***Area*** or ***Line***.
- **return -6**: Failed calling the function - An internal error occurred.
- **return -7**: Failed calling the function - An internal error occurred.
- **return -8**: Failed calling the function - **width** is less than or equal to 0.

New functions in *e3Device* - *GetBomType*, *GetRouteCableInSamePath*, *JumpToPanel3D* and *SetRouteCableInSamePath*

The following function was added to the interface ***e3Device*** for querying the object type of entries in the Bill of Materials, that was output by the function **e3Job.GetBomPartList** with the parameter **result**:

return = e3Device.GetBomType (types, flags)

- **types** `[out]`: Outputs a two-dimensional array on which the object type and further information about the object are output.
  For more information on the content of **types** see the documentation of the E3.series API.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.

Return values:

- **return -1**: Failed calling the function - No project open.
- **return -2**: Failed calling the function - No ID of a device selected.

The following function was added to the interface ***e3Device*** for querying whether all conductors of a cable are to be installed in the same cable duct:

return = e3Device.GetRouteCableInSamePath (embeddedSheetId, flags)

Return values:

- **return 1**: Successfully called the function - All conductors of a cable are installed in the same cable duct.
- **return 0**: Successfully called the function - Conductors of a cable are not necessarily installed in the same cable duct.
- **return -1**: Failed calling the function - No project open or no cable/device selected.
- **return -2**: Failed calling the function - The selected device is not a cable.
- **return -3**: Failed calling the function - The cable has a type, that is not supported, like cable bundle.

The following function was added to the interface ***e3Device*** for jumping to placed models in the ***Panel - Not Placed*** tree and on the sheet:

return = e3Device.JumpToPanel3D (embeddedSheetId, flags)

Parameter:

- **embeddedSheetId** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter specifies the ID of a sheet area, which is jumped to first if the design is placed across multiple sheets.
  When either the ID belongs to a panel sheet, but the device is not placed on the sheet **or** when **embeddedSheetId** is not used/has the value 0, the panel sheet on which the model is placed is jumped to.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines the camera orientation with which the placed model is displayed and supports the following values:
  - **0x001** (Decimal value **1**): The camera is aligned based on the first slot of the type ***Area*** on which the model is placed.
    The camera angle refers to the 3D view from the front.
    In the tree view of the placed models this corresponds to the slot that is situated directly above the model.
  - **0x002** (Decimal value **2**): The camera is aligned based on the last slot of the type ***Area*** on which the model is placed.
    The camera angle refers to the 3D view from the front.
    In the tree view of the placed model this corresponds to the top most slot in the branch.
  - **0x004** (Decimal value **4**): The camera is aligned with one of the six 3D views (***Front***, ***Back***, ***Right***, ***Left***, ***Top***, ***Bottom***) whose angle relative to the selected model is the smallest, as measured from the origin of the 3D world.
  - When **flags** is not used or has the value 0, the model is display in front view.

Return values:

- **return 1**: Successfully called the function - The changes were made.
- **return -1**: Failed calling the function - No project open or no device selected.
- **return -2**: Failed calling the function - The device is not placed in the panel.

- **return -3**: Failed calling the function - **embeddedSheetId** is not the ID of a sheet area.
- **return -4**: Failed calling the function - The value specified for **flags** is invalid.

The following function was added to the interface ***e3Device*** for defining whether all conductors of a cable are to be installed in the same cable duct:

return = e3Device.SetRouteCableInSamePath (samepath, flags)

Parameter:

- **samepath** `[in]`: The parameter defines whether all conductors of a cable are to be installed in the same cable duct.
  When the parameter has the value 0, conductors are not necessarily installed in the same cable duct.
  When the parameter has the value 1, all conductors of a cable are installed in the same cable duct.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.

Return values:

- **return 1**: Successfully called the function - The setting to define whether all conductors of a cable are to be installed in the same cable duct is set.
- **return -1**: Failed calling the function - No project open or no cable/device selected.
- **return -2**: Failed calling the function - The selected device is not a cable.
- **return -3**: Failed calling the function - The cable has type, that is not supported, like cable bundle.
- **return -4**: Failed calling the function - The cable is locked.
- **return -5**: Failed calling the function - The value specified for **flags** is invalid.

New functions in *e3Job* - *Get3DAnimationEnabled* and *Set3DAnimationEnabled*

The following function was added to the interface ***e3Job*** for querying whether changes of 3D views are to be animated in the panel:

return = e3Job.Get3DAnimationEnabled ( )

Return values:

- **return 1**: Successfully called the function - 3D animation is activated.
- **return 0**: Successfully called the function - 3D animation is deactivated.
- **return -1**: Failed calling the function - No project open.

The following function was added to the interface ***e3Job*** for defining whether changed of 3D views in the panel are to be animated:

return = e3Job.Set3DAnimationEnabled (newval)

Parameter:

- **newval** `[in]`: The parameter defines whether changing 3D views in the panel are animated.
  The parameter supports the following values:
  - **0**: 3D animation is activated.
  - **1**: 3D animation is deactivated.

Return values:

- **return 1**: Successfully called the function - The setting for 3D animations was changed.
- **return -1**: Failed calling the function - No project open.

New functions in *e3Outline* - *CreateDynamicContour*, *CreateDynamicDrillHole* and *Delete*

The following function was added to the interface ***e3Outline*** for creating contours for models dynamically in the project:

return = e3Outline.CreateDynamicContour (deviceId, contourType, points, xCoordinates, yCoordinates, zPosition, height, flags)

Parameter:

- **deviceId** `[in]`: Defines the ID of the device to which a contour is added.
- **contourType** `[in]`: Defines the type of contour that is created.
  The parameter supports the following values:
  - **2**: Creates a contour of the type ***Restricted for all***.
  - **3**: Creates a contour of the type ***Restricted for components***.
  - **4**: Creates a contour of the type ***Restricted for connections***.
  - **5**: Creates a contour of the type ***Cutout area***.
  - **6**: Creates a contour of the type ***Drill-hole***.
- **points** `[in]`: Defines the number of points used for creating the contour.
  When a contour of the type ***Drill-hole*** is to be created, at least 15 points are to be used.
- **xCoordinates** `[in]`: Defines the X coordinates of all points of the polygon with which the contour is defined.
- **yCoordinates** `[in]`: Defines the Y coordinates of all points of the polygon with which the contour is defined.
  Note: The last value of **xCoordinates** and **yCoordinates** must be identical. When a contour of the type ***Drill-hole*** is to be created, all values must be the same distance from the center point of the polygon.
- **zPosition** `[in]`: Defines the Z position of the contour.
- **height** `[in]`: Defines the height of the contour.
  The value must be at least 0.1.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether contours of the type ***Cutout area*** or ***Drill-hole*** are used in

E3.CutOut or whether a threaded hole is caused for contours of the type ***Drill-hole*** when manufacturing on the carrier object of the model.
The parameter supports the following values:

- **0x001** (Decimal value **1**): Contours of the type ***Cutout area*** or ***Drill-hole*** are not used in E3.CutOut.
- **0x002** (Decimal value **2**): Contours of the type ***Drill-hole*** cause a threaded hole when manufacturing on the carrier object of the model

Return values:

- **return > 0**: Successfully called the function - Outputs the ID of the created contour.
- **return -1**: Failed calling the function - Contours cannot be created with an active E3.redliner or E3.view license.
- **return -2**: Failed calling the function - No project open.
- **return -3**: Failed calling the function - No device selected.
- **return -4**: Failed calling the function - Device is locked.
- **return -5**: Failed calling the function - The value specified for **flags** is invalid.
- **return -6**: Failed calling the function - The value specified for **contourType** is invalid.
- **return -7**: Failed calling the function - The selected device is not a model.
- **return -8**: Failed calling the function - The device has variant positions.
- **return -9**: Failed calling the function - The selected object is a cable duct inlet / outlet.
- **return -10**: Failed calling the function - The device is not placed on a panel sheet.
- **return -11**: Failed calling the function - The sheet on which the device is placed, is read-only or locked.
- **return -12**: Failed calling the function - The value specified for **height** is invalid.
- **return -13**: Failed calling the function - The polygon is invalid.
- **return -14**: Failed calling the function - The contour cannot be created, because it would cause a collision.
- **return -15**: Failed calling the function - An internal error occurred.

The following function was added to the interface ***e3Outline*** for creating drill-holes for models dynamically in the project:

return = e3Outline.CreateDynamicDrillHole (deviceId, xCenter, yCenter, zPosition, height, radius, flags)

Parameter:

- **deviceId** `[in]`: Defines the ID of the device to which a drill-hole is added.
- **xCoordinates** `[in]`: Defines the X coordinate of the center point of the drill-hole.
- **yCoordinates** `[in]`: Defines the Y coordinate of the center point of the drill-hole.

- **zPosition** `[in]`: Defines the Z coordinate of the drill-hole.
- **height** `[in]`: Defines the height of the contour.
  The value must be at least 0.1.
- **radius** `[in]`: Defines the radius of the contour.
  The value must be at least 0.1.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced.
  The parameter defines whether the drill-hole is used in E3.CutOut or whether a threaded hole is caused for the drill-hole when manufacturing on the carrier object of the model.
  The parameter supports the following values:
  - **0x001** (Decimal value **1**): The drill-hole is not used in E3.CutOut.
  - **0x002** (Decimal value **2**): The drill-hole causes a threaded hole when manufacturing on the carrier object of the model.

Return values:

- **return > 0**: Successfully called the function - Outputs the ID of the created contour.
- **return -1**: Failed calling the function - Contours cannot be created with an active E3.redliner or E3.view license.
- **return -2**: Failed calling the function - No project open.
- **return -3**: Failed calling the function - No device selected.
- **return -4**: Failed calling the function - Device is locked.
- **return -5**: Failed calling the function - The value specified for **flags** is invalid.
- **return -6**: Failed calling the function - The value specified for **radius** is invalid.
- **return -7**: Failed calling the function - The selected device is not a model.
- **return -8**: Failed calling the function - The device has variant positions.
- **return -9**: Failed calling the function - The selected object is a cable duct inlet / outlet.
- **return -10**: Failed calling the function - The device is not placed on a panel sheet.
- **return -11**: Failed calling the function - The sheet on which the device is placed, is read-only or locked.
- **return -12**: Failed calling the function - The value specified for **height** is invalid.
- **return -14**: Failed calling the function - The contour cannot be created, because it would cause a collision.
- **return -15**: Failed calling the function - An internal error occurred.

The following function was added to the interface ***e3Outline*** for deleting contours dynamically created in the project:

return = e3Outline.Delete (flags)

Parameter:

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

Return values:

- **return 1**: Successfully called the function - The selected contour was deleted.
- **return -1**: Failed calling the function - Contours cannot be deleted with an active E3.redliner or E3.view license.
- **return -2**: Failed calling the function - No project open.
- **return -3**: Failed calling the function - No contour selected
- **return -4**: Failed calling the function - Device is locked.
- **return -5**: Failed calling the function - The value specified for **flags** is invalid.
- **return -6**: Failed calling the function - Only contours of the type ***Restricted for all***, ***Restricted for components***, ***Restricted for connections*** and ***Cutout area*** can be deleted.
- **return -7**: Failed calling the function - The contour was not created dynamically in the project and can therefore not be deleted.
- **return -8**: Failed calling the function - The contour cannot be deleted because the are objects placed on it.
- **return -9**: Failed calling the function - An internal error occurred.

New functions in *e3Pin* - *GetLogicalEquivalencePinIds*, *Get3DTransparency* and *Set3DTransparency*

The following function was added to the ***e3Pin*** interface for querying ID pin with equivalent signals:

return = e3Pin.GetLogicalEquivalencePinIds (ids, flags)

Parameter:

- **ids** `[out]`: Outputs the IDs of signal equivalent pin.
- **flags** `[in][optional]`: An optional parameter with which the function can be enhanced in the future.
  Presently, the parameter does not support any values.

Return values:

- **return ≥ 0**: Successfully called the function - Outputs number of signal equivalent pin.
- **return -1**: Failed calling the function - No project open or no pin selected.
- **return -2**: Failed calling the function - The selected device in not a pin.
- **return -3**: Failed calling the function - An internal error occurred.

The following function was added to the ***e3Pin*** interface for checking whether installed conductors or wires are transparent in the 3D view of panels:

return = e3Pin.Get3DTransparency ( )

Return values:

- **return 3**: Successfully called the function - The installed conductors or wires are invisible.
- **return 2**: Successfully called the function - The installed conductors or wires are transparent.
- **return 1**: Successfully called the function - The installed conductors or wires are visible.
- **return 0**: Failed calling the function - No installed conductors or wires selected.

The following function was added to the ***e3Pin*** interface for defining whether installed conductors or wires are transparent in the 3D view of panels:

return = e3Pin.Set3DTransparency (mode)

Parameter:

- **mode** `[in]`: The parameter defines the visibility of conductors or wires in the 3D view of panels. The parameter supports the following values:
  - **1** and **2**: The installed conductors or wires are invisible.
  - **3** and **4**: All installed conductors or wires are visible, except for the selected objects.
  - **5** and **6**: The installed conductors or wires are transparent.
  - **7** and **8**: All installed conductors or wires are transparent, except for the selected objects.
  - **9** and **10**:The installed conductors or wires are visible.
  - **11** and **12**: All installed conductors or wires are invisible, except for the selected objects.

Return values:

- **return 0**: Inconclusive - The function was called up.

New functions in *e3Sheet* - *GetPreferSinglePinPlacement* and *SetPrefer-SinglePinPlacement*

The following function was added to the ***e3Sheet*** interface for checking whether sheets have the property, that when placing connectors via drag & drop single pin are preferably used:

return = e3Sheet.GetPreferSinglePinPlacement ( )

Return values:

- **return 1**: Successfully called the function - The property is active for the selected sheet.
- **return 0**: Successfully called the function - The property is not active for the selected sheet.
- **return -1**: Failed calling the function - No project open.
- **return -2**: Failed calling the function - No sheet selected.

The following function was added to the ***e3Sheet*** interface for defining whether or not sheets have the property, that when placing connectors via drag & drop single pin are used by default:

return = e3Sheet.SetPreferSinglePinPlacement (prefer_single_pin_placement)

Parameter:

- **prefer_single_pin_placement** `[in]`: The parameter defines whether the property of the selected sheet is set or removed, specifying that single pins are to be used by default for connectors when placing them via drag & drop.
  The parameter supports the following values:
  - **true**: When placing via drag & drop single pins are used by default.
  - **false**: When placing via drag & drop a pin group is used by default.

Return values:

- **return 1**: Successfully called the function - The property was deactivated or activated.
- **return -1**: Failed calling the function - No project open.
- **return -2**: Failed calling the function - No sheet selected.
- **return -3**: Failed calling the function - The property cannot be added with an active E3.redliner or E3.view license.
- **return -4**: Failed calling the function - Sheet is locked.
- **return -5**: Failed calling the function - The selected ID is part of an area.
- **return -6**: Failed calling the function - The sheet is read-only.

General Changes

- Enhancements to be able to use Python when working with E3.series
- Project window for displaying grouped objects
- Start-up parameters for opening the database editor without a project
- Start-up parameters for starting E3.series processes without triggers

Enhancements to be able to use Python when working with E3.series

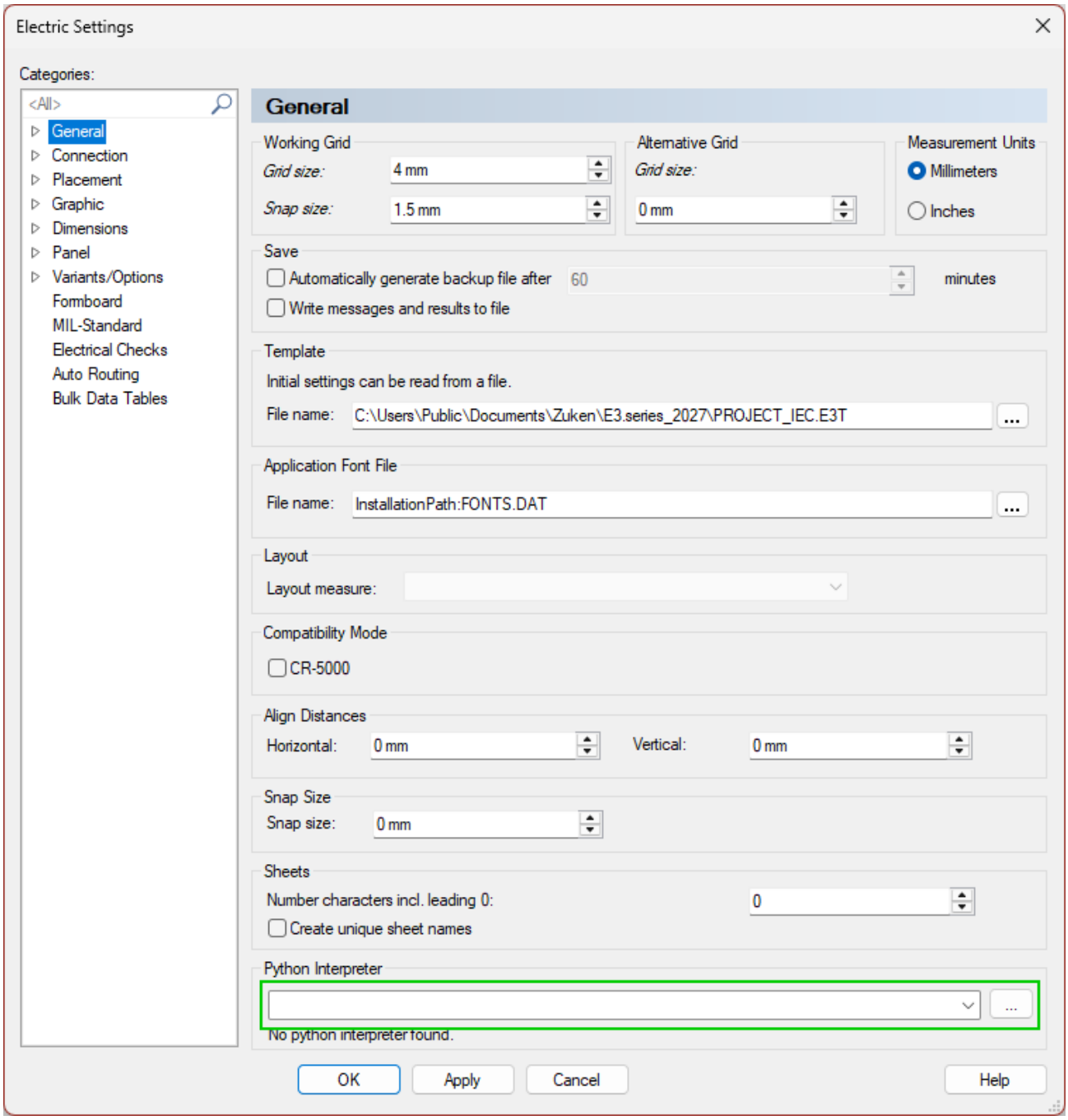
With enhancements the basis was created for using E3.series along with Python.
Changes were made to, on the one hand, be able to integrate Python scripts and on the other hand, be able to use the API more efficiently with Python.

E3.series thereby supports a widespread script language to offer an alternative for the gradual end of VBScript.

To select the interpreter, that is used to execute Python scripts, select ***Tools → Settings...*** in the main menu.
The ***Electric Settings*** dialog opens.

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

Next, enter the path of the interpreter, that is to be used when executing Python scripts, under ***Python Interpreter***:



Any environment variables that are used to define the path manually are resolved.

Click ... to search the interpreter via the ***Select File*** dialog.

The drop-down list contains all available interpreters.
Interpreters are searched for in the following folders:

- `%LOCALAPPDATA%\anaconda<x>\`
- `%LOCALAPPDATA%\miniconda<x>\`
- `%LOCALAPPDATA%\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.<xxx>`
- `%LOCALAPPDATA%\Programs\Python<xxx>\`

- %PROGRAMDATA%\anaconda<x>\
- %PROGRAMDATA%\miniconda<x>\
- %PROGRAMFILES%\Python<xxx>\

In additon, the Python interpreters python.exe and pythonw.exe, that are found in the environment variable %PATH%, are listed.

To integrate Python scripts in E3.series, select ***Tools → Customize...*** in the main menu window. The dialog ***Customize*** opens.

Next, select the ***Add-ons*** tab.

Then, define the Python script, that is to be integrated, under ***Command:***.
The script must be of the file type *.py, *.pyc or *.pyd.

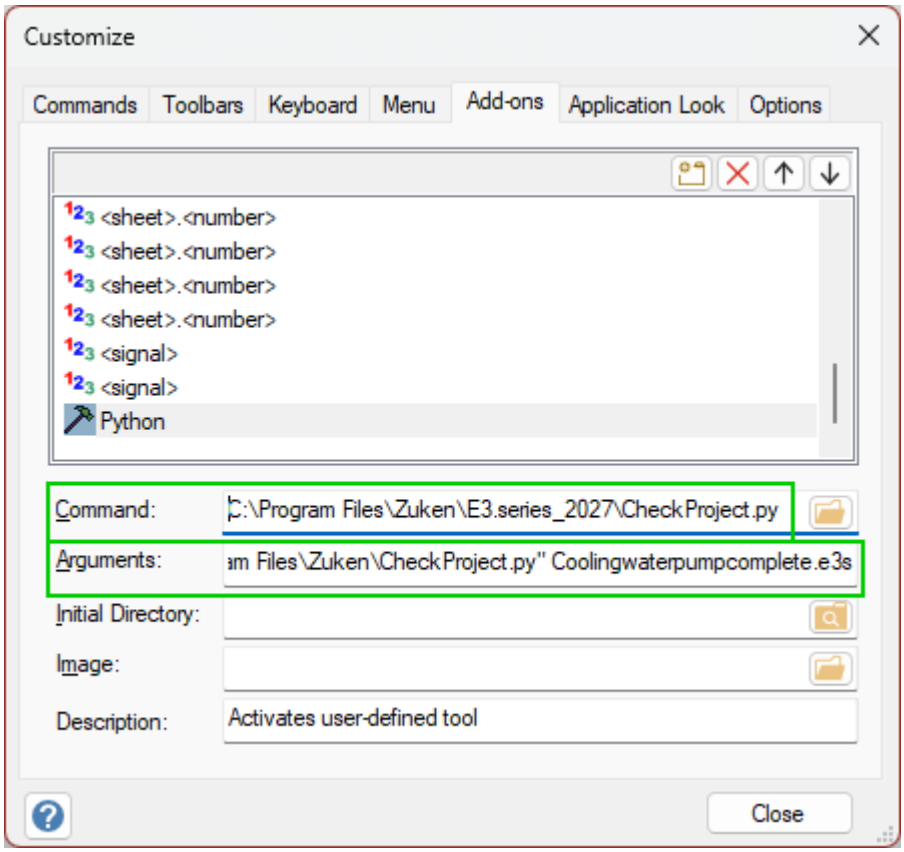
Additionally, define the parameters to be used when executing the script under ***Arguments:***.

By default, the interpreter is used as defined in the settings.

Alternatively, an interpreter, to be used when executing the script, can be selected for each script individually.

When the interpreter is to be defined individually, define the file path of the interpreter under ***Command:*** and the path to the script to be executed as the first parameter under ***Arguments:***.

Example: When a file path of an interpreter is defined under ***Command:*** and therefore a script is to be executed (e.g. CheckProject.py in the folder C:\Program Files\Zuken has as argument the name of an E3.series project, e.g. Coolingwaterpumpcomplete.e3s), enter the following under ***Arguments:***
"C:\Program Files\Zuken\CheckProject.py" Coolingwaterpumpcomplete.e3s



References: E3-1685

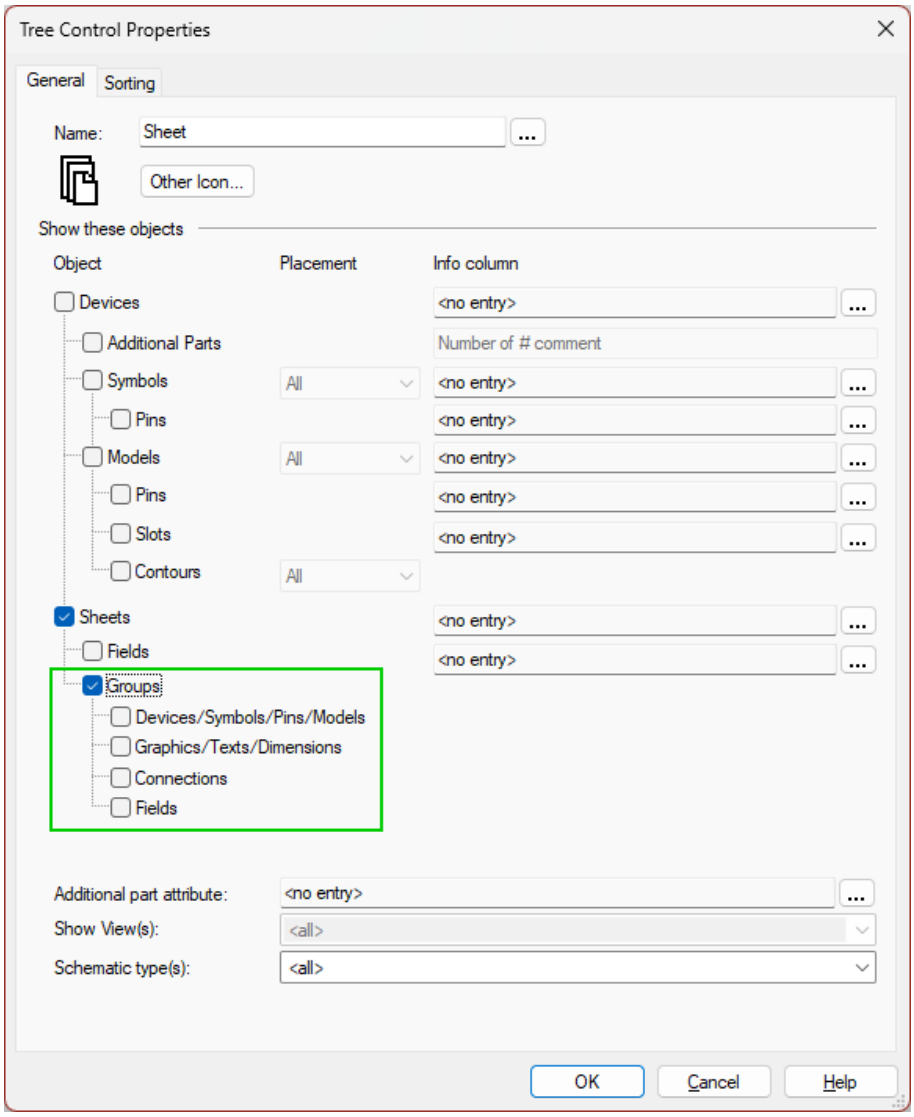
Project window for displaying grouped objects

Grouped objects can be displayed in the existing project windows ***Sheet***, ***Device***, ***Panel - Placed*** and ***Panel - Not Placed***.

This improves the overview of projects and the groups they contain, and makes it easier to edit existing groups.

To display groups in existing project windows, open the context menu of the desired project window and the ***Tree Control Properties...***
The ***Tree Control Properties*** dialog opens.

Next, activate the checkbox ***Groups*** to display all groups or select those objects out of the supported objects ***Devices/Symbols/Pins/Models***, ***Graphics/Texts/Dimensions***, ***Connections*** and ***Fields*** that are to be displayed:



Note: When displaying ***Groups*** is activated, displaying ***Sheets*** is automatically activated as well.

Placed object groups and their elements are displayed in the tree.
Sheet groups and unplaced object groups are not displayed.

For groups, the info column displays the same properties as those set in the upper part of the dialog box.
For devices in groups, the info column displays the information set under ***Devices***.
For models in groups, the info column displays the information set under ***Models***.
For grouped ***Fields***, the information displayed in the info column is that set for regular fields.
For grouped ***Graphics/Texts/Dimensions*** and ***Connections***, no information can be displayed in the info column.

Via the context menu of the group, it is possible to jump to the placed, grouped objects.
The color, line weight and the zoom factor for the highlight corresponds to the settings under ***General → Highlight***.

It is also possible to unplace objects within a group or entire groups. This command unplaces the selected object or group, including all groups containing the object, up to the uppermost group.

There is also a limited range of commands available in the context menu for each group element.

To display a distinct project window for groups, select ***View → Project window → New Project Window...*** in the main menu window.

Next, enter a name for the project window and define which groups are to be displayed and which information is to be displayed in the info column.

To further improve working with groups, objects can be removed from groups.

For grouped objects, the ***Remove from Group*** command is available on sheets and in the Project window.

In the following cases the command is unavailable:

- The sheet, on which the group is placed, is read-only or locked.
- The group is part of a sheet group.
- Sheets, that are part of a sheet group, are selected.

Start-up parameters for opening the database editor without a project

With a start-up parameter, it is possible to open the database editor without an empty project.

This makes it possible to customize the start-up behavior of the database editor as required, so that it matches the start-up behavior of E3.series.

To open the database editor without an empty project use the parameter **`/nonew`**.

Start-up parameters for starting E3.series processes without triggers

With a start-up parameter it is possible to start an E3.series in which no triggers are executed.

This makes it possible to temporarily ignore triggers that are activated without first having to disable them via the API or in the registry.

To start an E3.series process, in which no triggers are executed, use the parameter **`/notrigger`**.

Database Editor

- Changing models of components
- Checking the database for errors
- The command 'Place at original position' is available again in the Database editor
- Enhancements and adaptions for working with block connectors and connectors with inserts
- Jumping to the model tree directly from components with models
- Additional area for user-defined text types

Changing models of components

Models of components can be swapped with other components with the same number of pins.

This makes it easier to replace models and reduces the amount of processing required when, for example, a component can be used in production with models of different dimensions.

To change the model of a component modify the desired component.

Next, place the desired model on the sheet using drag & drop or the ***Place*** command in the ***Model*** tab of the database window on the work sheet.

Note that models can only be changed when the original and the model to be swapped have the same number of pins.

References: `Designer-07317` and `E3-1425`

Checking the database for errors

The defined database can be checked for formal errors.
For a successful check, the primary keys have to be correct, as well as all necessary tables, columns and indexes have to be present.

This makes it possible, to avoid working with corrupt databases, which also reduces the risk of subsequent conversion errors.

To check a connected database for error, open the context menu on the uppermost element of the database window and select ***Check Database***.
The database is checked and a summary of the check is output in the ***Messages*** window.

References: `E3-5012`

The command '*Place at original position*' is available again in the Database editor

The command ***Place at original position*** in the database editor lead to unexpected results for some users and was consequently temporarily removed.

It is now again possible to copy or cut objects on a sheet in the database editor and to paste them on another sheet in their original position.

Please note the following remarks:

- When placing, the sheet format of the sheet on which the object is to be placed, must be at least as big as the original sheet. There cannot be any objects already in the placement position.
- The command inserts the copied or cut content on the same sheet coordinates on which the content is or was originally placed.
- When placing, for example subcircuits the value for ***Grid size:*** defined under ***Settings... → General*** in the current project, is used.

References: E3-4127

Enhancements and adaptions for working with block connectors and connectors with inserts

For existing and newly created connectors with inserts, it is possible to create one block connector symbol each in the database editor.

When connectors with inserts are created or modified a block, with which the block connector symbol can be edited, is placed on the sheet automatically.

Note: The definition of multiple block connector symbols of MIL connectors is not made possible through the enhancement.

For connectors with inserts the rule of determining symbols of mating connectors as well as the behavior of the commands ***Place Complete Device*** and ***Update in Project*** were adjusted.

When all pins of connectors with inserts and only those pins are places, the following behavior applies:

- When placing the ***Place Complete Device*** command, the symbol of the active connector pin terminal of the connector with inserts is placed.
  In earlier versions the symbol of the first insert of the connector was placed.
- When the option ***Keep connector symbols*** under ***Settings... → General → Update in Project*** is inactive and the command ***Update in Project*** is used, the symbol of the connector with inserts is replaced by the symbol of the active insert of the connector with inserts.
  In earlier versions the symbol of the first insert of the connector was placed.
- For connections the active insert of the active mating connector is used as the symbol of the mating connector.

If the connector with inserts contains other connectors with inserts, the rules above apply based on the connector with inserts that is lowest in the tree structure.

It is possible to prevent switching of inserts of connectors with inserts on block components in the project.

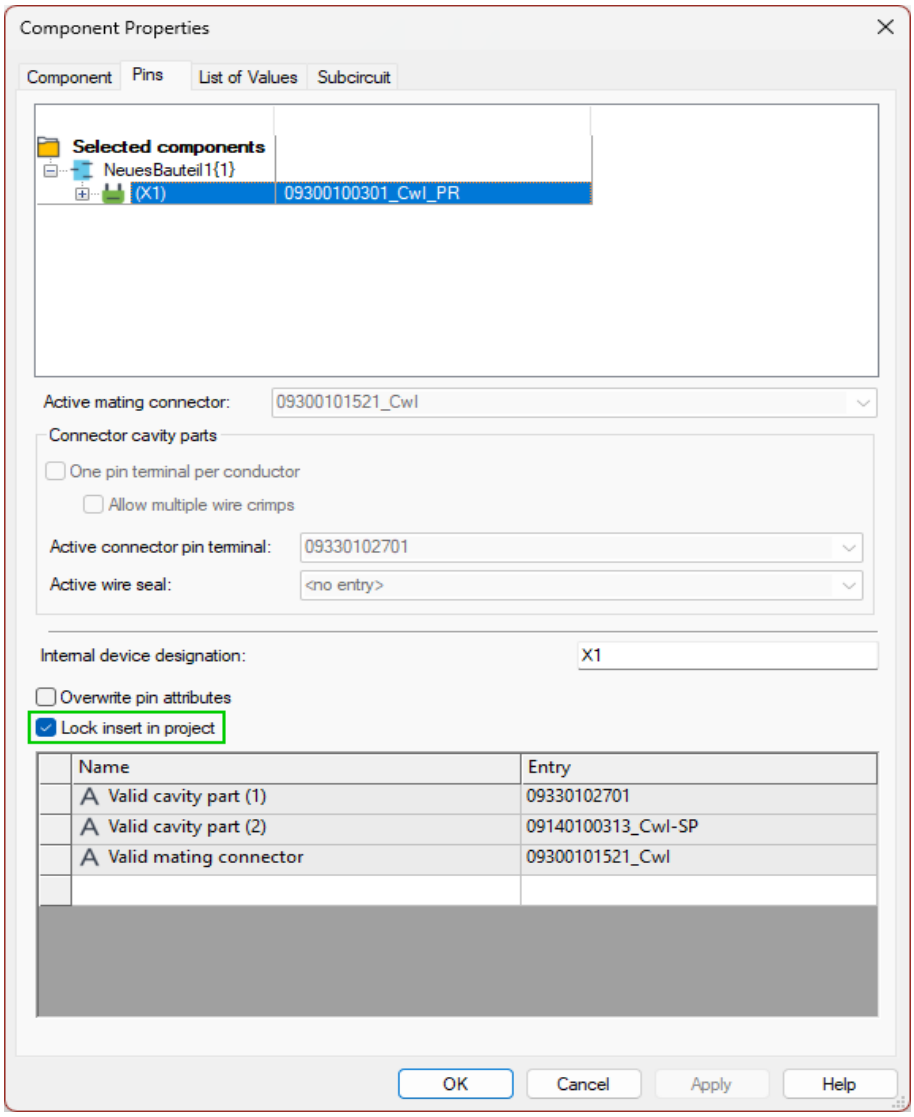
This makes it possible to protect block components, that are based on inserts of connectors with inserts, from unwanted changes.

To prevent switching of inserts of connectors with inserts on block components, create or modify a block component that uses a connector with inserts.

Next, open the context menu of the block in the project window ***Database Editor*** or on the open sheet and select ***Properties...*** or ***Component Properties***.
The ***Component Properties*** dialog opens.

Then, select the ***Pins*** tab.

Select one or more connectors with inserts of the block and activate the ***Lock insert in project*** setting.
When using the block the insert of the selected connectors with inserts can no longer be changed:



Jumping to the model tree directly from components with models

When editing components with models, it is possible to jump directly to the model tree.

This makes it easier to get form components to assigned models, making editing models in the database editor more efficient.

To jump from components with models directly to the model tree, edit a component with a model.

Next, open the context menu of the block in the project window ***Database Editor*** or on the open sheet and select ***Jump → Jump to Model Tree***.
In the model tree the selected model is jumped to and can, for example, be opened for editing.

References: `Designer-37586` and `E3-4127`

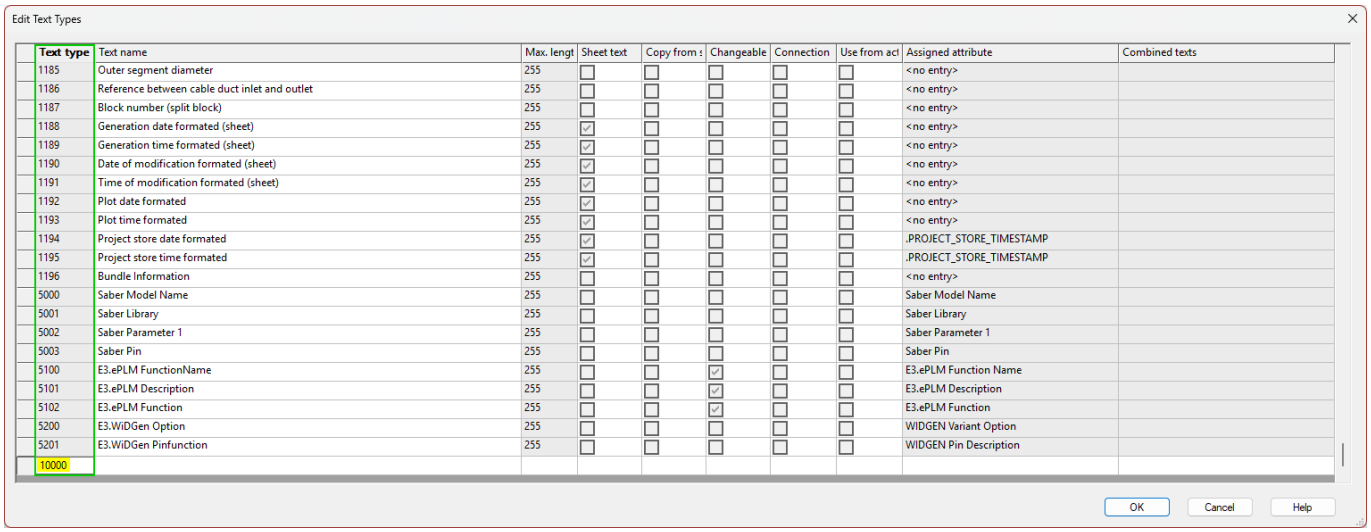
Additional area for user-defined text types

There is an additional area 10.000 to 19.999 for creating user-specific text types.

This makes it possible to create and use more text types for specific use cases.

to create user-defined text types in the new area, select ***Format → Text Types...*** in the main menu window.
The ***Edit Text Types*** dialog opens.

Go to the end of the list and enter the number of the text type within the area 10.000 and 19.999 in the ***Text type*** column and enter the desired properties of the text type.



Text type	Text name	Max. lengt	Sheet text	Copy from s	Changeable	Connection	Use from acl	Assigned attribute	Combined texts
1185	Outer segment diameter	255	☐	☐	☐	☐	☐	<no entry>	
1186	Reference between cable duct inlet and outlet	255	☐	☐	☐	☐	☐	<no entry>	
1187	Block number (split block)	255	☐	☐	☐	☐	☐	<no entry>	
1188	Generation date formated (sheet)	255	☑	☐	☐	☐	☐	<no entry>	
1189	Generation time formated (sheet)	255	☑	☐	☐	☐	☐	<no entry>	
1190	Date of modification formated (sheet)	255	☑	☐	☐	☐	☐	<no entry>	
1191	Time of modification formated (sheet)	255	☑	☐	☐	☐	☐	<no entry>	
1192	Plot date formated	255	☑	☐	☐	☐	☐	<no entry>	
1193	Plot time formated	255	☑	☐	☐	☐	☐	<no entry>	
1194	Project store date formated	255	☑	☐	☐	☐	☐	.PROJECT_STORE_TIMESTAMP	
1195	Project store time formated	255	☑	☐	☐	☐	☐	.PROJECT_STORE_TIMESTAMP	
1196	Bundle Information	255	☐	☐	☐	☐	☐	<no entry>	
5000	Saber Model Name	255	☐	☐	☐	☐	☐	Saber Model Name	
5001	Saber Library	255	☐	☐	☐	☐	☐	Saber Library	
5002	Saber Parameter 1	255	☐	☐	☐	☐	☐	Saber Parameter 1	
5003	Saber Pin	255	☐	☐	☐	☐	☐	Saber Pin	
5100	E3.ePLM FunctionName	255	☐	☐	☑	☐	☐	E3.ePLM Function Name	
5101	E3.ePLM Description	255	☐	☐	☑	☐	☐	E3.ePLM Description	
5102	E3.ePLM Function	255	☐	☐	☑	☐	☐	E3.ePLM Function	
5200	E3.WiDGen Option	255	☐	☐	☐	☐	☐	WIDGEN Variant Option	
5201	E3.WiDGen Pinfunction	255	☐	☐	☐	☐	☐	WIDGEN Pin Description	
10000									

References: `Designer-21997` and `E3-1685`

Importing and Exporting Data

Importing Data

- **Importing and exporting configuration files with supported keyboard shortcuts**
- **Defining measurement unit when importing DXF/DWG files**

Exporting Data

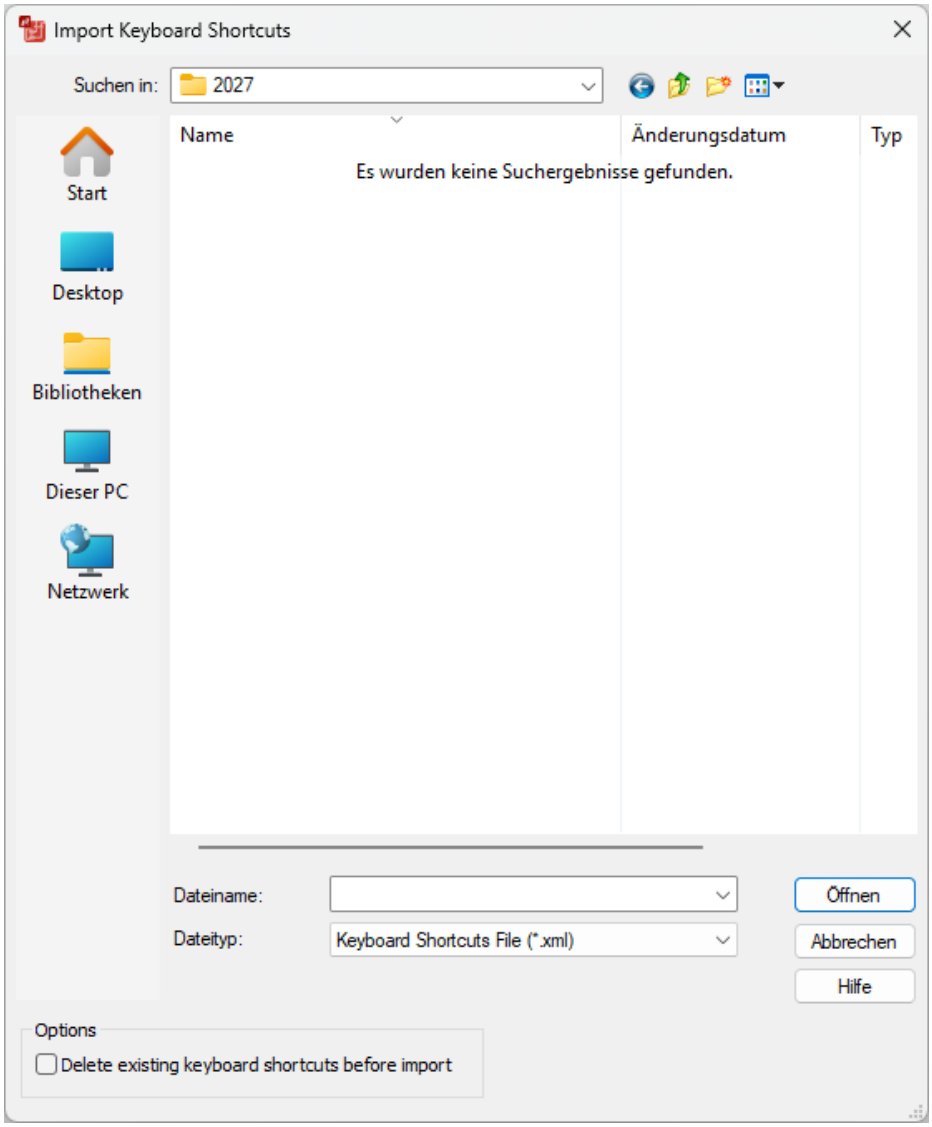
- **Importing and exporting configuration files with supported keyboard shortcuts**

Importing and exporting configuration files with supported keyboard shortcuts

Configuration files, that contain all defined keyboard shortcuts, can be imported and exported via two dialogs.

This makes it possible to, for example, exchange user-defined keyboard shortcuts for E3.series between different users or computers.

To import a configuration file with supported keyboard shortcuts, select ***Tools→Import Keyboard Shortcuts*** in the main menu window.
The ***Import Keyboard Shortcuts*** dialog opens:

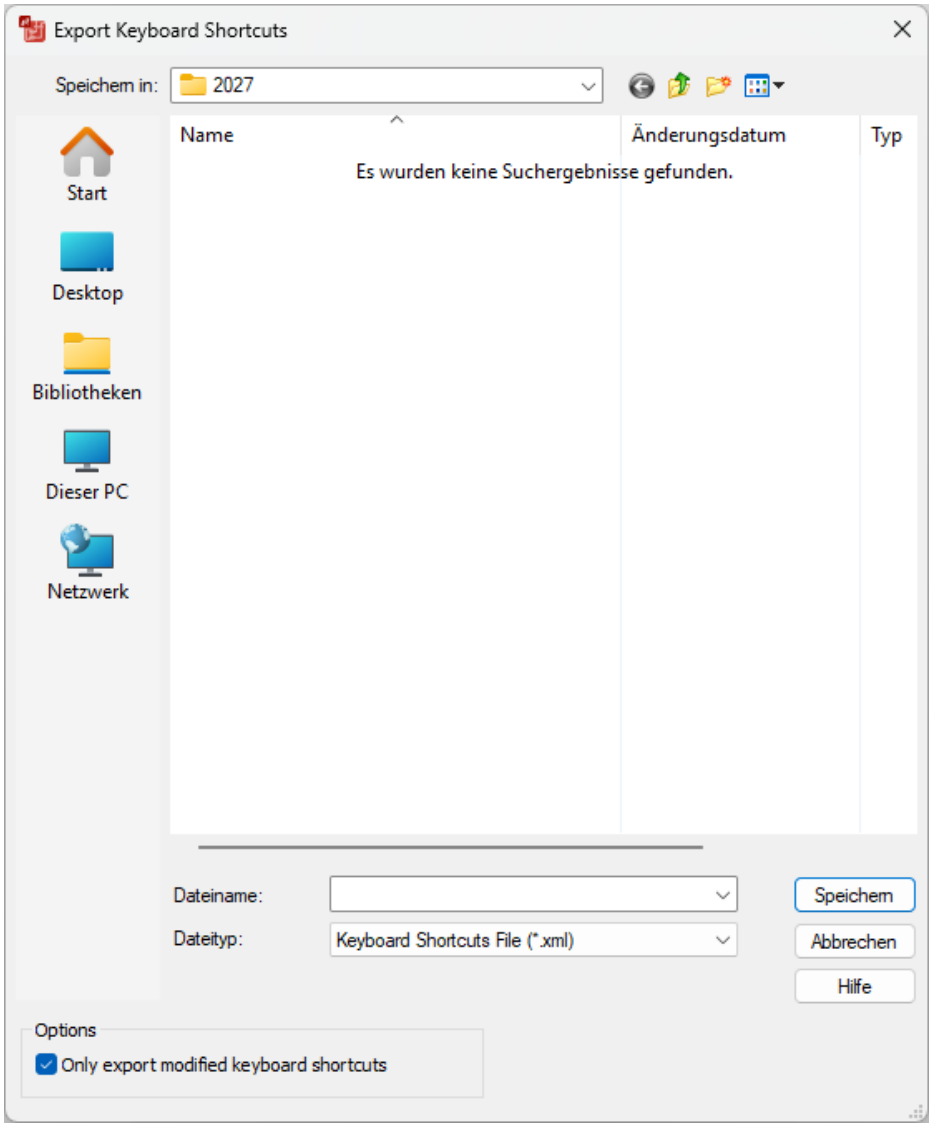


The dialog has the option to delete all existing keyboard shortcuts with ***Delete existing keyboard shortcuts before import***.

The keyboard shortcuts of the configuration file are added to the already defined keyboard shortcuts. When keyboard shortcuts from the configuration file conflict with the preexisting ones, the keyboard shortcuts form the configuration file are used.

Conflicts arise when there identical keyboard shortcuts for multiple different commands.
Multiple keyboard shortcuts for the same command are possible.

To export a configuration file with supported keyboard combinations, select ***Tools → Export Keyboard Shortcuts*** in the main menu window.
The dialog ***Export Keyboard Shortcuts*** opens:



The dialog has the option to export only keyboard shortcuts, that were changed, with ***Only export modified keyboard shortcuts***.

The folders, that are opened by default for importing and exporting, can be defined in the setting under ***General → Default Directories***.

Configuration files for keyboard shortcuts are written in XML format:

Part of a configuration file

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
                <e3Keybinds keyboardLayoutId="00000407">
                <Keybind>
                <Modifiers>
                <KeyName>Alt</KeyName>
                </Modifiers>
                <Key>
                <KeyCode>0x73</KeyCode>
                <KeyName>F4</KeyName>
                </Key>
```

```
            <Command>57665</Command>
            <!--Beenden-->
        </Keybind>
```

The schema contains the following essential elements:

- `<e3Keybinds>`: The root element.
  When exporting with E3.series the element automatically receives an attribute `keyboadLayoutId`, that contains as its value the language code and keyboard layout.
- `<Keybind>`: Every element defines a complete keyboard shortcut and its respective command.
- `<Modifiers>`: Defines up to three modification keys (***Ctrl***, ***Alt*** and ***Shift***) that are part of the keyboard shortcut.
  For each of the modification keys, that belongs to the combination, there is an element `<KeyName>`.
- `<Key>`: Defines the main key of the combination. Keys can be defined either with a language-dependent name in `<KeyName>` or with a language-independent, Windows code in `<KeyCode>`.
- `<KeyName>`: Defines the key with a language-dependent name.
  Language is the user language with which E3.series was started for the export.
- `<KeyCode>`: Defines the key with a language-independent, Windows code.
- `<Command>`: Defines the command as an E3.series identifier.
  In the XML commentary under the element, the language-dependent name of the command is output.

For keyboard shortcuts that are used multiple times, only the combination, that was last defined in the XML file, is used.

When multiple elements `<KeyName>` or `<KeyCode>` exist in `<Key>`, the key defined in the last `<KeyCode>` element is used.
When no element `<KeyCode>` is defined, the key, that is used in the last `<KeyName>` element, is used.

Note: Because `<KeyName>` is language-dependent, it is possible to be used for manual modification of the language-independent code in the `<KeyCode>` element to avoid conflicts when exchanging the configuration file between different users.
When the language-independent code `<KeyCode>` is not known, `<KeyName>`can also be used for the definition. In that case the same keyboard layout in Windows has to set when importing as the one set when exporting the configuration file.

References: `E3-8133`

Defining measurement unit when importing DXF/DWG files

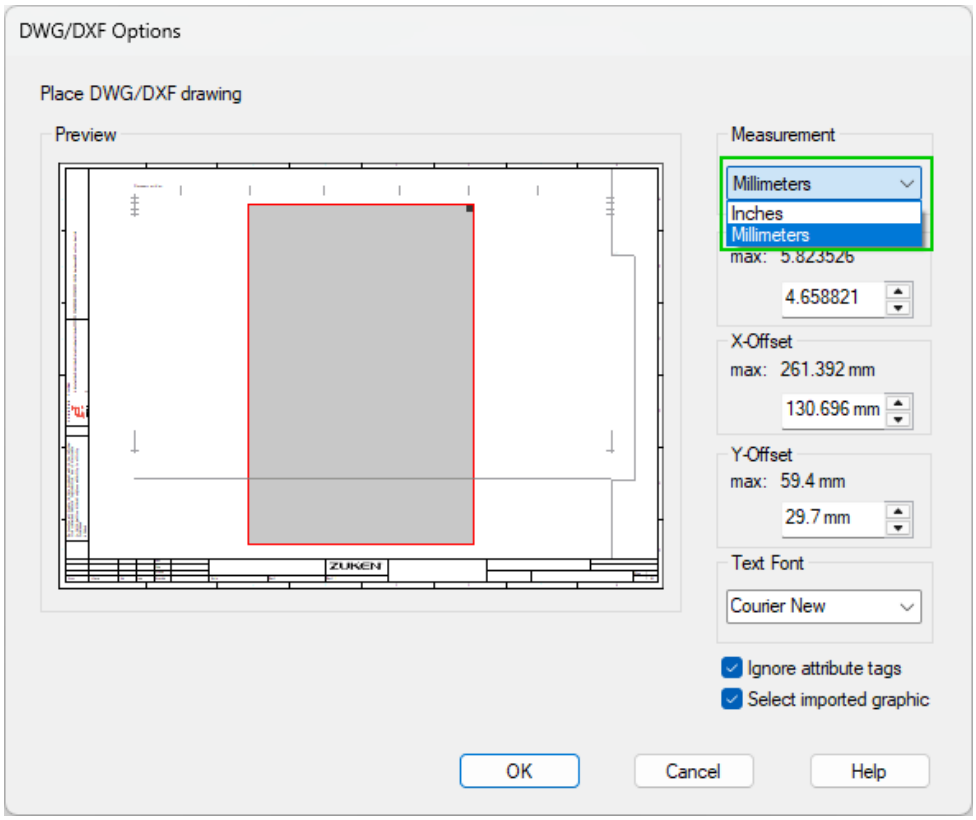
When importing DXF and DWG files, it is possible to define in which measurement unit values are interpreted.

This makes it possible to define the measurement unit, even if files, for example, don't use the variable `$MEASUREMENT` for defining measurement units.

To define the measurement unit for importing DXF/DWG files, select ***File → Import → DXF/DWG...*** in the main menu window.
The ***Import DWG/DXF File*** dialog opens.

Next, select the file that is to be imported.
The ***DWG/DXF Options*** dialog opens.

Define the measurement unit under ***Measurement***:



By default, the measurement unit, that is defined for the variable `$MEASUREMENT` of the file or the configuration file `DXFImport.cfg`, is preselected.

References: `Designer-41248` and `E3-2964`

Formboard

- Command in the context menu for jumping to formboard sheets
- Moving nodes automatically along with any attribute text template symbols placed on them

Command in the context menu for jumping to formboard sheets

For devices, that are placed on formboard sheets, there is a separate context menu command to jump directly to the formboard sheet.

This makes it possible to actively select whether the schematic or the formboard sheet is to be jumped to.

The command ***Jump to Formboard*** is available in the context menu of the following objects:

- Devices or models placed on sheets
- Device views placed on sheets
- Pins of devices and models on sheets and the tree
- Pins in the pin table
- Pin views on sheets and in the tree
- Devices in the device tree
- Devices in the device table

If there are multiple formboard views available, that could be jumped to, one view is jumped to and all other formboard views are output in the ***Results*** window.

References: `Designer-17199` and `E3-2406`

Moving nodes automatically along with any attribute text template symbols placed on them

Attribute text template symbols, that are placed on nodes, can be automatically moved together.

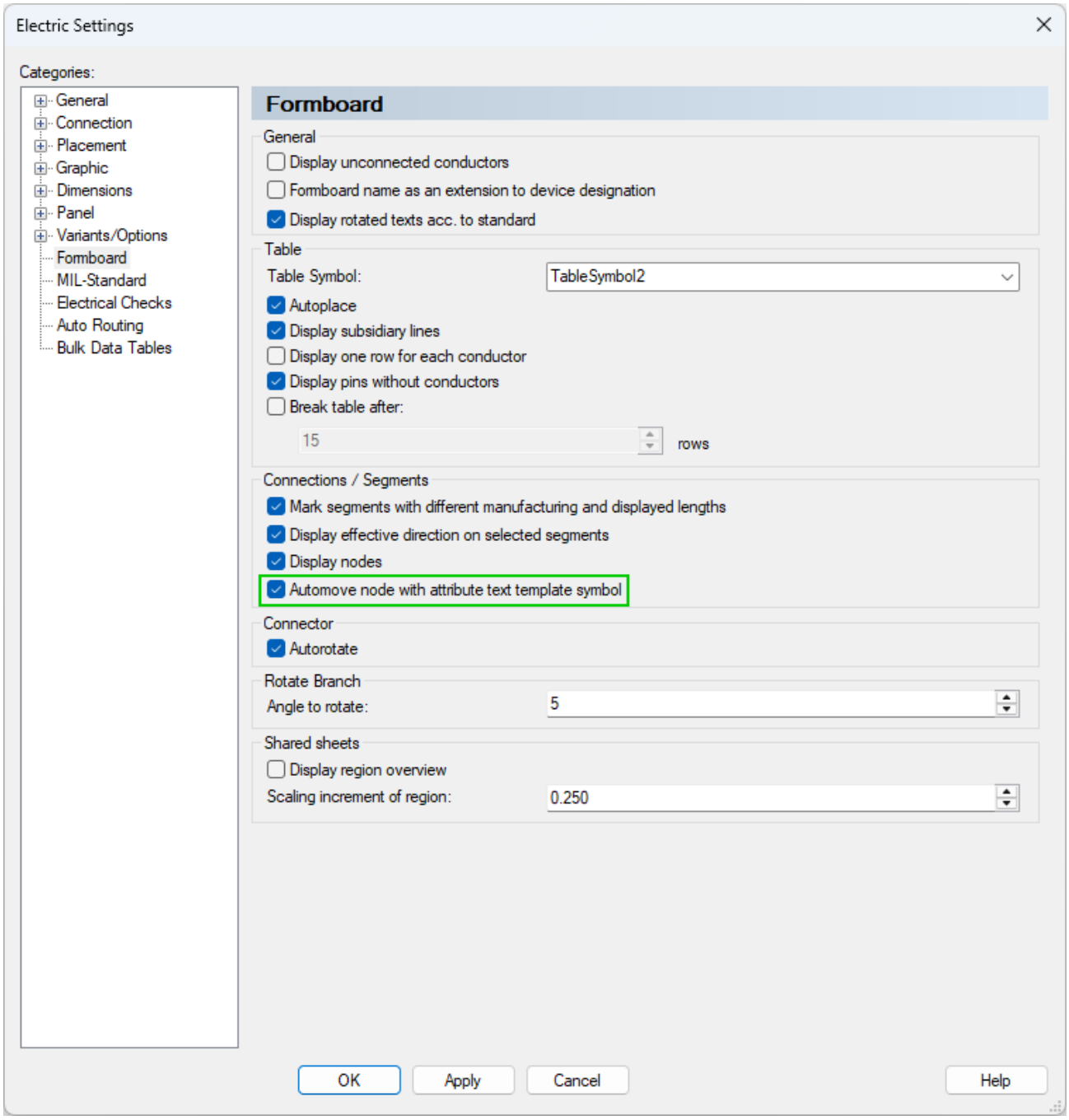
This makes it possible for objects, that need to be connected to nodes when manufacturing (e.g. mountings), to be moved together on their respective formboard sheets.
Thereby nodes stay connected to the attribute text template symbols placed on them, even if a subsequent adjustment is necessary.

To move nodes together with any attribute text template symbols placed on them by default, use either a command in the symbol bar ***Formboard*** or a setting.

To adjust the behavior using the symbol bar, select the command ***Automove node with attribute text template symbol***.

To adjust the behavior using the settings, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

Next, activate the setting ***Automove node with attribute text template symbol*** under ***Formboard***.
The setting is user-specific:



When the setting is active, nodes and attribute text template symbols, that are placed on them, are moved together automatically.
Attribute text template symbols, that are placed on a fixed node, can be moved by keeping the F key pressed.

References: `Designer-40503` and `E3-2404`

Multiuser

- Archiving multiuser projects

Archiving multiuser projects

Multiuser projects can be archived using the ***MUAdmin-Tool*** and the API which reduces the amount of databases needed.
Archived projects are automatically restored when they are used. An archived project is restored when, for example, it is opened, copied or exported.

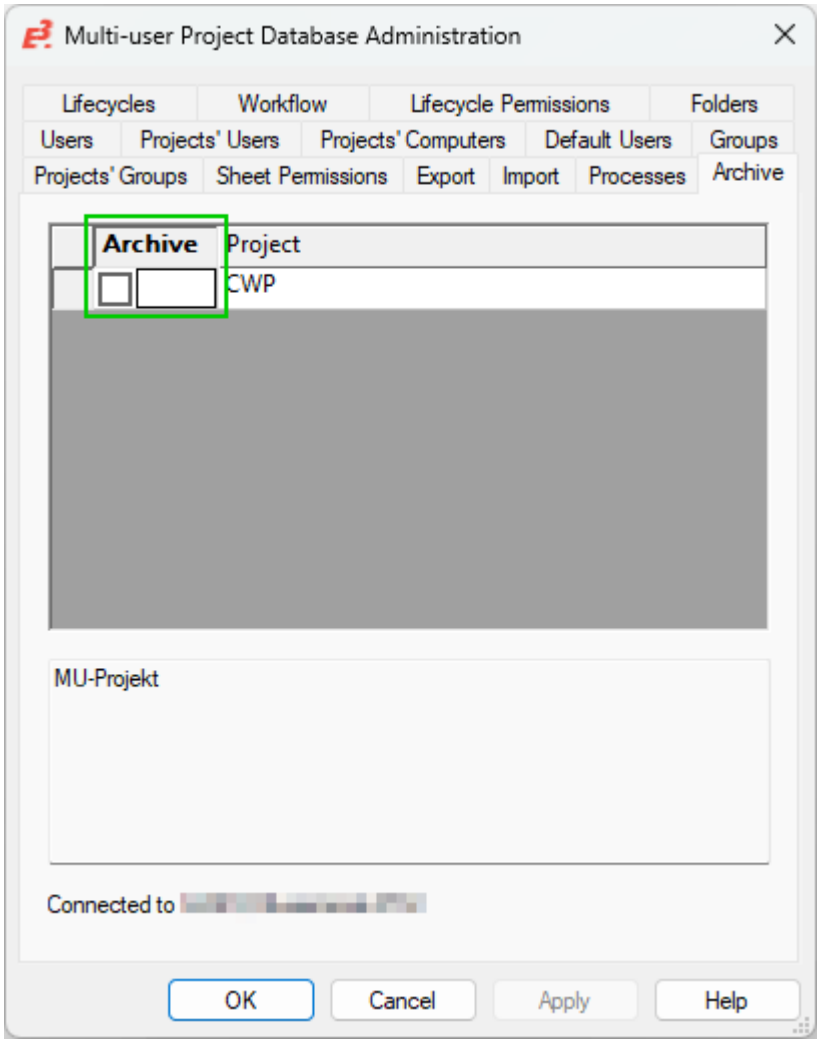
This makes it possible to reduce the used storage space for the multiuser server and the database. Administrators of multiuser servers can, for instance, regularly archive projects that are no longer edited actively.
Furthermore, this reduces costs caused by running multiuser servers and the management of backups.

For this enhancement, the following changes were made:

1. **Enhancement of the application *MUAdmin-Tool* for archiving projects**

In the application ***MUAdmin-Tool*** there is an ***Archive*** tab in which all projects on the server are listed as a table.

To archive projects, select the checkbox in the ***Archive*** column for the desired projects:



The projects are archived with the version of E3.series that they are currently in.

Archived files are automatically stored in a folder, that is defined in the multiuser configuration file and can be selected in the installation assistant ***MuSetup***.

When the project is opened with a newer version of E3.series, it is unarchived and then converted.

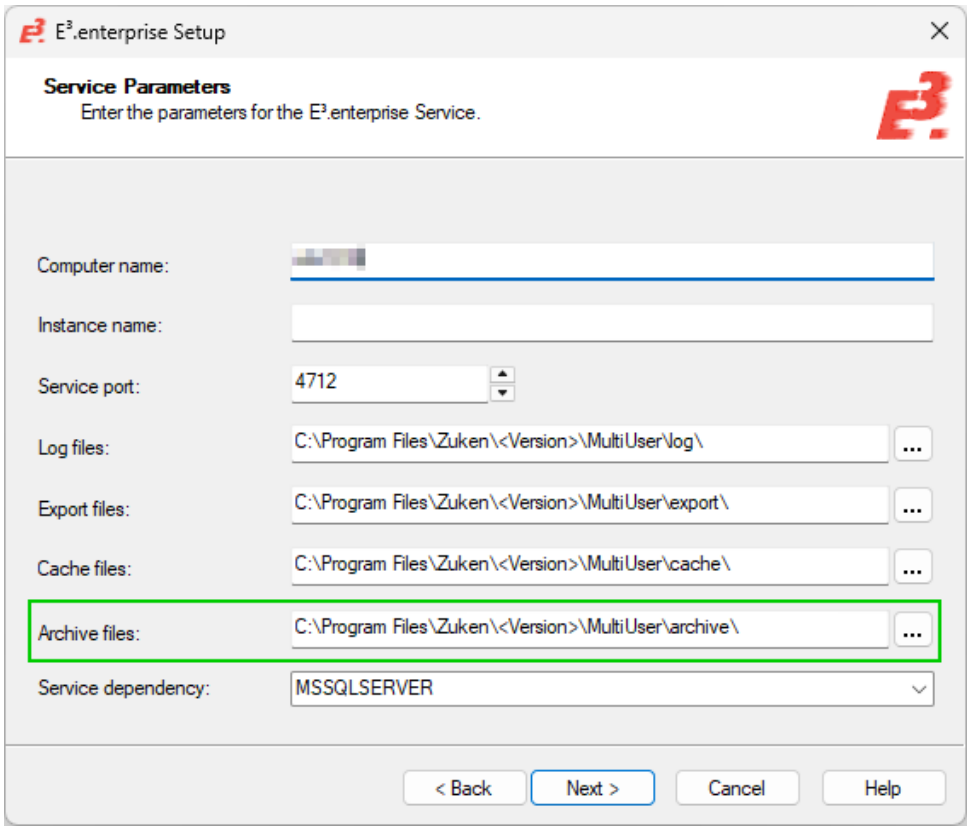
Projects that are already archived, are displayed grayed out and can only be unarchived using ***MUAd-min-Tool*** when they are imported or exported.

The name of the database remains the same as before archiving after unarchiving projects.

---

2. **Enhancement of the multiuser configuration file and the installation assistant *MuSetup***

In the installation assistant it is possible to define the folder for archived projects:



The folder is defined in the multiuser configuration file `configure.xml`.

The XML element **<Archive>** and the element **<Path>** contained within it, have to be used:

XML-Konfiguration

```
<Archive>
    <Path>D:\Temp\Zuken\archive\</Path>
</Archive>
```

Note: It is recommended to define the folder via the installation assistant ***MUSetup***, to avoid potential mistakes when editing the XML file manually.

---

3. **Enhancement of the API**

The API ***MultiuserAdministrationTool*** was enhanced to include the function ***ArchiveProject***.

Project Handling

- Ignoring changes of text types when comparing projects
- Defining 'FONTS.DAT' file with user-specific line types via a setting
- Settings and text types for defining the format of time and date
- Removing empty structure nodes from the project using 'Purge unused objects'
- Placing multiple symbols of components and devices together
- Placing connectors on sheets using drag & drop preferably as single pins
- Text type for displaying bundle names
- Text types for wires, conductors and cables support more characters
- Enhancements for working with clipboards created by Paste Extended

Ignoring changes of text types when comparing projects

The configuration file for the graphical comparison of projects has been enhanced to allow the definition of text types whose content and properties (e.g. font color and size) are ignored during the comparison.

This allows the content and properties of text types to be excluded from the project comparison. This may be the case, for example, if the change to the text type is not relevant to the comparison. That might be the case, for instance, if only the font size of the displayed text has been changed or if changes to the underlying information of the text type are compared via other means, such as the revision status of the sheet.

To ignore the content and properties of text types during the project comparison, modify the configuration file used for the project comparison.

To do this, add the configuration key `TextTypesToIgnore` within the configuration section `[General]` and specify the text types whose changes are to be ignored during the comparison.

To define multiple text types, list them separated by commas.

To ignore text types within a range of values during the comparison, enter the start and end values separated by a hyphen.

Example: `TextTypesToIgnore = 1020, 1041-1044, 1142` ensures that text types 1020, 1041, 1042, 1043, 1044, and 1142 are ignored during the comparison.

References: `E3-1386`

Defining 'FONTS.DAT' file with user-specific line types via a setting

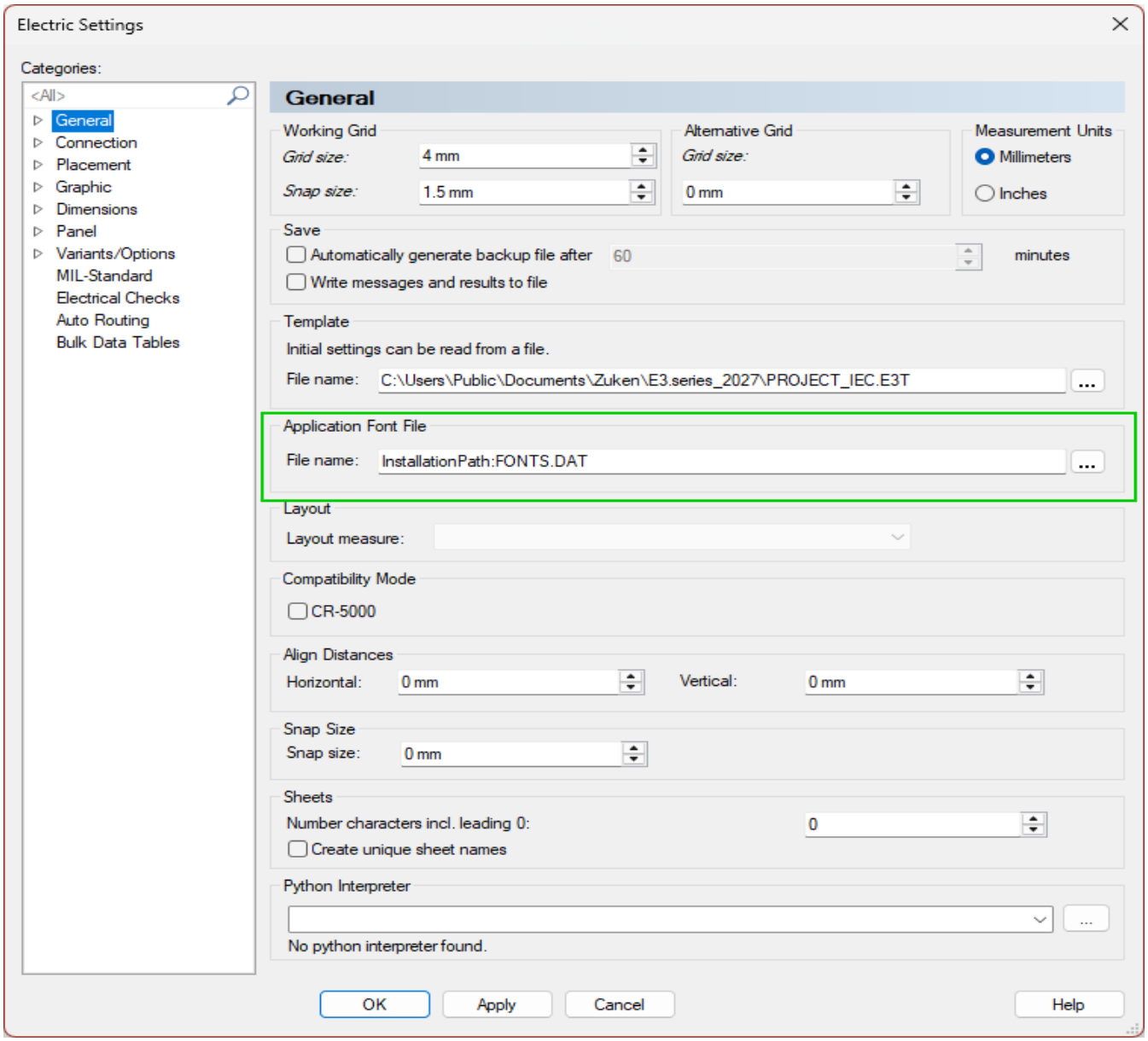
The file `FONTS.DAT` for user-defined line types can be specified via a setting.

This improves the use of custom line types, and the files for them can, for example, be set more easily on different computers.

To define the file for user-specific line types via the settings, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

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

Next, define the desired file under ***General → Application Font File → File name:***.
The setting is user-specific:



To define the path, regular path names, environment variables, `InstallationPath:` and other registry values from `Computer\HKEY_CURRENT_USER\Software\Zuken\E3.series\<E3-Version>` or `Computer\HKEY_LOCAL_MACHINE\Software\Zuken\E3.series\<E3-Version>` can be used.

The defined file path is saved in the registry with the key `Application Font File` under `Computer\HKEY_CURRENT_USER\Software\Zuken\E3.series\<E3-Version>`.

The defined file is used when E3.series is started next and the line types defined in the file can be used.

References: `Designer-28484` and `E3-6776`

Settings and text types for defining the format of time and date

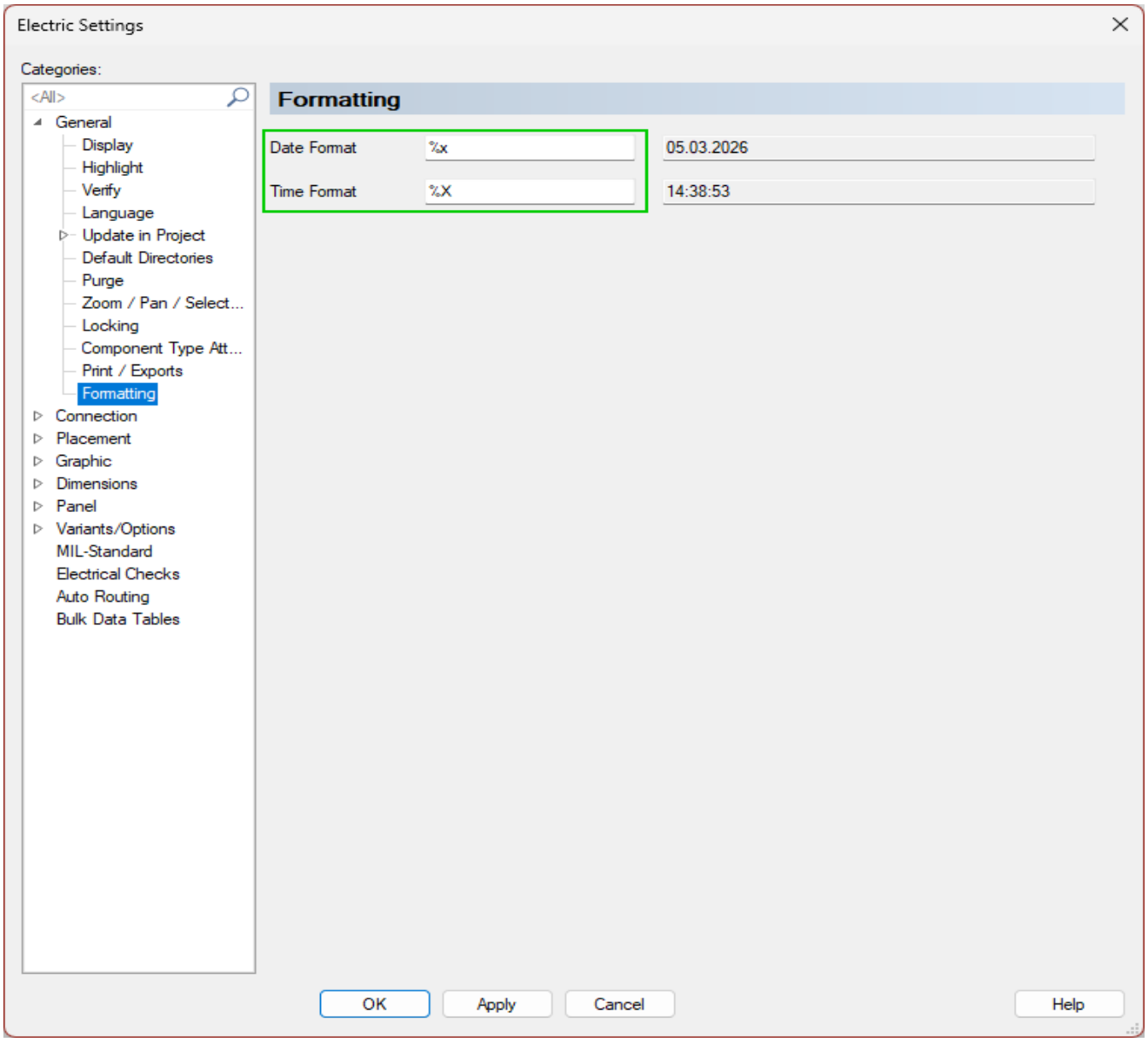
Time and date formats can be freely defined using settings and text types.

This makes, for example, working in international teams with different conventions for writing times and dates easier.

To define time and date formats freely, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

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

Next, define the desired format for times and dates under ***General → Formatting*** for ***Date Format*** and ***Time Format*** .
The setting is user-specific:



The formats can be defined using the following variables:

- `%a`: Abbreviation of the day of the week in the operating system's selected regional format (*Locale*)
- `%A`: The day of the week as displayed in the operating system's regional settings
- `%b`: Abbreviation of the month in the operating system's selected regional format
- `%B`: The month as displayed in the operating system's regional settings
- `%c`: Date and time as displayed in the operating system's regional settings
- `%C`: The result of the calculation *Year ÷ 100*, rounded down to the nearest whole number
- `%d`: The day of the month from 01 to 31
- `%D`: A variable that yields the same result as the format `%m/%d/%y`
- `%e`: The day of the month with a leading zero for single digit numbers
- `%F`: A variable that yields the same result as the format `%Y-%m-%d`
- `%g`: The last two digits of the year according to ISO 8601
- `%G`: The four digit year according to ISO 8601
- `%h`: Abbreviation of the month in the operating system's selected regional format
- `%H`: Hour of the day in the 24-hour clock system (from 00 to 23)
- `%I`: Hour of the day in the 12-hour clock system (from 01 to 12)
- `%j`: Day of the year from 001 to 366
- `%m`: Month of the year from 01 to 12
- `%M`: Minutes of the hour from 01 to 59
- `%n`: A line break
- `%p`: The designation of morning and afternoon using the abbreviations AM (morning, 00:00 to 11:59) or PM (afternoon, 12:00 to 23:59)

  Note: The variable is only replaced when the defined regional format of the operating system supports the 12 hour system.
- `%r`: The current time in the 12-hour format, displayed in the operating system's regional settings
- `%R`: A variable that yields the same result as the format `%H:%M`
- `%S`: Seconds of the minute from 00 to 59
- `%t`: An indentation equal to the width of a tab
- `%T`: A variable that yields the same result as the format `%H:%M:%S` according to ISO 8601
- `%u`: The day of the week as a number from 1 to 7, with counting starting at 1 on Monday according to ISO 8601
- `%U`: Calender week of the year from 00 to 53, with the first day of week 1 being the first Sunday of the year
- `%V`: Calender week of the year from 00 to 53 according to ISO 8601

- `%w`: The day of the week as a number from 0 to 6, with counting starting at 0 on Sunday
- `%W`: Calender week of the year from 00 to 53, with the first day of week 1 being the first Monday of the year
- `%x`: Date format set by the operating system
- `%X`: Time format set by the operating system
- `%y`: The last two digits of the year
- `%Y`: All four digits of the year
- `%z`: The time difference from UTC according to ISO 8601; if the current time zone is unknown, the variable remains empty
- `%Z`: The name of the time zone or the time zone abbreviation, depending on the settings in the registry; if the current time zone is unknown, the variable remains empty
- `%%`: Percent sign

In new projects, `%x` is used automatically for the date format and `%X` for the time format.
A change to the settings applies to all new projects in the current process.

The defined formats will be displayed withe the following new text types:

- Text type number 1188 - ***Generation date formated (sheet)***
- Text type number 1189 - ***Generation time formated (sheet)***
- Text type number 1190 - ***Date of modification formated (sheet)***
- Text type number 1191 - ***Time of modification formated (sheet)***
- Text type number 1192 - ***Plot date formated***
- Text type number 1193 - ***Plot time formated***
- Text type number 1194 - ***Project store date formated***
- Text type number 1195 - ***Project store time formated***

Preexisting text types for displaying dates and times, for example text type number 26 - ***Date***, are not affected by the user-defined formatting.

References: `Designer-05992` and `E3-1348`

Removing empty structure nodes from the project using 'Purge unused objects'

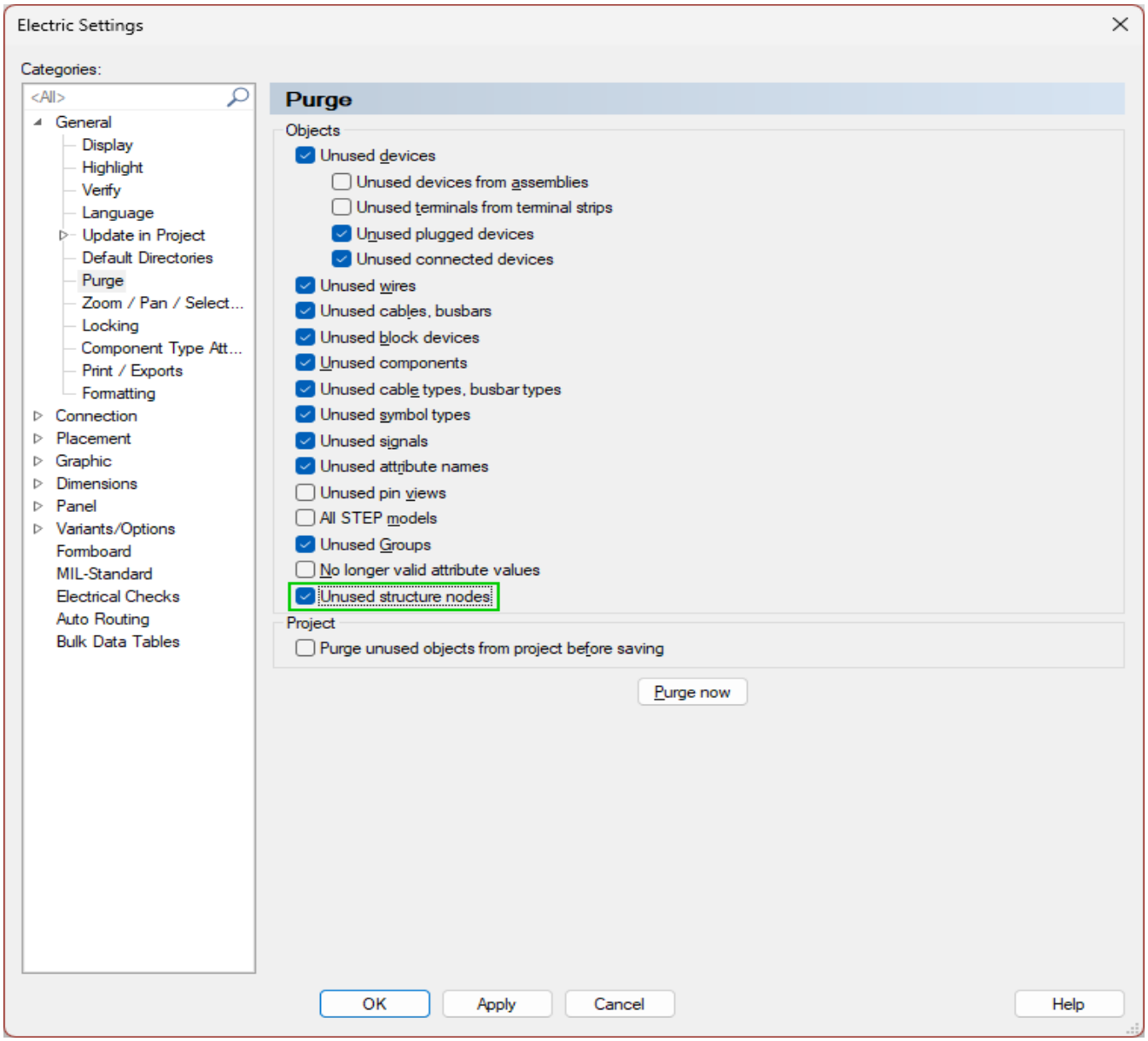
Empty structure nodes can be deleted with the command ***Purge Unused Objects***.

This makes it possible to delete structure nodes, that are no longer needed for the development process, from the project.

To delete empty structure nodes from the project, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

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

Next, activate ***Unused structure nodes*** under ***General → Purge***.
The setting is project specific:



When the command ***Purge Unused Objects*** is used, the option ***Unused structure nodes*** in the ***Purge*** dialog has the value defined in the ***Electric Settings*** or ***Fluid Settings*** as its default value.

Independent of the setting ***Electric Settings → Purge*** and ***Fluid Settings → Purge*** the behavior can also be selected and deselected dynamically before deleting.

References: `Designer-09098` and `E3-4004`

Placing multiple symbols of components and devices together

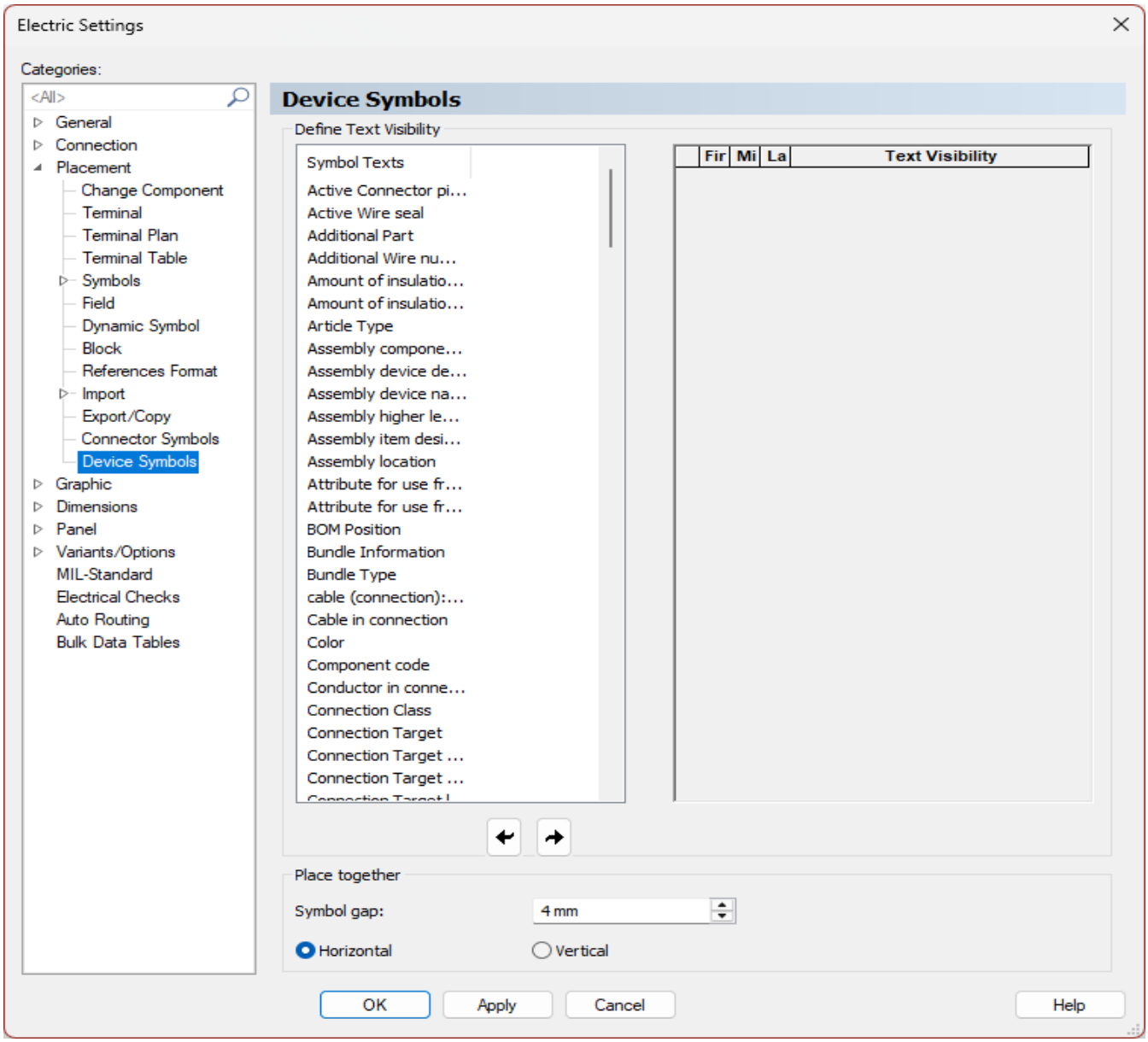
From the device and database tree it is possible to place multiple symbols of a device or component. It is possible to place either all unplaced symbols of a component or only part.

This makes it possible to place, for example, all PLC entries of a device all at once.
Overall, the enhancement makes placing symbols more user-friendly and efficient.

To be able to place multiple symbols of a component or device together, open a sheet and select a component, a device or is desired symbols in the device or database tree.

Next, open the context menu and select the command ***Place Together***.
The symbols are attached to the cursor and can be placed.

The displayed symbol texts, orientation of the symbols and the distance between them can be adjusted in the ***Electric Settings*** or the ***Fluid Settings*** under ***Placement→ Device Symbols***:



The settings are user-specific.

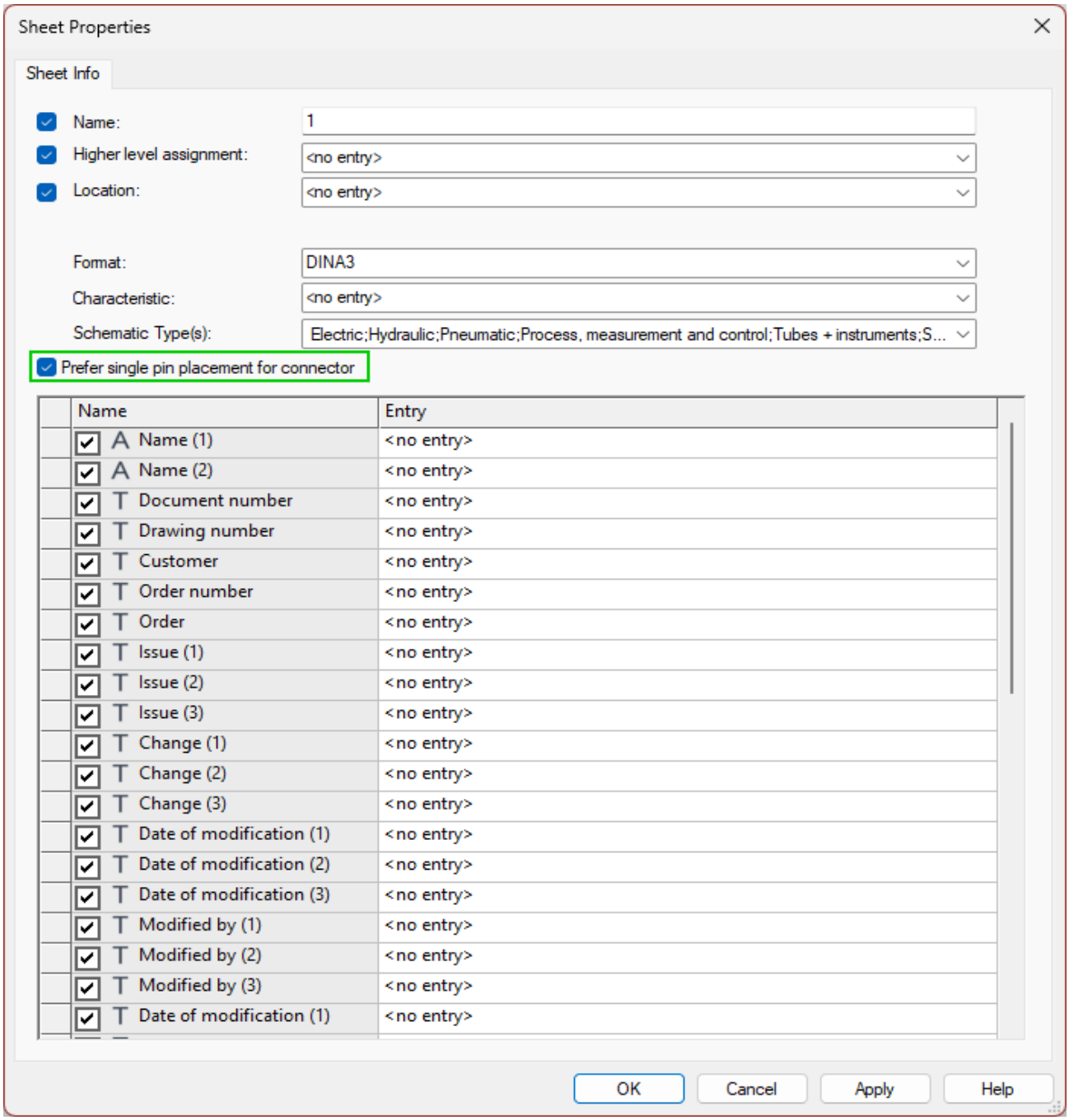
Placing connectors on sheets using drag & drop preferably as single pins

With a new sheet property it is possible to place connectors, that are placed using drag & drop, preferably as single pins.

This makes it possible to, for example, reduce the amount of steps required for sheets on which only single pins instead of pin groups are to be placed.

To define for single sheets in the project, that single pins are used preferably when placing connectors with drag & drop, open the context menu in the sheet tree of the required sheet or on the sheet itself and select ***Sheet Properties...***
The ***Sheet Properties*** dialog opens.

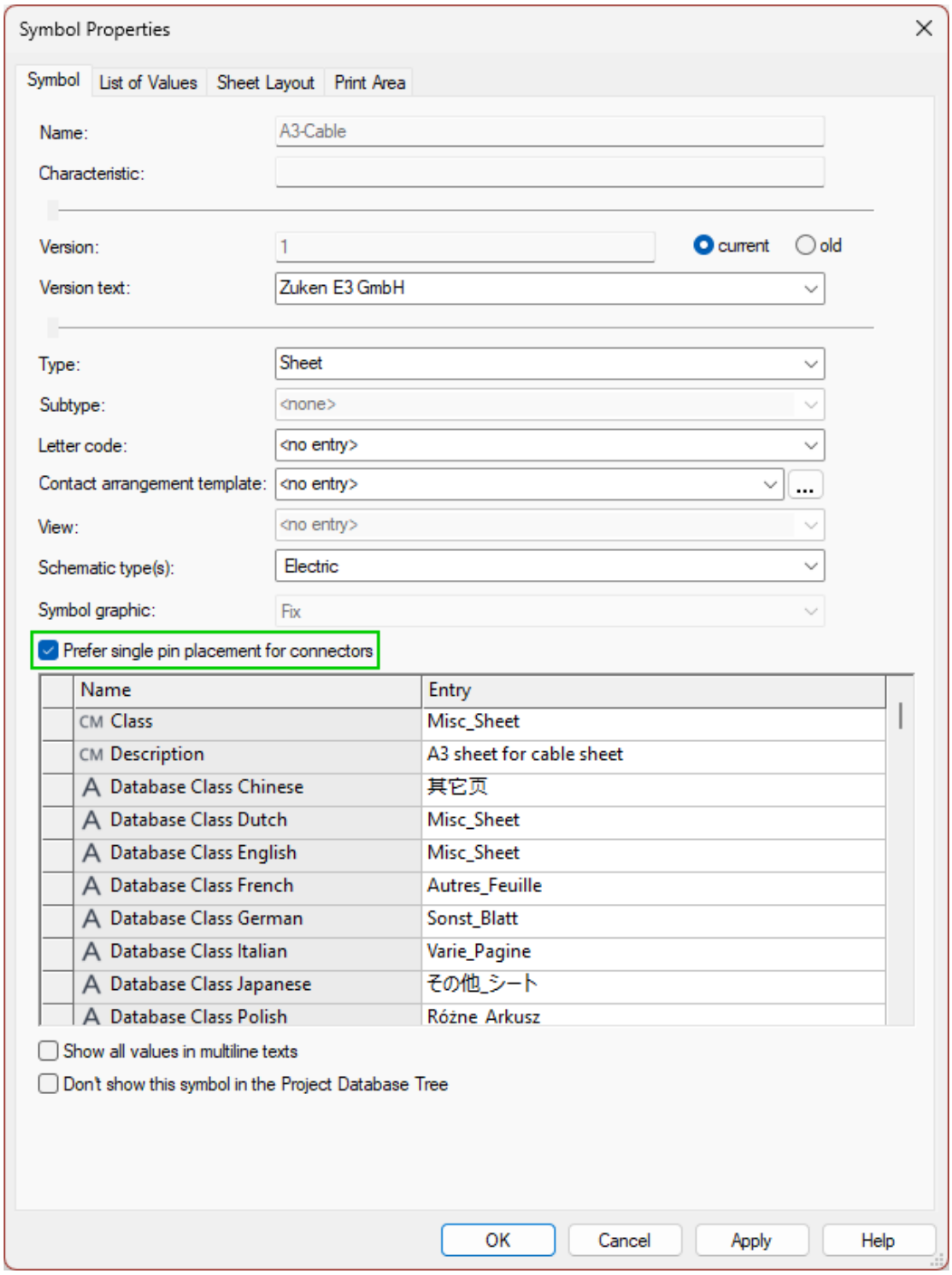
Next, activate the property ***Prefer single pin placement for connectors***:



To define for all sheets of a sheet format, that single pins are used preferably when placing connectors with drag & drop, open the database editor.

Edit the desired sheet format, open the context menu of the sheet symbol in the project window or the work space of the database editor and select ***Properties...*** or ***Symbol Properties***.
The ***Symbol Properties*** dialog opens.

Activate the setting ***Prefer single pin placement for connectors***:



Connectors, that are placed form the tree on a sheet with this property using drag & drop, are placed as single pins.
The behavior is analog to the command ***Place Single Pins***

Changes to the setting do not affect previously placed pins.

References: E3-6048

Text type for displaying bundle names

Names of bundles, that are assigned to wires/conductors, can be displayed with a text type.

This makes it easier to maintain and display information about bundles of wires/conductors in the project.

To display bundle information in the project use the text type number **1196** - ***Bundle Information***.

The text type can be assigned to normal symbols, connector symbols and formboard table symbols.

For normal symbols and connector symbols, the text type must be assigned to a pin.
For formboard table symbols, the text type must be assigned to a row or sub-row.

When the wires/conductors are assigned to a bundle and the bundle is part of another bundle, the individual bundle names are displayed comma-separated.
When multiple wires/conductors are connected to the pin, the bundles of the individual wires/-conductors are displayed separated with a comma and a space.

References: `E3-8117`

Text types for wires, conductors and cables support more characters

Text types, that display information about wires, conductors and cables, support more characters than previously.

This allows

This allows the text types to be longer and, as a result, to display more information.

For the following text types the maximum length was increased to **2048** characters:

- Text type number 1003 - ***Conductor in connection***
- Text type number 1005 - ***Cable type in connection***
- Text type number 1008 - ***Length of segment***
- Text type number 1024 - ***Wire group***
- Text type number 1025 - ***wire***
- Text type number 1026 - ***Cross-section***
- Text type number 1027 - ***Color***
- Text type number 1168 - ***Default wire group***
- Text type number 1169 - ***Default Wire Type***
- Text type number 1170 - ***Default wire color***
- Text type number 1171 - ***Default wire cross-section***
- Text type number 1172 - ***Default wire outer diameter***
- Text type number 1173 - ***Default wire name***
- Text type number 1174 - ***Outer Diameter***

Enhancements for working with clipboards created by *Paste Extended*

1. **View for the clipboard *Merge as view***

It is possible to insert views from the clipboard to the project using the command ***Merge as view***. Additionally, the views from terminal strip views can be merged.

This makes it possible to not only merge views in the clipboard with preexisting objects, giving them new names, but also as views, giving them new view numbers.

The command ***Merge as view*** is available when only originals or only originals with exactly one view are available in the clipboard.

When there are originals with exactly one view in the clipboard, the original has to be completely unplaced and the view number must be between 1 and 99. Formboard views are not supported.

References: `Designer-35648` and `E3-3965`

---

2. **Assemblies, connectors with inserts and terminal strip views *Merge as view***

Assemblies, connectors with inserts and terminal strip views, that are contained in a clipboard, can be inserted in the project using the command ***Merge as view***.

This makes it possible to, for example, save assemblies, connectors with inserts and terminal strip views and then merge them with preexisting objects of the project.

The command merges all elements that can be merged.
Objects in the clipboard, that cannot be merged, remain in the clipboard.

Terminal strip views can only be merged as views when the terminal strips do not use terminals that are already available for terminal views.

References: `Designer-33668` and `E3-2908`

---

3. **Renaming clipboards**

Clipboards, that are available in the project, can be renamed.

This improves the overview of available clipboards by, for example, summarizing the contents of each clipboard using custom names.

To rename a clipboard, open the context menu of the desired clipboard, select the command ***Rename*** and enter the new name.
Alternatively, select a clipboard, click F2 and enter a new name.

Note: The same name can be used for multiple clipboards.

Panel (also: 3D)

- Creating 2D manufacturing views of devices, models and slots
- Routing conductors of cables through cable ducts in the same path
- Pins with different signals can be connected automatically
- Prefer entry to cable ducts with same HLA and LOC as respective device when routing automatically
- Enhancements for defining the visibility of devices, conductors and wires
- Colored highlighting of conductors/wires, tubes/hoses and busbars that are not connected in the panel
- Additional command for automatic routing and enhancement of statistic

Creating 2D manufacturing views of devices, models and slots

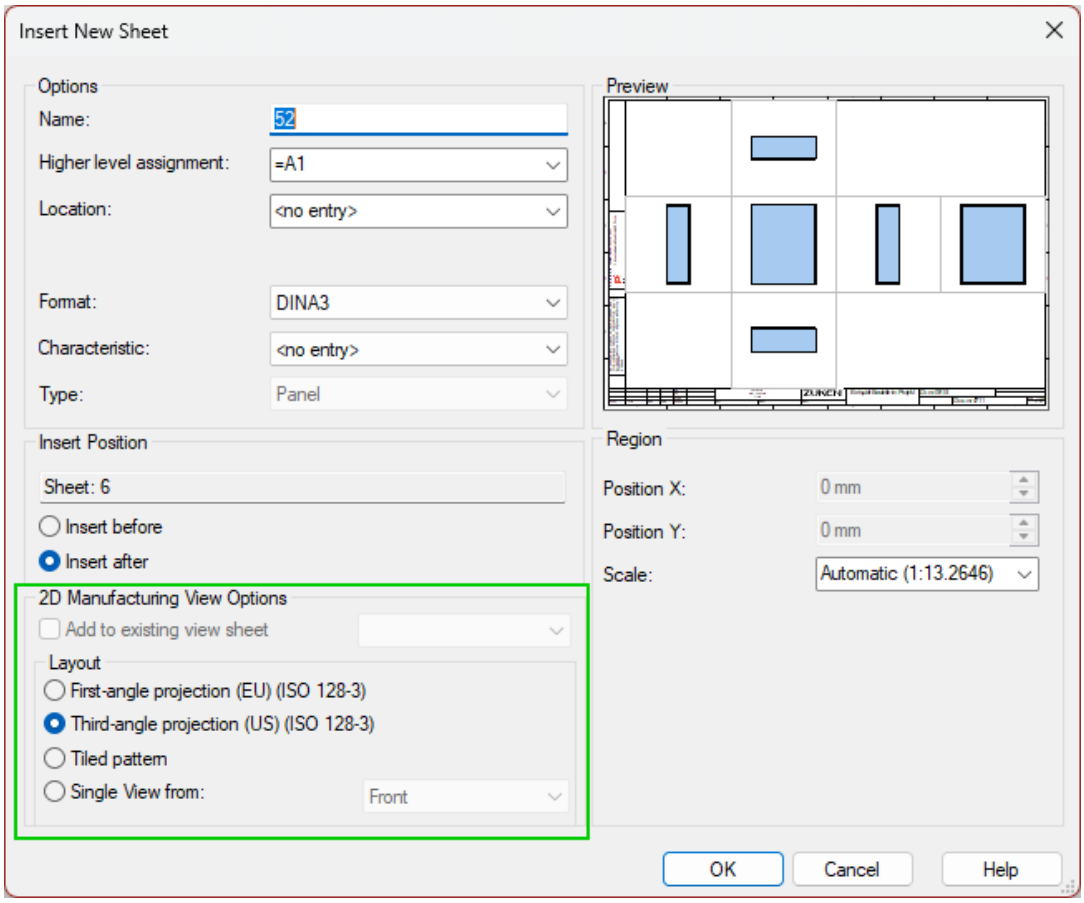
For manufacturing panels, it is possible to create a single, standardized 2D manufacturing view of devices, models and slots displaying all 6 three-dimensional views (***Front***, ***Back***, ***Right***, ***Left***, ***Top*** and ***Bottom***)

This allows devices, models, and slots—along with all objects placed on them—to be displayed on a single sheet from various perspectives, thereby enabling manufacturing processes to be organized more efficiently overall.

To create a 2D manufacturing view of objects, select a placed object in a project window ***Panel - Placed*** or ***device*** or one a panel sheet.

When the 2D manufacturing view is created based on an object in a project window, the object is displayed with all objects placed on it.
When the 2D manufacturing view is created based on an object on a sheet, the bottommost owner object is displayed with all objects placed on it.

Next, select the command ***Create 2D manufacturing view*** in the context menu.
The ***Insert New Sheet*** dialog opens:



Define general settings, that are to apply to the new sheet, under ***Options***, ***Insert Position*** and ***Region***.

Note: ***Position X:*** and ***Position Y:*** can only be set when the option ***Single View from:*** is selected. When values are changed, the position of the view is moved with in the view area.

Scale: is, by default, always defined so that all individual views in the manufacturing view can be displayed as well as possible. The scale always affects all view that are created.

Under ***2D Manufacturing View Options*** the following changes can be made specifically for 2D manufacturing views:

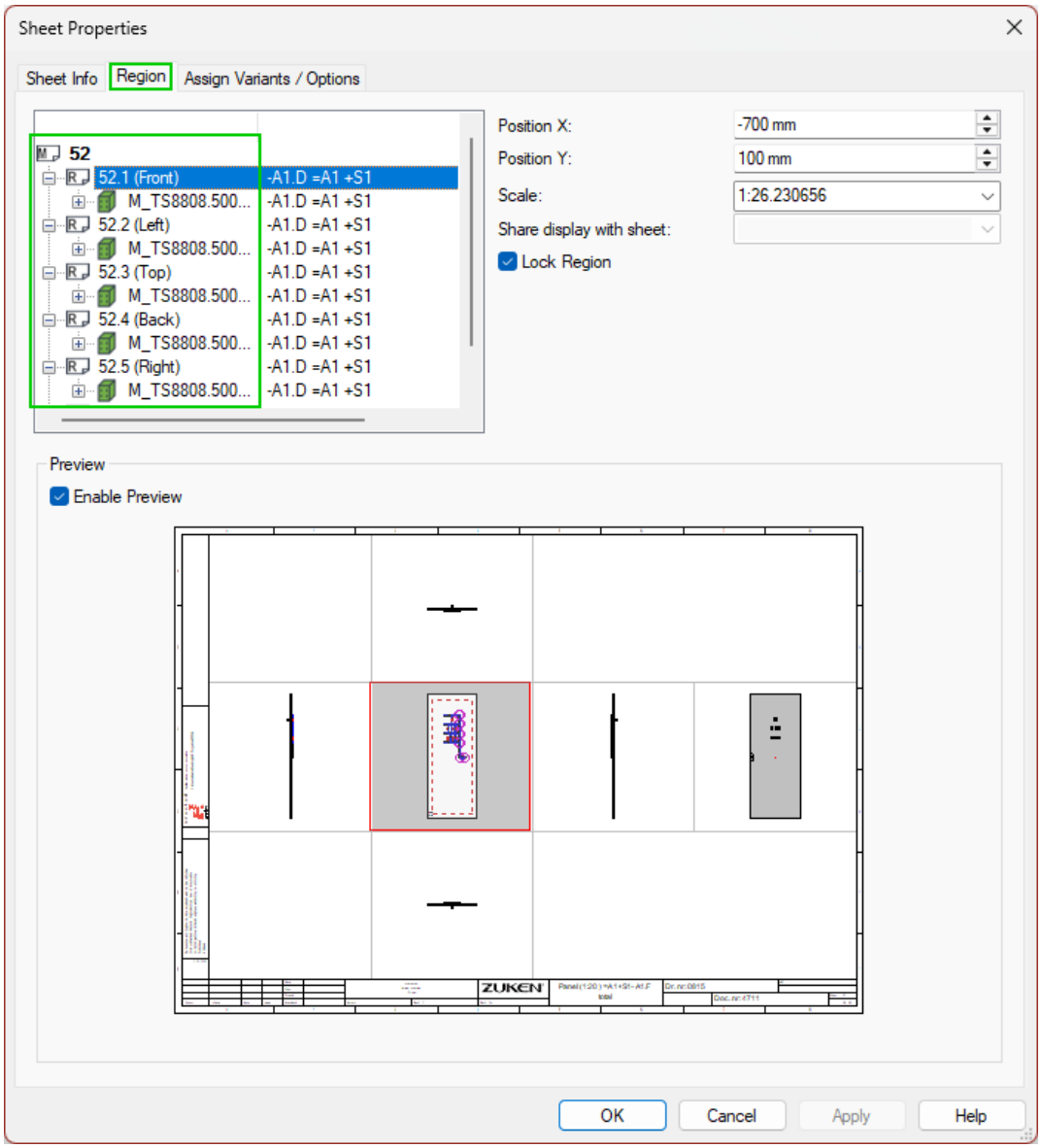
- ***Add to existing 2D view sheet***: When a sheet of a 2D manufacturing view has already been created in the project, a view can be selected under ***Single View from:*** that is added to the sheet.
- ***First-angle projection (EU) (ISO 128-3)***: Creates all 6 views of the object and arranges them according to ISO 128-3 (as used predominantly in the EU).
- ***Third-angle projection (US) (ISO 128-3)***: Creates all 6 views of the object and arranges them according to ISO 128-3 (as used predominantly in the US).
- ***Tiled pattern***: Creates all 6 views of the object.
  On sheets in portrait mode, the views are arranged in two columns with three views each.
  On sheets in landscape mode, the views are arranged in two rows with three views each.

- ***Single View from:*** Creates on of the views of the object ***Front***, ***Back***, ***Right***, ***Left***, ***Top*** or ***Bottom***.
  When the option is selected, the area to be created is highlighted with a red border.
  The area can be moved and scaled.

Sheets with 2D manufacturing views are automatically updated, when the objects displayed on them are edited in the panel.
Additionally, texts, graphics and dimensioning can be added.

The command ***Jump*** → ***Jump to 2D Manufacturing View*** in the context menu can be used to jump from objects to already created 2D manufacturing views.

In the sheet properties of a 2D manufacturing view, a separate region is displayed for each of the 6 views:



References: E3-2859

Routing conductors of cables through cable ducts in the same path

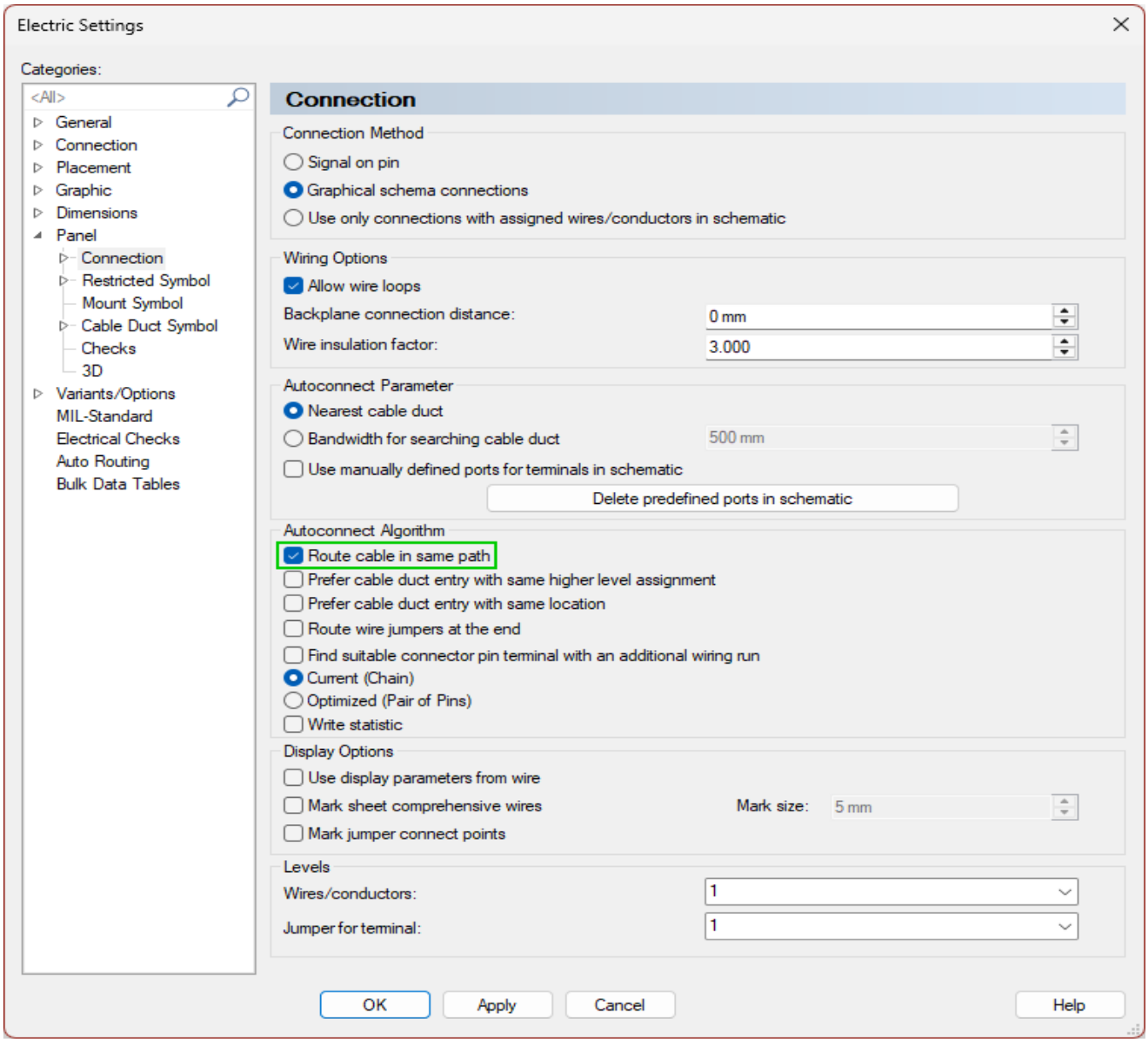
With a setting and property it is possible to ensure that conductors of cables are routed through cable ducts in the same path.

Conductors of cables are thereby routed through the cable duct system in the same path, even if the wires would have a shorter routing path when considered individually.

To activate the setting ***Route cable in same path*** select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

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

Next, activate the setting ***Route cable in same path*** under ***Panel → Connection***.
The setting is project specific:



When the setting is active, conductors of cables are routed in the same path.
The following objects are not affected by the setting:

- Wires of cables
  Note: Wire of dynamic cables are affected by the setting
- Cable bundles and parts of cable bundles
- Hoses and tubes

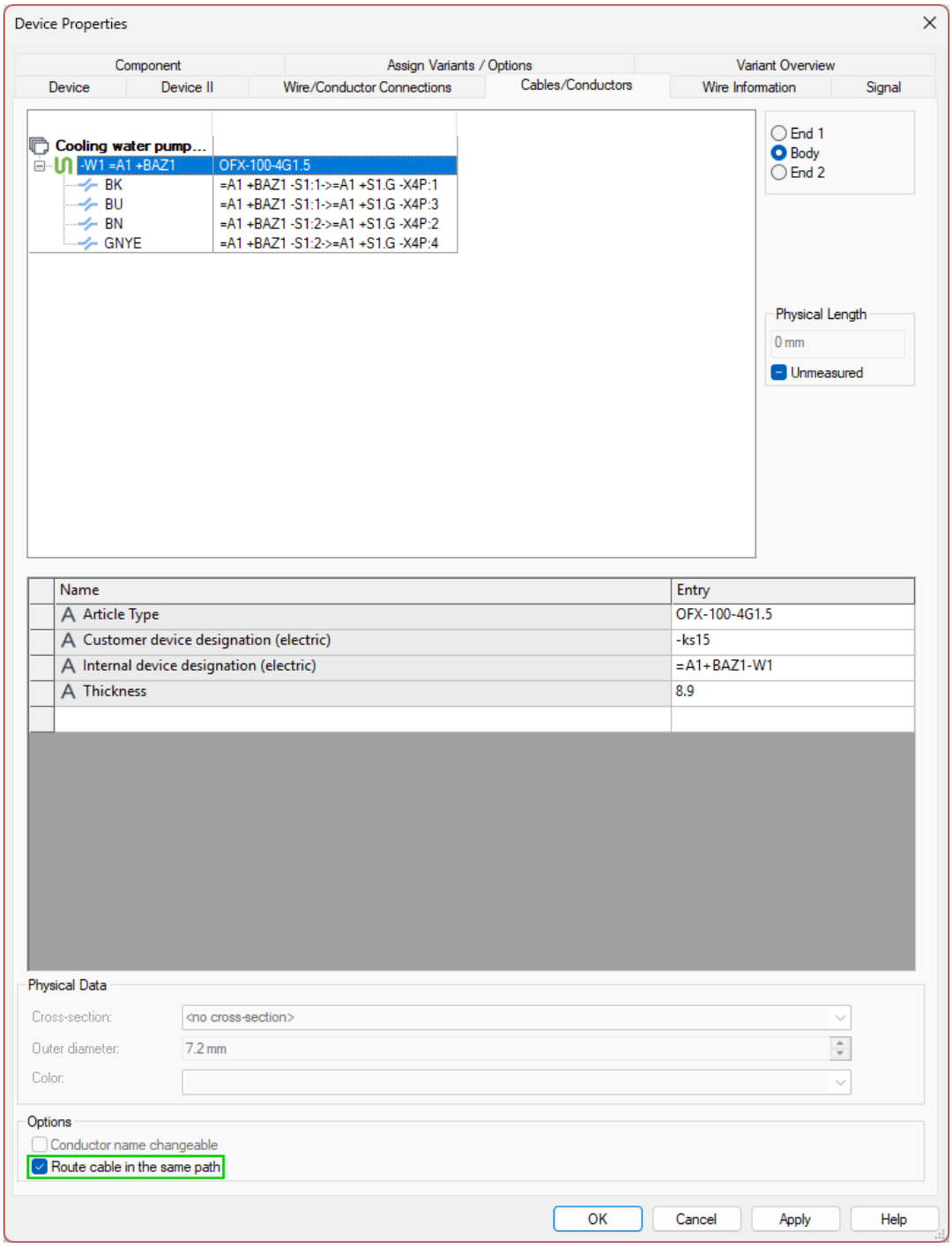
To determine the shared path through the cable duct, the available cable duct entry and exit points are examined. It is also checked whether there are incompatible attributes for automatic connection in the panel that prevent the routing of individual conductors.
When there are multiple shared routing paths possible for the conductors, the one with the shortest manufacture length is used.

For cables, that have a component definition, the property ***Route cable in same path*** is active by default.
For cables, that have don't have a component definition, the property is `Inactive` by default.

The setting can also be adjusted individually for every cable.

To do so, open the device properties of a cable.

Then select the ***Cables/Conductors*** tab and activate or deactivate the property ***Route cable in same path***:



Alternatively the setting can be activated or deactivated in the device table.

Pins with different signals can be connected automatically

Pins of devices, that have different signals, in the panel can be connected automatically using the commands ***Panel Autoconnect*** and ***Panel Autoconnect (both ends selected)***.
The wiring statistic in the ***Results*** window lists all connections that could not be routed.

As a result, connections are routed even if they have different signals or if a pin has no signal.

Even if there is no graphic or signal carrying connection, the pins are routed automatically.

The behavior is independent of the connection method defined under ***Panel → Connection***.

References: E3-5367

Prefer entry to cable ducts with same HLA and LOC as respective device when routing automatically

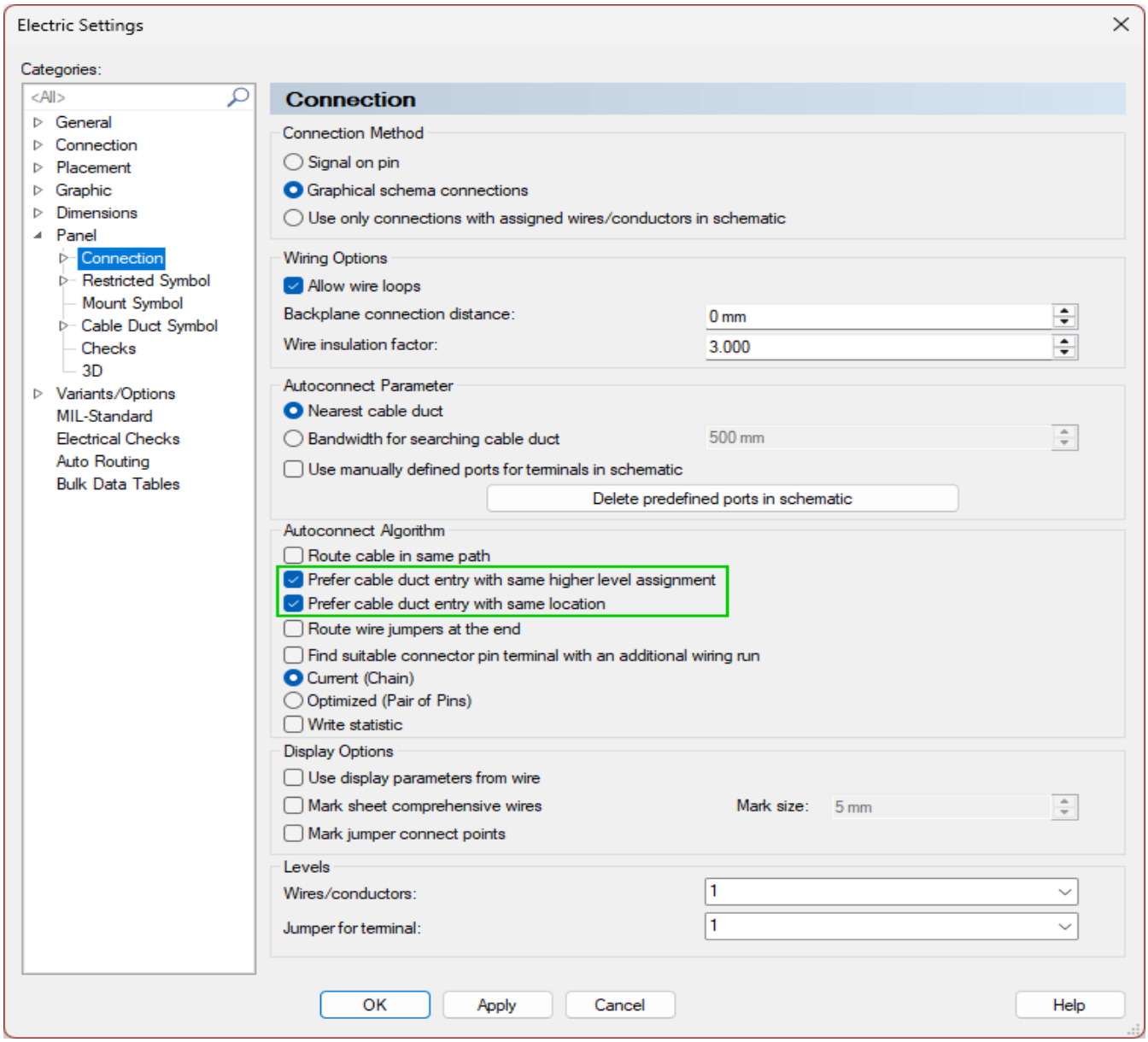
When routing automatically it is possible to prefer cable ducts, that have the same higher level assignment or location as the devices that are to be routed, for the cable duct entry.

This makes it possible to apply another general rule to automatic routing making the results of the routing more predictable.

To do this, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

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

Activate the setting ***Prefer cable duct entry with same higher level assignment*** or ***Prefer cable duct entry with same higher level assignment*** under ***Panel → Connection***.
The settings are project specific:



When one of the settings is active, cable ducts, that have the same higher level assignment or location as the device to be wired, are preferred as cable duct entries.
In some cases, cable ducts, that are further away than others, have to be used.

When there are multiple possible cable ducts, that could be used as cable duct entries, based on the settings, the cable duct with the shortest distance is used.

Cable ducts, that are defined for devices under ***Device Properties → Slots/Pins → Destination cable duct***, are preferred when routing automatically regardless of their higher level assignment or location.

Enhancements for defining the visibility of devices, conductors and wires

The possibility of displaying objects transparently in the panel was expanded to include conductors and wires.

Additionally, the display options were enhanced to be able to define different transparency values for different types of objects.

This makes working on panels in constructing easier, for instance by making all objects, that are not required at a specific point, transparent depending on the individual work step.

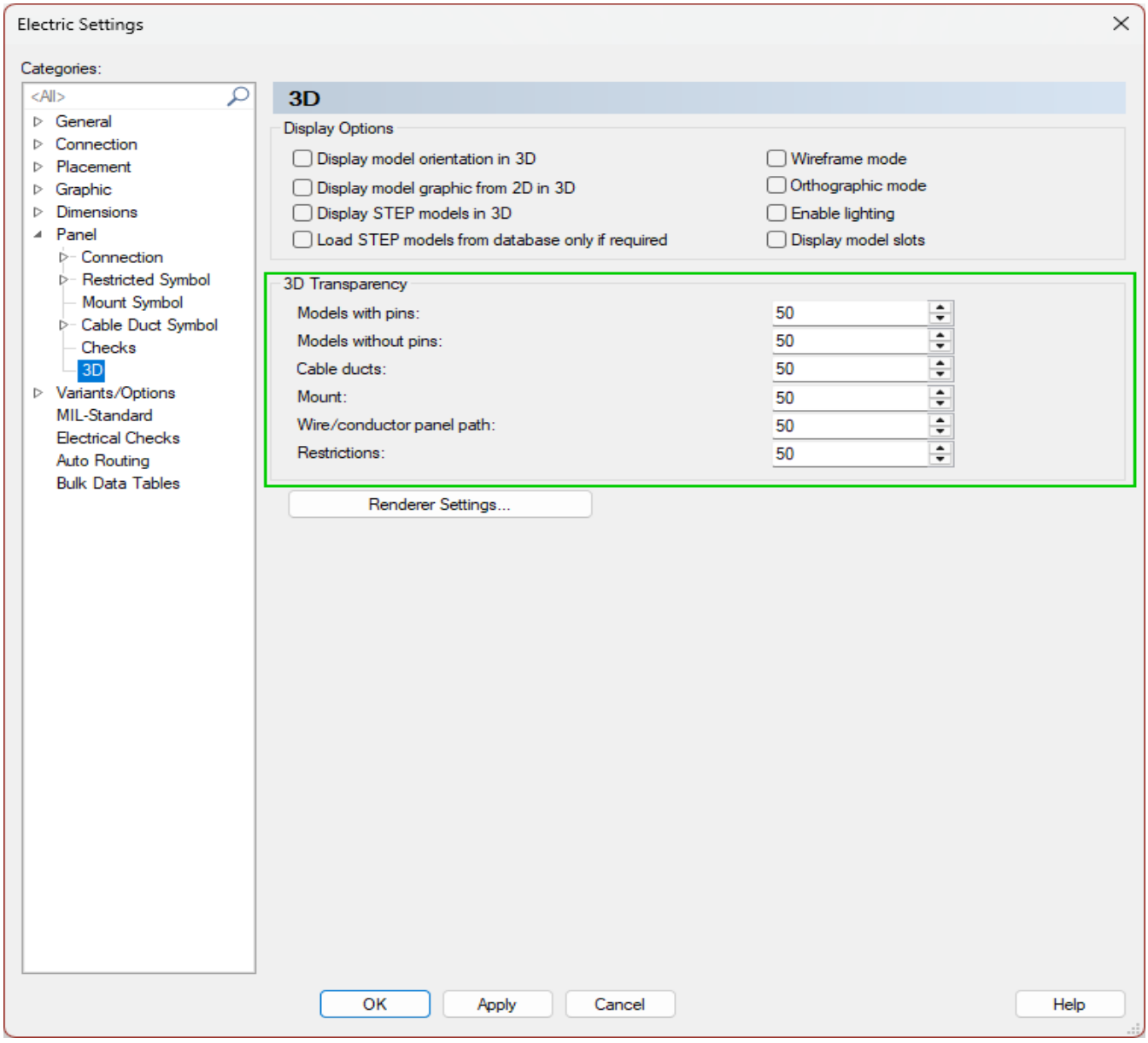
The following options are available for defining the transparency of individual objects:

1. **Defining transparency via the settings**

To define the transparency for objects via the settings, select ***Tools → Settings...*** in the main menu window.
The ***Electric Settings*** dialog opens.

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

Define the desired transparency for the available objects under ***Panel → 3D → 3D Transparency***.
The setting is user specific:



Transparency values between 10% for low transparency and 90% for high transparency, can be defined.

Note: As part of the enhancement, the setting ***Panel → Display Options*** and the button ***Renderer Settings...*** were moved to the ***Panel → 3D*** menu.

---

2. **Defining transparency via the symbol bar**

To define the transparency of objects via the symbol bar display the symbol bar ***3D Transparency***.

The transparency can be defined individually for the following objects:

- ***Models with pins:***
- ***Models without pins:***
- ***Cable ducts:***
- ***Mount:***
- ***Wire/conductor panel path:***
- ***Restrictions:***

For the available objects, transparency values between 10% for low transparency and 90% for high transparency, can be defined.

Confirm the changed values with .

When there are no changed values, that need to be confirmed, the symbol bar instead displays .

To change all values back to the default value 50, click .

Referenzen: `E3-5134` and `E3-5127`

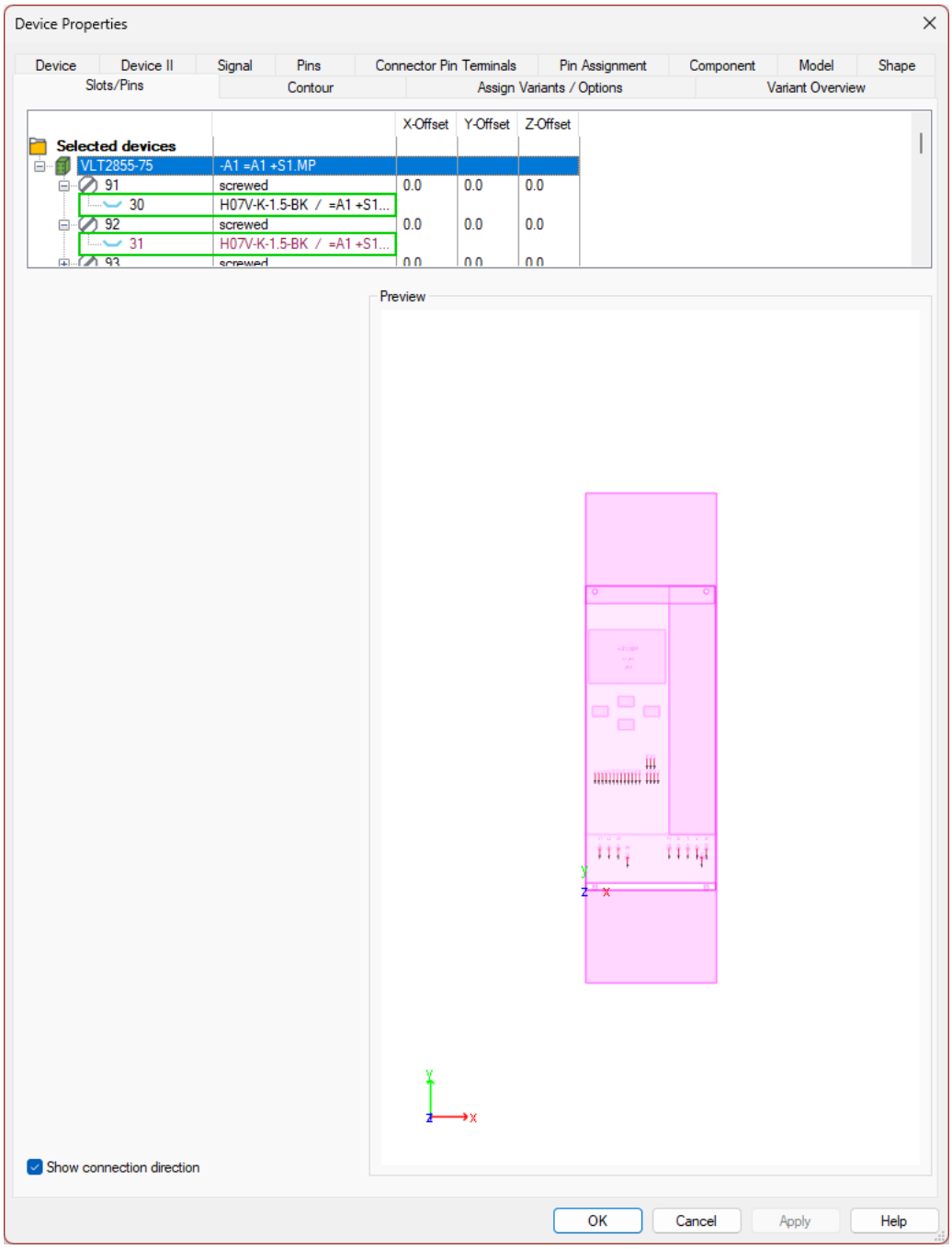
Colored highlighting of conductors/wires, tubes/hoses and busbars that are not connected in the panel

Conductors/wires, tubes/hoses and busbars of connections between devices, that are not connected in the panel in the schema, are highlighted in color in the device settings.

This makes it easier to identify and, if necessary adjust, conductors/wires, tubes/hoses and busbars that can be used for connections, but do not have a respective connection in the panel.

To check connections of a device for connected conductors/wires, tubes/hoses and busbars, open the context menu of the placed device in the device tree or on the sheet and select ***Device Properties...*** The ***Device Properties...*** dialog opens.

Next, select the ***Slots/Pins*** tab:



Slots with connected conductors/wires, tubes/hoses or busbars have an additional sub-menu in which the name of the connected object is displayed.

Conductors/wires, tubes/hoses and busbars of connections between devices, that are not connected in the panel, are written in magenta.
Conductors/wires, tubes/hoses and busbars of connections between devices, that are connected in the panel, are written in black.

Additional command for automatic routing and enhancement of statistic

There is an additional command for the automatic routing of devices in the panel that allows devices to routed only if both a start and a target pin, that belongs to the device to routed, is selected.

Additionally, the statistic was enhanced to show whether any connections could not be assigned and if so between which pins. This can be the case when the pins of the devices are on different sheets.

This makes it possible to limit the automatic connection to the selected devices and unrouted devices can be identified more easily.

To route selected devices in the panel automatically, select the devices or pins that are to be routed automatically.

Select the command ***Panel Autoconnect*** or ***Panel Autoconnect (both ends selected)*** under ***Insert → Autoconnect*** in the context menu or the main menu bar.

Information about the automatically routed connections are output in the ***Results*** window.
This includes the used connection method, the used routing method and which connections could not be routed.

References: `Designer-33479` and `E3-156`

Variants/Options

- Jumping to the 'Variants/Options' tree from the context menu of objects with variants/options
- Editing variants/options in the sheet and device table

Jumping to the 'Variants/Options' tree from the context menu of objects with variants/options

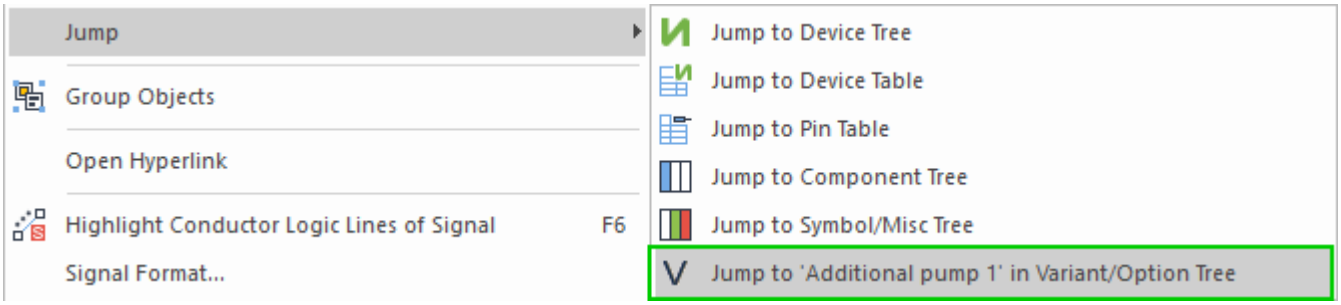
It is possible to jump directly to the tree ***Variants/Options*** via the context menu of objects, that have variants/options assigned to them.

This makes it easier to view which variants/options are assigned to selected objects and the workflow for variants/options becomes therefore generally more user-friendly.

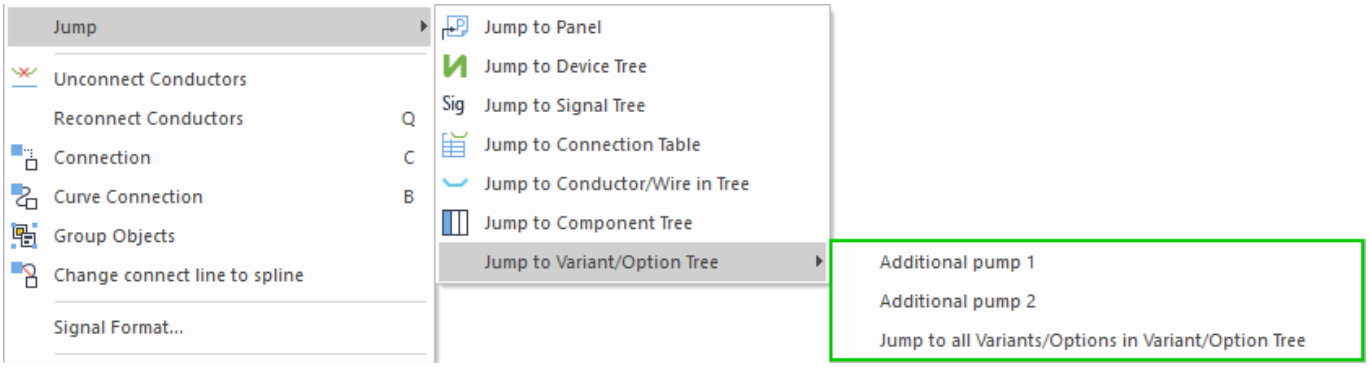
To jump to the ***Variants/Options*** tree from an object, that has variants/options assigned to it, select the object on a sheet in a table or in the tree view and open the context menu.

Note: Variants/Options can only be jumped to via a connection in the connection table, if a conductor/wire or a tube/hose is connected and the conductor/wire or a tube/hose has a variant/option assigned to it.

When the selected object is assigned to only one variant/option, the command to jump to the ***Variants/Options*** tree can be found in the submenu ***Jump***.



When the selected object is assigned to multiple variants/options, the commands for jumping to the possible targets in the ***Variants/Options*** tree can be found in the submenu ***Jump → Jump to Variant/Option Tree***.
Up to 30 variants/options can be displayed. The simple variant/option name is displayed.
When the simple variant/option name is ambiguous, the full name is displayed instead.



Editing variants/options in the sheet and device table

Variants and options can be adjusted in the sheet and device table.

This makes it possible to use the sheet and device table to, first, filter and sort all objects and then change the variant/option assignment.

To edit variants/options in the sheet or device table, open the respective table and add the ***Variants/Options*** column.

In the following cases variants/options are only displayed in the sheet and device table and cannot be changed:

- When multiple expressions for assigning variants/options are defined for an object.
  The individual expressions are displayed as comma-separated in the table.
- When the state ***Does not exist*** or ***Only exists*** is defined for an object.
  These states are output as text in the tables.

When variants/options are defined in the project, the assignment of variants/options can be changed in the sheet and device table as follows:

- To assign variants/options to sheets or devices, enter the name of the variant/option in the ***Variants/Options*** column.
- To remove assigned variants/options for sheets or devices, remove the entry in the ***Variants/Options*** column.
- To change assigned variants/options of sheets or devices, adjust the entry the ***Variants/Options*** column.

References: E3-4032

Connections/Busses, Signals/Logic Lines, Supplies

- Keeping bundle symbol types when updating or switching cables and overbraids
- Highlighting intersecting, unconnected connection lines
- Checking and preventing connections between pins with different pin numbers

Keeping bundle symbol types when updating or switching cables and overbraids

Cables and overbraids with bundle symbols can only be updated or switched when the types of the bundle symbols, that are placed in the project, remain the same after updating or switching models. For instance, switching a ***Shield symbol*** to a ***Twisted pair symbol*** symbol is not possible.

This prevents, for example, shields that have been set up from being removed from the connection afterward, or twisted cables from being untwisted.
The overall project structure thus remains consistent.

References: E3-6147

Highlighting intersecting, unconnected connection lines

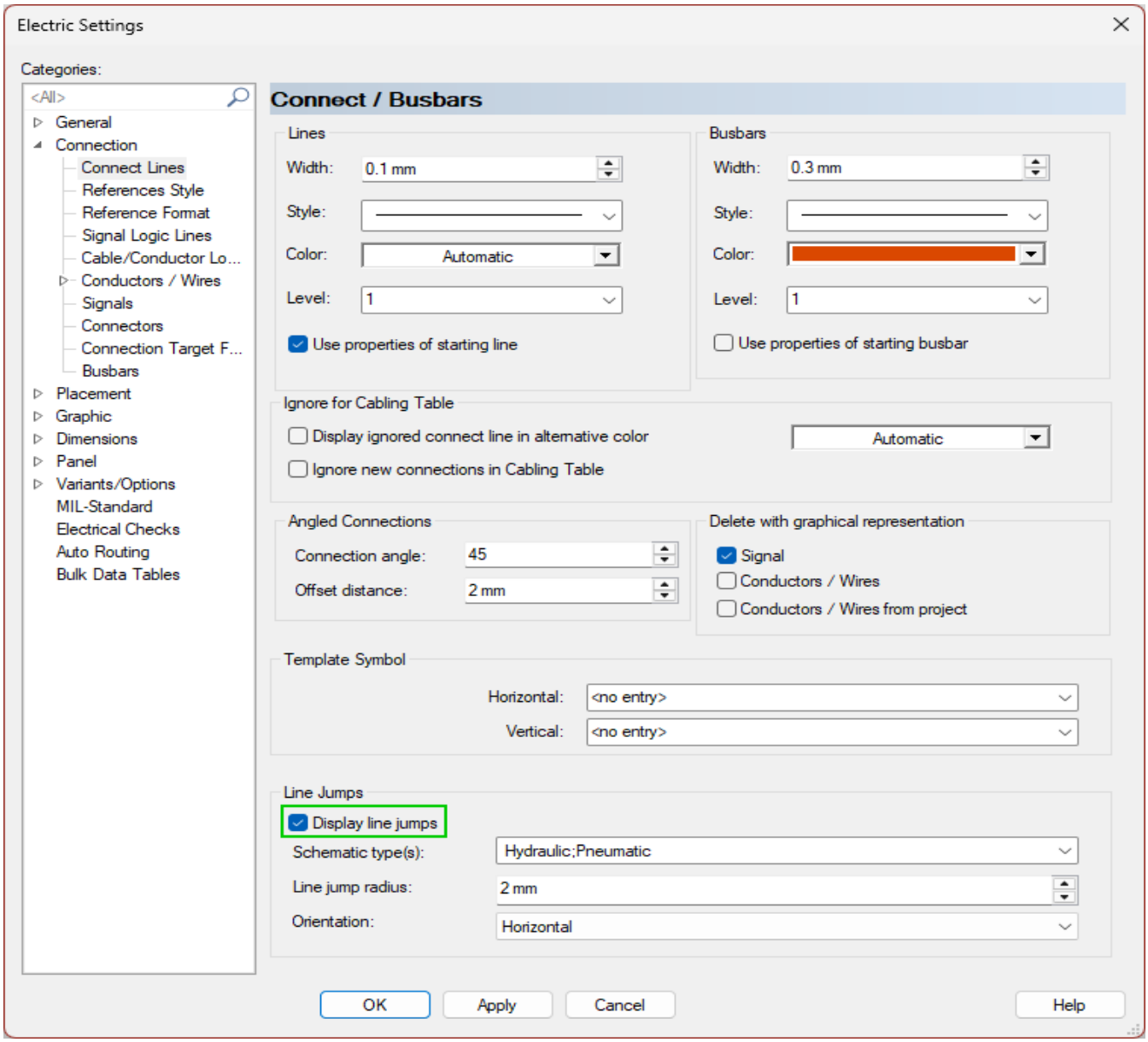
Connection lines, that intersect but are not connected, can be highlighted using a line jump.

This makes it possible, in particular, to present more complex circuit diagrams in a clearer manner or, for example, to implement company- or application-specific standards.

To highlight intersecting, unconnected connection lines, select ***Tools → Settings...*** in the main menu window.
The dialog ***Electric Settings*** opens.

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

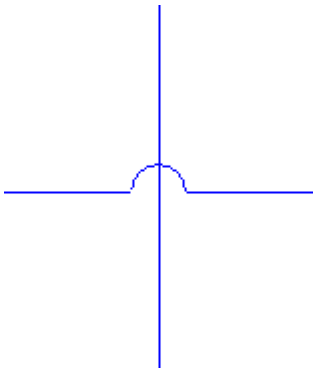
Next, activate the setting ***Display line jumps*** under ***Connection → Connect Lines***.
The setting is project specific:



Diagonal connection lines ensure that at each intersecting vertical or horizontal line a line jump is displayed.

Curved connection do not cause line jumps.

The following example shows how line jumps are displayed on vertical lines.
The representation on horizontal lines is analog:



The display of line jumps can be adjusted using the following settings:

- ***Schematic Type(s):*** This setting can be used to define on which panel types line jumps are created.

  Note: Line jumps for intersecting connections are not created on sheets if the sheet contains at least one schematic type that is not defined under ***Schematic Type(s):***. Furthermore, line jumps are only displayed on schematic sheets.

- ***Line jump radius:*** This setting defines the radius of line jumps.

  By default the value is half of that defined under ***Fluid Settings → General → Grid size:***

- ***Orientation:*** The setting defines whether a line jump is displayed on horizontal or vertical lines.

References: E3-2876

Checking and preventing connections between pins with different pin numbers

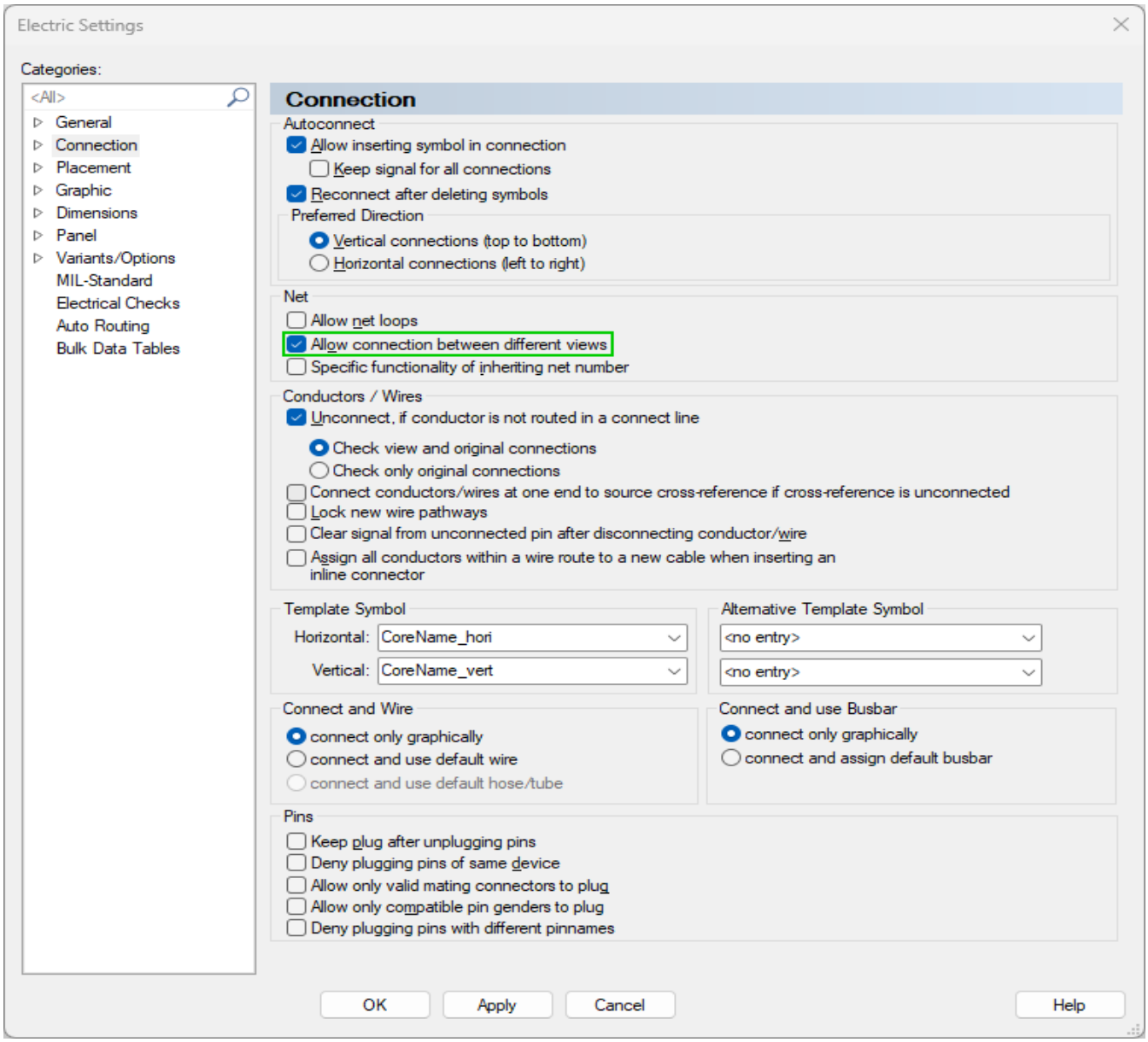
Connections between pins with different view numbers are allowed by default.
With a setting it is possible to prevent connections between views with different view numbers.
In addition, there is a check that checks nets for differing view numbers.

This makes it possible, for example, to prevent accidental, incorrect connections in the project of views with a predefined use.

To prevent views with different view numbers from being connected, select ***Tools → Settings...*** in the main menu window.
The dialog ***Electric Settings*** opens.

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

Next, deactivate the setting ***Allow connection between different views*** under ***Connection***.
The setting is project specific:



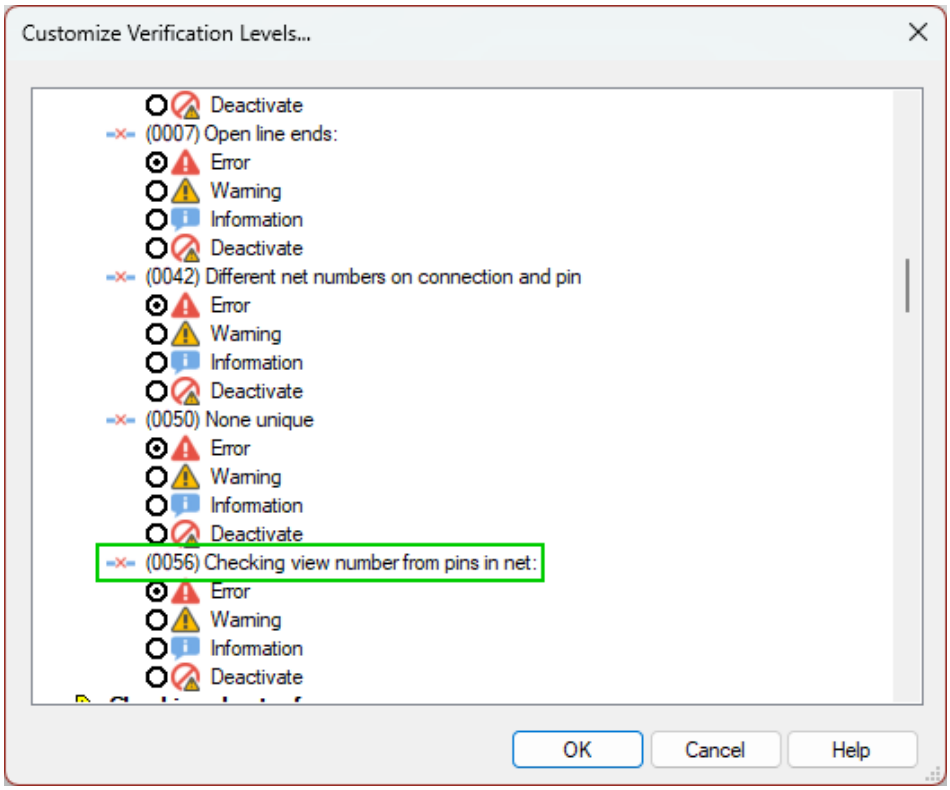
Changing this setting only influences connections that are created subsequently. Preexisting connections are not changed by the setting.

The following rules apply, if the setting is inactive:

- Views with different view numbers can no longer be connected.
- References between different view numbers can no longer be connected.
- Objects can only be copied or cut and pasted, when the then automatically defined view numbers are identical.
- Subcircuits can be read even if they contain connected views with different view numbers.
- Nets, that contain connection between views with different view numbers and that were created before the setting was deactivated, cannot be edited while the setting is inactive.
- The view numbers can be changed in the device setting even if the setting is active.

To edit the check for differing view numbers in nets, select ***Tools → Settings...*** in the main menu window.
The dialog ***Electric Settings*** opens.

Next, click***Customize Levels...*** under ***General → Verify*** .
The dialog ***Customize Verification Levels...*** opens:



Then, set the desired check behavior for the check ***(0056) Checking view number from pins in net:***.

References: E3-6642

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-89231 Neu-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

All trademarks mentioned are the property of their respective owners, (c) Zuken E3 GmbH. All rights