

WITZENMANN Flexperte PDMS & E3D Plugin

4.8

Installation and User Manual

November 28, 2025

Contents

List of Figures	7
List of Tables	9
Preface	11
0 Development history	13
0.1 Version annotations ^{4.3.0}	13
0.1.1 Out-dated features ^{†4.4.0}	13
0.2 Release notes	13
v4.2.0 Grouped supports	13
v4.2.1 Load table for grouped supports	13
v4.3.0 Excel export for parts list files	14
v4.3.1 Project-specific environment variables in start scripts	14
v4.3.2 Restraint UDAs in load tables	14
v4.4.0 Logging	14
v4.4.1 Bug fixes	14
v4.4.2 Units in PDMS 12.1	15
v4.4.3 Units for steelwork elements in PDMS 12.1	15
v4.4.4 Minor restructuring	15
v4.4.5 Selecting database elements in PDMS 12.1	15
v4.4.6 Internal release	15
v4.4.7 Mass of steelwork elements	15
v4.4.8 Bug fixes	15
v4.5.0 Extended FIN files	15
v4.5.1 New file extension for extended FIN files	15
v4.5.2 Old file extension for extended FIN files	15
v4.5.3 Warnings in PDMS 12.1.SP4	15
v4.5.4 Secondary steel elements in FIN files	16
v4.5.5 Bug fixes	16
v4.5.6 LVS bearing and LAW lift lock	16
v4.6.0 E3D integration	16
v4.6.1 LXF bearings	16
v4.7.0 E3D 2.1 integration, End of support for PDMS 11.6	16
v4.7.1 Bug fixes for drawing production in E3D 2.1	16
v4.7.2 Internal maintenance release	16
v4.7.3 Bug fixes for drawing production and FIN export in E3D 2.1	16
v4.7.4 Catalogue Extension	16
v4.7.5 Catalog changes	16
v4.8.0 Unified Engineering Compatibility	17

1	Installation	19
1.1	Requirements	19
1.2	Scope of delivery	19
1.3	Installing the database	19
1.3.1	Characteristics with ALSTOM projects	20
1.4	Installing the plugin files with PDMS	20
1.5	Installing the plugin files with E3D ^{4.6.0}	20
1.5.1	Pitfalls	21
1.5.2	Standard Installation: E3D 1.1 with a Single Plugin	21
1.5.3	Advanced Installation: E3D 1.1 with Multiple Plugins	22
2	Configuration	25
2.1	Additional environment variables	25
2.1.1	Multiple projects in a single start script ^{4.3.1}	27
2.2	The wm.ini file	30
2.2.1	General options	30
2.2.2	Design options	30
2.2.3	Draft options	31
2.2.4	Logging options ^{4.4.0}	32
2.3	Checking and editing the configuration	36
2.3.1	Editing the options	36
2.3.2	Resetting the options	37
3	Design	39
3.1	Exporting to Flexperte	39
3.1.1	Requirements	40
3.1.2	Adding attachments	40
3.1.3	Removing attachments	41
3.1.4	Choosing an axes system	41
3.1.5	Setting the installation height	41
3.1.6	Exporting the data	42
3.2	Importing from Flexperte	43
3.2.1	Requirements	43
3.2.2	Adding files	44
3.2.3	Grouping restraints ^{4.2.0}	44
3.2.4	Removing files	45
3.2.5	Creating the support constructions	45
3.2.6	Postprocessing	46
3.3	Rotating constructions	48
3.3.1	Requirements	48
3.3.2	Adding restraints	48
3.3.3	Removing restraints	49
3.3.4	The rotation	49
3.4	Parts list generation ^{4.3.0}	51
3.4.1	Requirements	51
3.4.2	Adding elements	51
3.4.3	Removing elements	52
3.4.4	Options	52
3.4.5	Creating the parts list	53
3.5	Comparing modifications	54
3.5.1	Requirements	54
3.5.2	Adding constructions	54
3.5.3	Removing constructions	55
3.5.4	Creating delta files ^{4.3.0}	55

3.5.5	Interpreting the results	55
4	Draft	57
4.1	Drawing production	57
4.1.1	Requirements	58
4.1.2	Choosing attachments	58
4.1.3	Choosing elements for a grouped drawing ^{4.2.0}	59
4.1.4	Draw list, backing sheet and views	60
4.1.5	Additional information	61
4.1.6	Creating the drawings	64
4.2	Backing sheet configuration	65
4.2.1	Requirements	65
4.2.2	Create and open	66
4.2.3	Defining the areas	66
4.2.4	Minimum and maximum view count	67
4.2.5	Editing style definitions	67
4.2.6	Import	69
4.2.7	Saving	69
5	General tools	71
5.1	The database browser	71
5.1.1	Navigation and selecting elements	71
Appendix		73
A	Combined E3D 1.1 Customization File	73
B	Application codes in PDMS	74
B.1	Design	74
B.2	Draft	75
C	Axes systems	76
C.1	Inner structure of an axes system	76
C.2	Determinig the closest axes	76
D	UDAs requiried for data interchange	77
E	Load table layout	79
E.1	The load table's body	79
F	Layout of Backing Sheet Configuration Files	81
F.1	XML Schema	81
F.2	Style definitions in a BCF	84

List of Figures

1.1	Changes in <i>evvars.bat</i>	20
1.2	Sample directory structure for two independent E3D 1.1 plugins	23
2.1	Environment variables without HANGER	26
2.2	Environment variables with HANGER	26
2.3	Environment variables with HANGER and WMIMPORT	27
2.4	Multiple projects in a single start script	28
2.5	Log handler definition syntax	33
2.6	Example output of a screen log handler	34
2.7	Example usage of the log handlers and their options	34
2.8	The WITZENMANN Configurator	36
3.1	Menu and toolbar in the Design module	39
3.2	The WITZENMANN Export Tool	40
3.3	Contents of the status bar during selection mode	41
3.4	Setting the elevation in the 3D model	42
3.5	The WITZENMANN Import Tool	43
3.6	Grouping of selected attachments	45
3.7	Successfully created support constructions	46
3.8	Wrongly aligned steel clamp	47
3.9	The WITZENMANN Rotate Tool	48
3.10	An entirely highlighted support construction	49
3.11	Rotating a restraint's upper parts	50
3.12	Rotation completed	50
3.13	The WITZENMANN Parts List Tool	51
3.14	The WITZENMANN Delta Tool	54
3.15	Output created by the Delta Tool for a changed restraint	55
4.1	The WITZENMANN Drawing Tool	57
4.2	Changing the registry of some support points	58
4.3	Chossing elements for a grouped drawing	59
4.4	Page 2 of the WITZENMANN Drawing Tool	60
4.5	The Backing Sheet Selector	61
4.6	Page 3 of the WITZENMANN Drawing Tool	62
4.7	Key plan of a sample drawing	64
4.8	The WITZENMANN Backing Sheet Configuration Tool	65
4.9	Area definitions in a sample BCF	66
4.10	Backing sheet with three views	67
4.11	Editing a BCF's style definitions	68
4.12	Appearance of the default styles	69
5.1	The database browser	71

A.1	Combined <i>DesignCustomization.xml</i> for two plugins in E3D 1.1	73
C.1	Hierarchy of the test project's axes system	76
F.1	Sample Backing Sheet Configuration File	82
F.2	XML Schema for BCFs	83

List of Tables

1.1	Databases contained in the WZM project	19
1.2	Test databases in the WZM project	20
2.1	Additional environment variables	25
2.2	Values of the environment variables without HANGER	26
2.3	Values of the environment variables with HANGER	27
2.4	Values of the environment variables while HANGER and WMIMPORT are set	27
2.5	Environment variables for the ABC project	28
2.6	Environment variables for the DEF project	29
2.7	Environment variables for the XYZ project	29
B.1	PDMS application codes in the Design module	74
B.2	PDMS application codes in the Draft module	75
D.1	UDAs required on attachment level	77
D.2	UDAs required on pipe level	78
D.3	UDAs required on restraint level	78
E.1	The load table's body	79
F.1	Supported style properties in the keyplan element	84
F.2	Supported style properties in the loadtable element	85
F.3	Supported style properties in the partslist element	85
F.4	Supported style properties in the view element	86

Preface

The plugin described in this manual extends the functionality of the WITZENMANN Flexperte software and allows the user to import the calculated supports into a PDMS project, which requires the WITZENMANN catalogue databases.

The plugin provides a tool for the exchange of attachment data between PDMS and Flexperte using Flexperte's FIN file format.

Drawing production was designed for WITZENMANN supports which were created and modified by this plugin only. Therefore, there is no guarantee that the drawing production works correctly for supports of other types.

Every tool of the WITZENMANN Flexperte PDMS Plugin mainly operates on attachment data and is bound to the following naming convention when gathering additional information:

$$\begin{aligned}\text{<restraint name>} &= \text{<attachment name>} + \text{"RE"} \\ \text{<secondary steel name>} &= \text{<attachment name>} + \text{"S"}\end{aligned}$$

In a restraint group the plugin will not search for secondary steel for every restraint. Instead, the following naming convention applies:

$$\text{<secondary steel name>} = \text{<group name>} + \text{"S"}$$

How to organize restraints into groups will be described in section 3.2.3.

The behaviour of the plugin can be configured in many different ways. Read the appropriate sections for further information.

Important information

Please note that it is not recommended to have characters like \, /, :, *, ?, ", <, > and | in your attachments' and restraints' names (especially for those elements which are important for data interchange), since Windows forbids them in filenames. Although the plugin can handle those characters and replaces them by underscores (_), elements having one or more of these characters in their name may decrease performance and can lead to data loss. For example, the elements /AB/BA and AB\BA would have the same filenames for their export files, namely AB_BA.fin, as a result of which one of the files could be overwritten unintendedly.

Disclaimer

Ingenieurbüro Werk GmbH has a policy of continuing product development: therefore, the information contained in this document may be subject to change without notice.

Ingenieurbüro Werk GmbH makes no warranty of any kind with regard to this document, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

While every effort has been made to verify the accuracy of this document, Ingenieurbüro Werk GmbH shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance or use of this material.

Chapter 0

Development history

0.1 Version annotations^{4.3.0}

If a particular section became valid with a particular plugin version, it will be marked with the appropriate version string (i.e. 4.3.0 for this section).

0.1.1 Out-dated features^{†4.4.0}

No longer supported features will be marked with a dagger (†) and the number of the version that abandoned support for this feature (i.e. 4.4.0 for this section).

0.2 Release notes

v4.2.0 Grouped supports

03/18/2010

The import tool can now combine multiple supports into a group and the drawing tool can create a single drawing of a group.

The drawing tool now also adds secondary steel elements and the primitives BOX and CYLI to the parts list and creates labels for them. BOX and CYLI elements are referred to as plate and round profile respectively.

Also, it is now possible to create drawings with PDMS 12.0 and above. Backing Sheet Configuration Files that have been created with PDMS 11.6 can also be used in PDMS 12.0 and vice versa.

v4.2.1 Load table for grouped supports

04/23/2010

The drawing tool can now create load tables on grouped drawings, which required changing the load table format (see appendix E). The `<style>` block in the `<loadtable>` element of the Backing Sheet Configuration Files now also supports a “`vertical-align`” property, which controls whether the load table should be aligned to the top or the bottom of its assigned area.

The load table now also contains the alstom-specific “POW” User Defined Attributes (UDA), which the export tool also writes to the FIN files.

Additionally, the internal format of the STK files changed. Negative values are now represented as a single value, e.g. “-3”, instead of the unclear “0 `<absolute value>`” format, e.g. “0 3”. This allows the STK files to contain arrays.

09/06/2010 v4.3.0 Excel export for parts list files

Parts lists can now be generated directly from the PDMS Design module. The parts list tool creates files in the Microsoft Spreadsheet XML Format, which is supported by Microsoft Excel 2003 and above as well as OpenOffice.org.

The delta tool also writes Excel files in the aforementioned XML format instead of the previous text format.

The drawing tool can now automatically extend the display volume to include the secondary steel elements of a support or support group. Also, it can create angular measurements for sloped pipes.

Additionally, the `<style>` block in the `<view>` element of the Backing Sheet Configuration Files now recognizes a `"dim-font-size"` property. This controls the dimensioning text font size independently of the label font size.

09/21/2010 v4.3.1 Project-specific environment variables in start scripts

It is now possible to start multiple PDMS projects with a single script file still provide each of them with individual environment variables. Thus, we are no longer restricted to use the same destinations for STK, FIN, and delta files in different projects. See also section [2.1.1](#).

11/25/2010 v4.3.2 Restraint UDAs in load tables

Restraint elements now have additional User Defined Attributes (UDAs), which simplify data access during import, export, and drawing production. In contrast to the `"UST"` attributes, the alstom-specific `"POW"` attributes could not hold forces and movements at restraint level. This caused the the load table to use the corresponding values from the attachment level instead, making the attachment data more significant than the (missing) restraint data. With the new `"WZM"` attributes that ship with the plugin databases, the import tool can now store all necessary data that it received from Flexperte on the restraint regardless of the project context. Appendix [E](#) shows the current load table format.

12/27/2011 v4.4.0 Logging

Wm.ini now has a logging section, which allows the definition of logging rules for the plugin's five major components (export, import, delta generation, parts list generation, and drawing production). Messages may be printed to text files or the PDMS console or may be presented to the user in a message box.

Users can now specify in which views the drawing tool should place dimensioning, parts list numbers, and other labels. Also, the viewing directions of generated views can be set more freely and views may have user-defined captions.

Moreover, nominal bores can be either printed in millimetres (DN) or inches (NPS).

You can now specify a custom north arrow in `wm.ini`.

All elevations in chain dimensions are printed with respect to the key plan's axes system. If a double load chain has to identical legs, only one of them will be dimensioned and labelled.

The load table now also contains values for spring rate, cold load, and skewed pull.

10/26/2012 v4.4.1 Bug fixes

Fixed some minor bugs during STK import.

The import tool now displays the restraints it created in the 3D view.

The plugin no longer relies on the `wm.ini` being referenced in the `pml.index` file. Thus, the plugin continues to work even if somebody issues the `"pml rehash all"` command. See also section [1.5.1](#).

v4.4.2 Units in PDMS 12.1 01/04/2013

In PDMS 12.1, the export tool removes units from all values, ensuring the same format for FIN files in all PDMS versions.

v4.4.3 Units for steelwork elements in PDMS 12.1 01/11/2013

Removed units for lengths of SCTN elements in FIN files.

v4.4.4 Minor restructuring 02/13/2013

Deleted some unused files.

v4.4.5 Selecting database elements in PDMS 12.1 02/24/2014

In PDMS 12.1, the database browser used to display only elements from the first MDB that has been registered with the project. It now displays elements from all MDBs.

v4.4.6 Internal release 06/06/2014

This is an internal release as a preliminary to [v4.4.7](#).

v4.4.7 Mass of steelwork elements 06/06/2014

The parts list generated in the PDMS Design and Draft modules now determine the mass of steelwork elements by examining the “:UstWeight” UDA of the element’s SpReference.

v4.4.8 Bug fixes 08/07/2014

The import tool now ignores certain empty values in the STK file. For example, the import used to fail on empty “ANGLE_X” and “ANGLE_Y” directives.

Also, in PDMS 12.0 Draft, the database browser now displays SITE elements from all MDBs.

v4.5.0 Extended FIN files 10/01/2014

The FIN files now also contain information about the restraint that is associated with a particular attachment. Thus, FIN files may be used to synchronize a Flexperte project file with the current state of the PDMS Design model.

The export may be started as usual with the export tool from the PDMS Design module. Additionally, there is a new option to start the export process during drawing production.

v4.5.1 New file extension for extended FIN files 10/16/2014

Extended FIN files now use the “efin” extension. Also, FIN files created during drawing production are stored in %WMEXPORT%\efin. FIN files created in the PDMS Design module are still stored in %WMEXPORT%.

v4.5.2 Old file extension for extended FIN files 10/24/2014

Extended FIN files now use the old “fin” extension again. FIN files are still stored in different directories depending on the PDMS module. See [v4.5.1](#).

v4.5.3 Warnings in PDMS 12.1.SP4 02/09/2015

Some forms issued warnings when they were opened in PDMS 12.1.SP4. This release fixes these warnings.

- 03/03/2015 **v4.5.4 Secondary steel elements in FIN files**
The “SECONDARY_STEEL-DATA” block of a FIN file now contains all secondary steel elements that have a valid SpReference. The block used to contain only SCTN, BOX, and CYLI elements.
- 03/16/2015 **v4.5.5 Bug fixes**
Parts lists generated in the PDMS Design and Draft modules now ignore SCTN elements with invalid SpReference or CatReference attributes.
- 03/17/2015 **v4.5.6 LVS bearing and LAW lift lock**
The import tool now recognizes LVS bearings and LAW lift locks.
- 04/20/2015 **v4.6.0 E3D integration**
The plugin now adds a WITZENMANN tab to the ribbon bar of the E3D modules Model and Draw. See section [1.5](#) for installation details.
- 11/30/2015 **v4.6.1 LXF bearings**
The import tool now recognizes LXF bearings.
- 03/23/2016 **v4.7.0 E3D 2.1 integration, End of support for PDMS 11.6**
The plugin now adds a WITZENMANN tab to the ribbon bar of the E3D 2.1 modules Model and Draw. See section [1.5](#) for installation details.
Also, the plugin is no longer supported with PDMS 11.6.
- 11/04/2016 **v4.7.1 Bug fixes for drawing production in E3D 2.1**
E3D 2.1.0.2 slightly changed the behavior of LDIM elements. When opening a plugin generated drawing, E3D initially only displayed a single projection line of one of the linear dimensions. The whole drawing was scaled down to a single pixel and thus effectively not visible.
This release fixes that problem.
- 01/26/2017 **v4.7.2 Internal maintenance release**
This is an internal release as a preliminary to [v4.7.3](#).
- 01/30/2017 **v4.7.3 Bug fixes for drawing production and FIN export in E3D 2.1**
The plugin now treats GENSEC and SCTN elements the same way. During drawing production the plugin now also adds the length and weight of GENSEC elements to the drawing’s parts list.
Additionally, the FIN files should now contain all relevant design data again. For example, E3D sometimes returns unset values when evaluating the :USTDTEMP attribute even though it actually has a value. This release works around E3D’s misbehavior and evaluates the UDAs correctly.
- 08/03/2018 **v4.7.4 Catalogue Extension**
The catalogue now includes sliding and fixed supports in NB 225.
- 03/23/2023 **v4.7.5 Catalog changes**
Some catalog parts are renamed to avoid naming conflicts.

v4.8.0 Unified Engineering Compatibility

11/28/2025

The plugin now supports Unified Engineering.

Chapter 1

Installation

1.1 Requirements

The WITZENMANN Flexperte PDMS & E3D Plugin requires Aveva PDMS 12.0.SP6 or higher or Aveva Everything 3D 1.1 or higher.

1.2 Scope of delivery

Among this documentation, the plugin contains files that manage the integration with PDMS and E3D, the actual forms, functions, objects and configuration files as well as a small number of Backing Sheet Configuration Files, which are used in the Draft module.

Furthermore, the plugin contains the databases, which provide the particular WITZENMANN components. In contrast to the macro package, there are different database files for PDMS 12.0 and PDMS 12.1.¹

1.3 Installing the database

In order to allow you to use the WITZENMANN components in your projects, a PDMS project named WZM is provided, whose databases have to be embedded as foreign databases in the appropriate MDBs of the particular PDMS project. Table 1.1 lists the included databases.

Table 1.1 – Databases contained in the WZM project

Database name	Type	#	File name	Access
MASTER/WITZPADD	PADD	4620	%WZM000%\wzm4620_0001	Update
MASTER/WITZDICT	DICT	4621	%WZM000%\wzm4621_0001	Update
MASTER/WITZCATA	CATA	6992	%WZM000%\wzm6992_0001	Multiwrite Im.
MASTER/H&S_DICT	DICT	6994	%WZM000%\wzm6994_0001	Update

Additionally, two test databases, one for Design and Draft each, complete the package, but should be embedded into a project for testing purposes only.

¹The database files for PDMS 12.1 can also be used with E3D.

Table 1.2 – Test databases in the WZM project

Database name	Type	#	File name	Access
MASTER/WITZPADD-TEST	PADD	4628	%WZM000%\wzm4628_0001	Update
MASTER/WITZDESI-TEST	DESI	4629	%WZM000%\wzm4629_0001	Update

1.3.1 Characteristics with ALSTOM projects

In ALSTOM projects, the catalogue and dictionary databases (number 6992 and 6994 respectively) may be already furnished externally. Hence, there will be no need to embed them.

1.4 Installing the plugin files with PDMS

To enable the plugin in PDMS, the environment variables PMLLIB and PDMSUI must be extended. Usually they are set in the file *evars.bat*.

Example

Let the folders *pmllib* and *pdmsui* be located in D:\WITZENMANN. This requires the following changes in *evars.bat*:

Figure 1.1 – Changes in *evars.bat*

```

1  if not "%PMLLIB%"==" goto pmlok
2  set  PMLLIB=D:\WITZENMANN\pmllib %1\pmllib
3  echo PMLLIB set to %PMLLIB%
4  :pmlok
5
6  if not "%PDMSUI%"==" goto uiok
7  set  PDMSUI=D:\WITZENMANN\pdmsui %1\pdmsui
8  echo PDMSUI set to %PDMSUI%
9  :uiok

```

The extension of PMLLIB and PDMSUI happens in the lines 2 and 7 respectively. In both cases %1 is replaced by the PDMS installation directory.

After a restart of PDMS, the plugin should be available in the Design and Draft modules. At least the WITZENMANN menu should be visible.

1.5 Installing the plugin files with E3D^{4.6.0}

Compared to PDMS, the installation of the Flexperte Plugin with E3D is rather involved. Also, the installation procedures differ slightly between E3D 1.1 and E3D 2.1.

With E3D 1.1, if the Flexperte Plugin is the only plugin you are planning to use with E3D, you only need to change three environment variables (see section 1.5.2). However, if you are using multiple plugins from different vendors with E3D, you will have to create your own user interface customization files, which is described in section 1.5.3.

With E3D 2.1, you always have to change the same three environment variables as with E3D 1.1; regardless of the number of plugins you are using with E3D (see section 1.5.3).

1.5.1 Pitfalls

With Everything 3D (E3D) AVEVA made some changes to its start scripts, which makes installing plugins with E3D somewhat different than with PDMS. Also, AVEVA introduced some new file types with E3D, which can lead to problems when you use plugins with both E3D and PDMS.

The following sections describe some common pitfalls that you should be aware of during the installation of the Flexperte Plugin with E3D.

Renamed Environment Variables

Like PDMS, the E3D installation directory contains an *evars.bat* file, which sets environment variables that are required for the correct operation of E3D and its plugins. Comparing this *evars.bat* file to the one from PDMS, you may notice that AVEVA renamed some of those environment variables. Most importantly, AVEVA renamed the variable “PDMSUI” to “PMLUI”, which most plugins in general and the Flexperte Plugin in particular require for their correct operation.

Multiple Paths in Environment Variables

Starting with E3D, AVEVA now allows spaces in directory names of search paths. That means, however, that you can’t use spaces to separate multiple paths in a search path. Instead, you should now use semicolon characters (;) to separate multiple paths (like in the Windows “PATH” environment variable, for example). Keep in mind to replace spaces with semicolons when you migrate old PDMS start scripts to E3D.

```
1 rem PDMS-style search path
2 set PDMSUI=%PDMSUI% E:\plugins\first E:\plugins\second
3
4 rem E3D-style search path
5 set PMLUI=%PMLUI%;E:\plugins\first;E:\plugins\second
```

New File Types: “pml rehash all” Considered Evil

The plugin’s program files are stored in a PMLLIB folder, which also contains a *pml.index* file. This file references the plugin’s actual program files in the PMLLIB folder. The “pml rehash all” command scans the PMLLIB folder and rewrites the *pml.index* file. Unfortunately, far too many installation manuals still consider this command as a vital installation step and far too many standard users execute it on a regular basis.

E3D introduces a new file type, the PMLCMD files, which control the integration of E3D plugins like the Flexperte Plugin into the E3D graphical user interface. Like the other program files, the PMLCMD files are stored in the PMLLIB folder and are thus referenced in the *pml.index* file that ships with the Flexperte Plugin. Since PDMS doesn’t know how it should handle PMLCMD files, it simply ignores them and their entries in the *pml.index* file. However, it also ignores them when you or another PDMS user execute the “pml rehash all” command. This leads to a rewritten *pml.index* file without any PMLCMD entries in it. When you now try to use the Flexperte Plugin in E3D, some menu entries in the WITZENMANN tab may be missing because of the rewritten *pml.index* file.

If you can’t keep your users from executing the “pml rehash all” command, you should consider setting the read-only attribute on the plugin’s *pml.index* file. This way, PDMS and E3D won’t make any changes to the *pml.index* file that ships with the plugin.

1.5.2 Standard Installation: E3D 1.1 with a Single Plugin

Similar to the PDMS installation procedure, you have to change the values of some environment variables to use the Flexperte Plugin in E3D. You can either do this in the *evars.bat* file in the E3D installation directory or in your custom start scripts, which may be accessible on a

network share. The relevant environment variables are “PMLLIB”, “PMLUI” (previously “PDMSUI”), and “CAF_UIC_PATH”.²

Suppose you unpacked the Flexperte Plugin to `E:\e3d-plugins\witzenmann` with the directories `pmllib`, `pdmsui`, and `uic` directly below this path. Then you usually only need to set the aforementioned environment variables as follows:

```
1 set PMLLIB=%PMLLIB%;E:\e3d-plugins\witzenmann\pmllib
2 set PMLUI=%PMLUI%;E:\e3d-plugins\witzenmann\pdmsui
3 set CAF_UIC_PATH=E:\e3d-plugins\witzenmann\uic\E3D_1.1
```

Note that we did not include the old value of the “CAF_UIC_PATH” when resetting it, so the variable only contains “E:\e3d-plugins\witzenmann\uic\E3D_1.1”. If you are using an E3D standard installation without any additional plugins, the three lines above cover the entire installation procedure. However, if you want to use multiple plugins that extend the E3D user interface, you have to make further adjustments.

1.5.3 Advanced Installation: E3D 1.1 with Multiple Plugins

If you want to use more than one E3D plugin that extends the E3D user interface, the three aforementioned changes to the *evars.bat* won’t suffice. You can, of course, simply extend the “PMLLIB” and “PMLUI” environment variables to refer to two, three, or more plugins. However, the “CAF_UIC_PATH” variable can’t handle multiple plugins very well.

Modifications to the E3D user interface are controlled by module-specific XML file named “<ModuleName>Customization.xml”.³ Usually, when an E3D module starts, it searches its installation directory for those files. However, if you set the “CAF_UIC_PATH” variable in your start script, E3D first looks in the directory specified by that variable. Only if it can’t find the required file in that directory, it reverts to loading the file from its installation directory.

Although “CAF_UIC_PATH” can hold multiple paths like “PMLLIB” and “PMLUI”, E3D 1.1 only loads the *first* “<ModuleName>Customization.xml” in its search path during module startup.⁴ Thus, you have to create a combined customization file that references the plugins you want to use. And you have to do that *for every module* that uses multiple plugins.

Sample Configuration

Suppose you unpacked the Flexperte Plugin to `E:\e3d-plugins\witzenmann` with the directories `pmllib`, `pdmsui`, and `uic` directly below this path. Suppose further that you unpacked an additional third party plugin to `E:\e3d-plugins\other`, which also contains a `uic` folder with a UIC file in it. As shown in figure 1.2, you could now create a common folder that contains a combined *DesignCustomization.xml*.

Then, you extend the “PMLLIB” and “PMLUI” environment variables as usual and point the “CAF_UIC_PATH” variable to the `E:\e3d-plugins\common` folder you created in the previous step. (I only split the assignments to “PMLLIB” and “PMLUI” across multiple lines to improve readability. You can, of course, assign both paths on a single line in your start scripts.)

```
1 set PMLLIB=%PMLLIB%;E:\e3d-plugins\witzenmann\pmllib
2 set PMLLIB=%PMLLIB%;E:\e3d-plugins\other\pmllib
3 set PMLUI=%PMLUI%;E:\e3d-plugins\witzenmann\pdmsui
4 set PMLUI=%PMLUI%;E:\e3d-plugins\other\pmlui
5 set CAF_UIC_PATH=E:\e3d-plugins\common
```

Finally, you have to create a combined *DesignCustomization.xml* file that references both the E3D UIC files as well as the UIC files of the two plugins, as shown in figure A.1. Note that the `Path`

²CAF stands for “Common Application Framework”, which provides an interface for E3D extensions like the Flexperte Plugin. UIC stands for “User Interface Customization”. Thus, the “CAF_UIC_PATH” points to a folder that contains modifications to the E3D user interface.

³The E3D module “Model” uses the name “Design” instead.

⁴E3D 2.1 loads all “<ModuleName>Customization.xml” files in its search path, so this setup isn’t necessary for E3D 2.1.

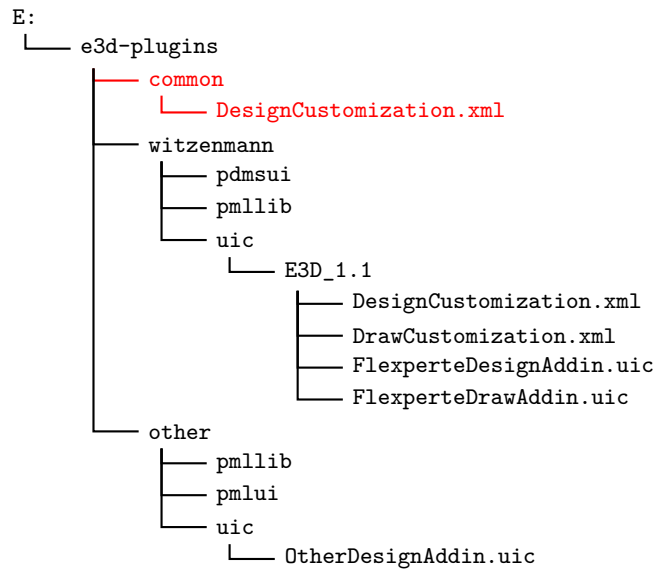


Figure 1.2 – Sample directory structure for two independent E3D 1.1 plugins. The plugins provide the `witzenmann` and `other` folders. You have to create the `common` folder and its `DesignCustomization.xml` file by yourself.

attributes can be paths relative to both the E3D installation directory and the “CAF_UIC_PATH”. That is, you can refer to UIC files using either relative paths as shown in lines 19 and 20 of figure A.1 or absolute paths like `E:\e3d-plugins\witzenmann\uic\FlexperteDesignAddin.uic`.

Standard Installation: E3D 2.1

As with E3D 1.1 you have to change the values of the “PMLLIB”, “PMLUI” (previously “PDMSUI”), and “CAF_UIC_PATH” environment variables to use the Flexperte Plugin in E3D 2.1. You can either do this in the `evvars.bat` file in your E3D installation directory, in the `projects.bat` file in the E3D 2.1 projects directory, or in your custom start scripts, which may be accessible on a network share.

You can set the “PMLLIB” and “PMLUI” environment variables in the same way as with E3D 1.1. Since version 2.1 E3D supports loading multiple “<ModuleName>Customization.xml” files from multiple directories in the “CAF_UIC_PATH” search path. However, if you want to have the WITZENMANN tab appended to the end of the ribbon menu rather than to its front, you have to explicitly include the E3D installation directory in the “CAF_UIC_PATH” variable. Otherwise E3D will pick up its own configuration files last.

Suppose you unpacked the Flexperte Plugin to `E:\e3d-plugins\witzenmann` with the directories `pmlib`, `pdmsui`, and `uic` directly below this path. Then you usually only need to set the aforementioned environment variables as follows:

```

1 rem This should go to the top of your projects.bat file otherwise
2 rem E3D will show the "WITZENMANN" tab as the first tab in the
3 rem ribbon menu; even before the "HOME" and "TOOLS" tabs.
4 set CAF_UIC_PATH=%AVEVA_DESIGN_INSTALLED_DIR%
5 rem ...
6 rem more projects and other plugin settings...
7 rem ...
8 set PMLLIB=%PMLLIB%;E:\e3d-plugins\witzenmann\pmlib
9 set PMLUI=%PMLUI%;E:\e3d-plugins\witzenmann\pdmsui
10 set CAF_UIC_PATH=%CAF_UIC_PATH%;E:\e3d-plugins\witzenmann\uic\E3D_2.1

```

Note that the “CAF_UIC_PATH” variable now contains its old value from line 4, the path to the WITZENMANN UIC files in line 10, and any paths that other plugins may add to it.

Chapter 2

Configuration

2.1 Additional environment variables

In order to simplify the plugin's usage, we recommend you to set some environment variables in addition to those mentioned in the previous section. The former, however, are project-specific and hence need to be set in project-related start routines.

Table 2.1 shows the names and meanings of the specific environment variables. It is necessary that the directories the variables point to exist.

Table 2.1 – Additional environment variables

Name	Description
WMIMPORT	Provides the directory the Flexperte generated STK files will be searched in.
WMOLDIMPORT	Provides the directory processed STK files will be stored in. If this variable is not set, the path from WMIMPORT with an additional \old will be used.
WMEXPORT	In this directory the FIN files, which contain attachment data for Flexperte, will be stored.
WMDELTA	In this directory the files for delta check will be stored.
HANGER	This variable was kept for compatibility to older projects. If it is set instead of the other four variables, the four other paths will be generated by the concatenation of the value of HANGER and a small suffix for the particular path (\import, \import\old, \export and \delta respectively).

Example

In a sample project “ABC” with the project directory `D:\pdms\abc`, the variables could be set like this:

Figure 2.1 – Environment variables without HANGER

```

1  set PROJECTPATH=D:\pdms\abc
2  set WMIMPORT=%PROJECTPATH%\wmsupport\import
3  set WMOLDIMPORT=%WMIMPORT%\old
4  set WMEXPORT=%PROJECTPATH%\wmsupport\export
5  set WMDELTA=%PROJECTPATH%\wmsupport\delta

```

Table 2.2 shows the resulting values of the environment variables. As you can see, every variable has its expected value. Please note, however, that the variable `PROJECTPATH` was already substituted by its value when it was assigned to the other variables. The subsequent examples make use of this technique in order to restore as much information as possible from a given subset of these variables.

Table 2.2 – Values of the environment variables without HANGER

Variable	Value
WMIMPORT	D:\pdms\abc\wmsupport\import
WMOLDIMPORT	D:\pdms\abc\wmsupport\import\old
WMEXPORT	D:\pdms\abc\wmsupport\export
WMDELTA	D:\pdms\abc\wmsupport\delta

Example

In a second project “DEF” with the project directory `D:\pdms\def`, only the environment variable `HANGER` is set.

The resulting values of the environment variables are listed in table 2.3. Again, the variable `PROJECTPATH` is substituted by its value during the definition of `HANGER`. Moreover, the plugin automatically generates the values for `WMIMPORT`, `WMOLDIMPORT`, `WMEXPORT` and `WMDELTA` by appending the respective suffixes to the value of `HANGER`.

Note that values written in *italics* are only accessible through the plugin, i.e. they can *not* be queried like this:

```

q evar WMIMPORT
q evar WMOLDIMPORT
q evar WMEXPORT
q evar WMDELTA

```

Figure 2.2 – Environment variables with HANGER

```

1  set PROJECTPATH=D:\pdms\def
2  set HANGER=%PROJECTPATH%\hanger

```

Table 2.3 – Values of the environment variables with HANGER

Variable	Value
HANGER	D:\pdms\def\hanger
WMIMPORT	<i>D:\pdms\def\hanger\import</i>
WMOLDIMPORT	<i>D:\pdms\def\hanger\import\old</i>
WMEXPORT	<i>D:\pdms\def\hanger\export</i>
WMDELTA	<i>D:\pdms\def\hanger\delta</i>

Example

The last sample project “XYZ” with the project directory D:\pdms\xyz, demonstrates which effects of simultaneously setting the HANGER and WMIMPORT variables. Table 2.4 shows the values of the

Figure 2.3 – Environment variables with HANGER and WMIMPORT

```
1  set PROJECTPATH=D:\pdms\xyz
2  set WMIMPORT=%PROJECTPATH%\wmsupport\import
3  set HANGER=%PROJECTPATH%\hanger
```

variables. As mentioned earlier, WMOLDIMPORT gets its value from WMIMPORT while WMEXPORT and WMDELTA are generated using HANGER. Again, note that the values written in italics are not set physically and hence can not be queried in PDMS.

Table 2.4 – Values of the environment variables while HANGER and WMIMPORT are set

Variable	Value
WMIMPORT	D:\pdms\xyz\wmsupport\import
WMOLDIMPORT	<i>D:\pdms\xyz\wmsupport\import\old</i>
HANGER	D:\pdms\xyz\hanger
WMEXPORT	<i>D:\pdms\xyz\hanger\export</i>
WMDELTA	<i>D:\pdms\xyz\hanger\delta</i>

2.1.1 Multiple projects in a single start script^{4.3.1}

If you have set up a common start script for multiple projects (as shown in figure 2.4), you can still provide each project with its individual environment variables. The plugin searches for the most specific environment variables first and continues – step by step – with the more general ones. This search continues until a given environment variable is set *and* its contents point to an existing directory.

Concerning the WMOLDIMPORT variable in the XYZ project, the plugin would try the following values in the given order:

1. %XYZWMOLDIMPORT%

Figure 2.4 – Multiple projects in a single start script

```

1  set PROJECTPATH=D:\pdms
2
3  Rem project ABC
4  set ABC000=%PROJECTPATH%\ABC\abc000
5  set ABCPIC=%PROJECTPATH%\ABC\abcpic
6  set ABCMAC=%PROJECTPATH%\ABC\abcmac
7  set ABCISO=%PROJECTPATH%\ABC\abciso
8
9  set ABCWMIMPORT=%PROJECTPATH%\ABC\wmsupport\import
10 set ABCWMOLDIMPORT=%ABCWMIMPORT%\old
11 set ABCWMEXPORT=%PROJECTPATH%\ABC\wmsupport\export
12 set ABCWMDELTA=%PROJECTPATH%\ABC\wmsupport\delta
13
14 Rem project DEF
15 set DEF000=%PROJECTPATH%\DEF\def000
16 set DEFPIC=%PROJECTPATH%\DEF\defpic
17 set DEFMAC=%PROJECTPATH%\DEF\defmac
18 set DEFISO=%PROJECTPATH%\DEF\defiso
19
20 set DEFHANGER=%PROJECTPATH%\DEF\hanger
21
22 Rem project XYZ
23 set XYZ000=%PROJECTPATH%\XYZ\xyz000
24 set XYZPIC=%PROJECTPATH%\XYZ\xyzpic
25 set XYZMAC=%PROJECTPATH%\XYZ\xyzmac
26 set XYZISO=%PROJECTPATH%\XYZ\xyziso
27
28 set XYZHANGER=%PROJECTPATH%\XYZ\hanger
29 set XYZWMIMPORT=%PROJECTPATH%\XYZ\wmsupport\import

```

Table 2.5 – Environment variables for the ABC project

Variable	Value
WMIMPORT	D:\pdms\abc\wmsupport\import
WMOLDIMPORT	D:\pdms\abc\wmsupport\import\old
WMEXPORT	D:\pdms\abc\wmsupport\export
WMDELTA	D:\pdms\abc\wmsupport\delta

2. %XYZWMIMPORT%\old
3. %XYZHANGER%\import\old
4. %WMOLDIMPORT%
5. %WMIMPORT%\old
6. %HANGER%\import\old

In our example, the lookup finished after the second try because there was an existing directory %XYZWMIMPORT%\old.

The configuration given in figure 2.4 would produce the project-specific environment variables as shown in tables 2.5 to 2.7. These match exactly the values given in the previous section, so both approaches are equally meaningful. Note that the values written in italics are calculated by the plugin and cannot be queried directly in PDMS.

Table 2.6 – Environment variables for the DEF project

Variable	Value
HANGER	D:\pdms\def\hanger
WMIMPORT	D:\pdms\def\hanger\import
WMOLDIMPORT	D:\pdms\def\hanger\import\old
WMEXPORT	D:\pdms\def\hanger\export
WMDELTA	D:\pdms\def\hanger\delta

Table 2.7 – Environment variables for the XYZ project

Variable	Wert
WMIMPORT	D:\pdms\xyz\wmsupport\import
WMOLDIMPORT	D:\pdms\xyz\wmsupport\import\old
HANGER	D:\pdms\xyz\hanger
WMEXPORT	D:\pdms\xyz\hanger\export
WMDELTA	D:\pdms\xyz\hanger\delta

2.2 The `wm.ini` file

The `wm.ini` file, which is located in the `pmlib` folder in the plugin directory, provides one more possibility to set up the plugin's behaviour. Its options are used to alter the behaviour of several tools of the plugin.

In PDMS, the values of the directives described here will be stored in a centrally accessible configuration object named `!wmConfig`. This object enables the plugin to directly reflect changes of a given value at runtime without requiring PDMS to restart. Most of the directives described in this section may be manipulated with the WITZENMANN Configurator, which will be covered in the next section.

2.2.1 General options

use_default_filebrowser^{14.7.0} Starting with version 4.7.0, the plugin now completely ignores this setting and always uses the Windows file browser when the user wants to select a file or a folder.

This setting will be removed in a future version of the plugin.

xml_character_encoding^{4.3.0} Sets the character encoding for all generated XML files. Since the german catalogue descriptions are encoded in ISO-8859-1, this directive also defaults to this value.

If you do not need XML files with german description texts, feel free to set this value to any other character encoding (e.g. UTF-8).

2.2.2 Design options

design_menus_visible_at sets the Design module's applications the menu and the toolbar are visible at. The value of this field is a comma-separated list of application codes.¹

The default value of this option is `ALL`, which causes the toolbar and menu to be visible at all Design applications.

restraint_created_action modifies the behaviour of the Import Tool (see section 3.2). This directive defines an action that will be performed after a restraint has successfully been created from an STK file. Possible actions are: simply change the file extension to `STOLD` (value: `RENAME`) or additionally move or copy the file (values `MOVE_RENAME` and `COPY_RENAME` respectively) to the `WMOLDIMPORT` directory. Alternatively, no action can be performed (value: `NONE`). If the `WMOLDIMPORT` directory does not exist or the according environment variable is not set, `MOVE_RENAME` will have the same effect as the `RENAME` option. `COPY_RENAME` will *first* copy the file to `WMOLDIMPORT` and *then* rename it. Hence, the original file will not be changed.

The action given by this directive only applies to STK files which were taken from the `WMIMPORT` directory. The **other_restraint_created_action** will be performed for other files.

By default, the STK files will be moved to the `WMOLDIMPORT` directory and get the `STOLD` extension (`MOVE_RENAME`).

other_restraint_created_action provides the same possibilities as the previous option, but applies to those STK files which were *not* taken from `WMIMPORT`.

By default, however, these files, too, will be moved to `WMOLDIMPORT` and get the `STOLD` extension (`MOVE_RENAME`).

¹Table B.1 in appendix B shows the default Design application codes.

rotate_full_colour sets the colour which the Rotate Tool (see section 3.3) uses to highlight the full restraint. Accepted values are colour names like RED, GREEN or ROYALBLUE as well as the respective colour codes. The latter are numbers in the range of 1 to 255.

The default value of this option is GREEN.

rotate_first_hanger_colour sets the colour which the Rotate Tool uses to highlight the first hanger.

The option's default value is MAGENTA.

rotate_last_hanger_colour sets the colour which the Rotate Tool uses to highlight the last hanger.

The default value of this option is FORESTGREEN.

rotate_highlight_paragraphs Flag for the Rotate Tool. Determines whether to highlight the paragraphs next to the angle sliders. If this is set to **true**, the paragraphs will get the same colour as the items in the 3D view their respective slider controls.

The default value of this option is **true**.

delta_write_logfile^{†4.4.0} Deprecated.

Use the **delta_logging** directive instead.

delta_logfilename^{†4.4.0} Deprecated.

Use the **delta_logging** directive instead.

delta_logfile_mode^{†4.4.0} Deprecated.

Use the **delta_logging** directive instead.

parts_list_default_output_filename^{4.3.0} Defines the default name of the output file generated by the Parts List Tool. The parts list file will be produced in Microsoft's Spreadsheet XML format, which requires either Microsoft Excel 2003 (or higher) or OpenOffice.org 3.0 (or higher).

Note well that OpenOffice does not handle *.xls files in Spreadsheet XML format appropriately, so you might want to change the file extension to *.xml. Excel instead is (of course) familiar with *.xls files in Spreadsheet XML, so there is no need to change the file extension if you are using Excel. (It also might be more convenient for your users to have the *.xls extension.)

The default value is %PDMSUSER%\Parts list.xml

2.2.3 Draft options

draft_menus_visible_at sets the Draft module's applications the WITZENMANN menu and toolbar are visible at. The value of this field is a comma-separated list of application codes.²

By default the plugin is visible at all Draft User Applications (GEN, ADPCORE, EDITOR), which also is the recommended value of this directive.

bcf_dir The Directory the Backing Sheet Configuration Files (*.bcf) are stored in and loaded from.

bcf_default_style specifies a BCF containing the default style definitions.

²Table B.2 in appendix B contains a list of Draft's default application codes.

bcf_create_dir_at_slash Flag for the Backing Sheet Configuration Tool. Determines whether to create directories for each slash (/) in a backing sheet's name.

If you want to create many BCFs (or already have them), it could be a good idea to store them in different folders. The Backing Sheet Configuration Tool can handle this by creating new folders for every slash in a backing sheet's name. For a backing sheet named /WM_HANG/BACKS/MET/AO, the following directory structure would be created in the **bcf_dir**.

```
WM_HANG/
  BACKS/
    MET/
      AO.bcf
```

However, PDMS needs writing access to this folder.

If you are not sure whether PDMS can create new folders or you do not wish to spread your BCFs over different folders, just set the value to **false**. In this case all slashes in the backing sheet's name (except the first) are replaced by an underscore character (_).³

The default value of this option is **false**.

north_arrow_template^{4.4.0} Defines the template (element type SYTM) that should be used for the north arrow on the key plan and all top views. If no north arrow should be painted, leave this option empty.

Please note that for technical reasons this directive will not be evaluated and validated at PDMS startup but at drawing creation time.

This option's default value is /WM_HANG/SYMBOLS/GEN/MISC/NORTH-ARROW.

north_arrow_scale^{4.4.0} Defines a factor that is used to scale the north arrow to an appropriate size. A value between 0 and 1 shrinks the symbol, a value greater than 1 enlarges it.

The default value of this option is 0,5.

north_arrow_width, north_arrow_height^{4.4.0} These two options define the width and the height of the north arrow symbol respectively (with respect to a scaling factor of 1). They are required in order to correctly position the north arrow on the drawing.

The default north arrow has dimensions of 11 mm×41 mm.

2.2.4 Logging options^{4.4.0}

startup_logging Defines the events that should be logged during PDMS startup (this also includes the evaluation of wm.ini itself).

Please note that it is only possible to log to files during startup.

export_logging Defines the events that should be logged during FIN file creation (files that transfer attachment data from PDMS to Flexperte).

import_logging Defines the events that should be logged during STK import (files generated by Flexperte that describe the support constructions that should be built in PDMS).

delta_logging Defines the events that should be logged during modification checks.

This option supersedes the design directives **delta_write_logfile**, **delta_logfilename** and **delta_logfile_mode**.

parts_list_logging Defines the events that should be logged during parts list generation in the Design module.

drawing_logging Defines the events that should be logged during drawing production.

³Regardless of the value of this option, all characters that windows forbids in filenames (\, :, *, ?, ", <, > and |) will be replaced by underscores, as well.

Option format

Each of the six aforementioned directives expects a list of so-called log handlers. These log handlers define which events will actually be logged. Hence, an empty log handler list for the **export_logging** option results in no logging for the export process.

```
*_logging = [Log-Handler [Log-Handler [...]]]
```

A log handler is defined by a log level, its type and a set of options. These parts are separated by pipe characters (|) and enclosed by angle brackets (<, >).

Figure 2.5 – Log handler definition syntax

```
<log level|type>  
<log level|type|option 1>  
<log level|type|option 1|option 2>
```

Log level The log level controls which events will be processed by the particular log handler and which will be ignored. The following log levels exist (with decreasing severity): **FATAL**, **ERROR**, **WARNING**, **INFO**, **FINE**, **FINER**, **FINEST**.⁴

A log handler only processes those events whose log level is greater than or equal to its own log level. For example a **WARNING** handler will log **FATAL**, **ERROR** and **WARNING** events and ignore all others.

Type The log handler types defines how events (that are not ignored by that handler) will actually be logged. Currently, three different log handler types exist:

CONSOLE The console log handler instantaneously prints arising events to the PDMS console window.

This handler's output could look like this:

```
ATTACHMENT /0811  
[INFO] Asking user whether to overwrite existing element  
RESTRAINT /0811/RE  
[INFO] Answer: YES  
[INFO] Deleting existing element RESTRAINT /0811/RE
```

SCREEN Screen log handlers collect events they are interested in and show them in a dialog window at the end of the executed operation. Figure 2.6 shows an example dialog generated by a screen log handler.

Please note that screen log handlers cannot be used in the **startup_logging** directive.

FILE The file log handler writes arising events into a text file which will be formatted like the console handler's output except for an additional timestamp in front of each log entry.

In contrast to the screen and the console handlers, the files log handler requires an argument (a file name) that follows the **FILE** type after an equality sign.

Please note that the log handler will create the file if it does not exist, but will not attempt to create directories. If you want to log into `C:\pdms\log\import.log` for example, you have to make sure that the directory `pdms\log` on drive C already exists. Otherwise the handler cannot and therefore will not create `import.log` in that directory.

⁴There are also **DEBUG** and **TRACE** log levels, which are even less severe than the **FINEST** level. However, it should almost never be necessary to use these levels for general operation.

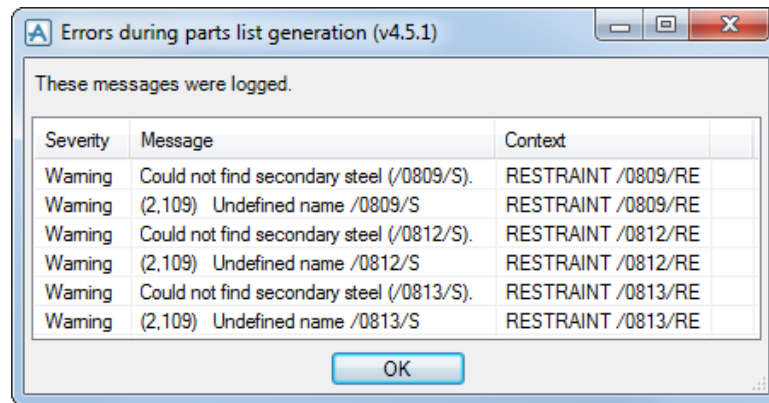


Figure 2.6 – Example output of a screen log handler

Figure 2.7 shows how to configure the three log handler types.

Figure 2.7 – Example usage of the log handlers and their options

```
startup_logging = <INFO|file=%PDMSUSER%\startup.log>

import_logging = <INFO|console> <WARNING|screen|empty-message=Import
                  completed>
export_logging = <WARNING|screen|empty-message=Export completed|title=
                  Messages>

drawing_logging = <FINE|file=%PDMSUSER%\drawing.log|mode=APPEND>
```

Screen log handler options

title Sets the dialogs title text. All characters are permitted except for the pipe character(|) and the closing angle bracket (>).

empty-message This text will be shown to the user if the operation completed without any events being logged by the screen log handler. This could be “Import completed successfully” or any other text that you like. If this option is missing or only contains an empty text, no message will be displayed.

The text must not contain the pipe character (|) or the closing angle bracket (>).

File log handler options

mode Controls whether the log file is overwritten on every run (default) or new messages are appended to the end of the file.

Valid values are **OVERWRITE** (default) and **APPEND**.

Log handler usage


You can use multiple log handlers on one logging directive and there are no restrictions regarding the multiple usage of equal log levels or log handler types. You should note, however, that the use of two or more screen log handlers at the same time will most likely confuse your users and that two or more file log handlers writing to the same file can lead to unexpected results.

2.2 The wm.ini file

Furthermore, all log handlers for a single logging directive have to fit on one line. This imposes a limit of 1024 characters per logging directive.

During drawing production PDMS itself prints some messages on its console windows, hence the console handler's output could get mixed up with the messages generated by PDMS.

2.3 Checking and editing the configuration

The WITZENMANN Configurator helps you to check the environment variables and options set in *wm.ini*. It is available both in the Design and Draft modules and can be started via the WITZENMANN menu and the menu item *Show Configuration...* or the  button in the toolbar. Figure 2.8 shows the form for one of our sample configurations (XYZ).

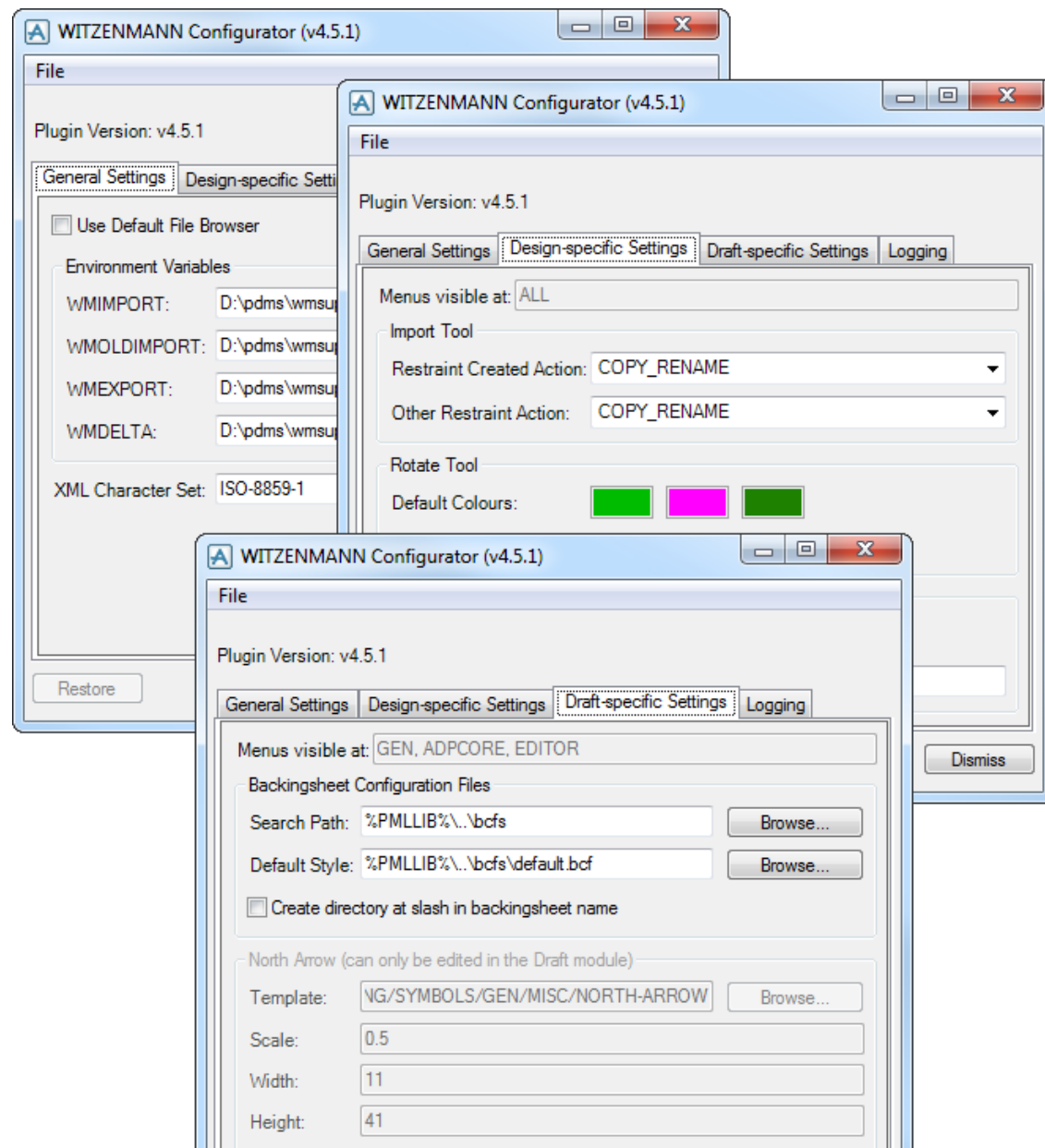


Figure 2.8 – The WITZENMANN Configurator

2.3.1 Editing the options

Except for the deactivated fields (the plugin's visibility in Design and Draft, colour choice for the Rotate Tool), all options can be edited subsequently. Any changes, in case they are valid, will be

updated instantly, but will only apply for your current session because the settings from *wm.ini* are reloaded at every restart of PDMS.

Deactivated form fields

The option for the plugin's visibility in Design and Draft is only relevant while PDMS starts up. Thus, the option is deactivated, since later changes would not have any effect.

The Rotate Tool uses the given colours as default values. The user can change the highlight colours in the Rotate Tool itself, so editing the default values subsequently would be useless.

2.3.2 Resetting the options

When opening the form, the *current* configuration is shown. Any changes since the form's last opening can be restored by pressing the *Restore* button.

At any time, the menu item *Restore settings from wm.ini* in the form's File menu allows you to reload the settings from *wm.ini* without restarting PDMS. Note that the values of WMIMPORT, WMOLDIMPORT, WMEXPORT and WMDELTA will be reset, as well.

Chapter 3

Design

The plugin includes itself in the PDMS Design module in two ways: on the one hand via the WITZENMANN menu in the menu bar, at the other hand as an independent toolbar. Both can be seen in figure 3.1. The toolbar contains one button for each the Import Tool, which provides functions to create the support constructions calculated by Flexperte in Design, the Export Tool, which allows you to pass attachment data to Flexperte, the Rotate Tool, which simplifies the process of rotating support constructions, the Delta Tool, which checks and prints out differences between the 3D model and STK files, the Parts List Tool, which generate parts lists for imported constructions, and finally the WITZENMANN Configurator (see section 2.3).

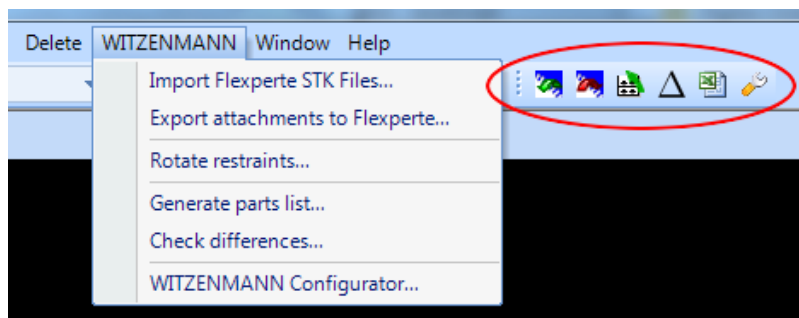



Figure 3.1 – Menu and toolbar in the Design module

3.1 Exporting to Flexperte

The Export Tool of the Flexperte PDMS Plugin allows you to readout attachment data in so-called FIN files (**F**lexperte **I**Nput) that can be imported by Flexperte and thus simplify the design of support constructions. Among others, the FIN files provide information about forces acting on the support point, elevation of anchorage points and the desired hanger type. The tool can be started either via the WITZENMANN menu and the menu item *Export attachments to Flexperte...* or via the  button in the toolbar.

In version 4.5.0 the FIN files have been extended to also contain restraint data. These extended FIN files may be used to synchronize a Flexperte project file with the current state of the PDMS Design model.

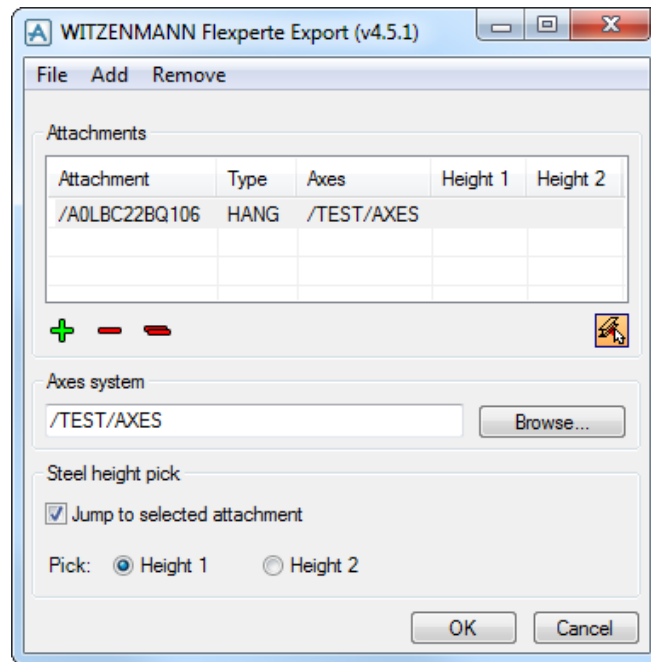


Figure 3.2 – The WITZENMANN Export Tool

3.1.1 Requirements

To use the Export Tool, one of the environment variables `WMEXPORT` and `HANGER`¹ must be set and point to a path PDMS can create files at. Furthermore, a calculating tool that saves its calculated values on the according attachment is required, for example Rohr 2. Appendix D contains a list of User Defined Attributes (UDAs) required for data interchange.

Note that data export is available for named attachments only.

3.1.2 Adding attachments

The Add Menu provides menu items to add single attachments or all attachments of the current pipe, zone, etc. to the list. The particular menu items are (de-)activated regarding the Current Element (CE).

The plus button below the attachment list allows a selective choice of attachments from the database hierarchy. Therefore, the plugin's Database Browser (see section 5.1) opens up, which helps you to insert single attachments or full pipes into the list.

The several list columns show the name of the corresponding attachment, its type (from the attribute `ATTTYPE`²), the support point's axes system and the elevation of the upper (or lower) anchorage points. For single load chains of course only the first value is relevant while double load chains have two values set, which may be different.

¹If only `HANGER` is set, `%HANGER%\export` must be a valid path.
For further information see section 2.1.

²Typically, only attachments of the type `HANG` are chosen for support constructions. However, the Export Tool permits the export of attachments of any type.

3.1.3 Removing attachments

The added attachments can be removed in a quite similar way. You can either delete the selected entries only (single minus button or the *Remove selected attachments from list* option in the Remove menu) or clear the entire list by pressing the double minus button or using the corresponding item from the Remove menu.


3.1.4 Choosing an axes system

Selecting an axes system is optional, but helps Flexperte to put the attachments in the right spatial context by giving it the axes in north, south, east and west direction nearest to the support point.

The coordinate system can be specified directly or via the *Browse...* button,³ which lists all zones in the site /AXES. Note that changes of the axes system will only be applied to the attachments selected in the list. Therefore, the *Axes system* frame is deactivated if no entry is selected.

3.1.5 Setting the installation height

Although the construction's installation height will be transferred automatically, you may set the elevation of the anchorage points manually. Changing the anchorage point elevation will not overwrite the values that eventually were provided by a calculation tool. Instead, the changes will only be stored temporarily.

Clicking the  button starts the selection, which requires at least one list entry to be selected. If your selection contains more than one attachment, it will be reduced to one entry, since setting the elevation is only possible for one element at a time.

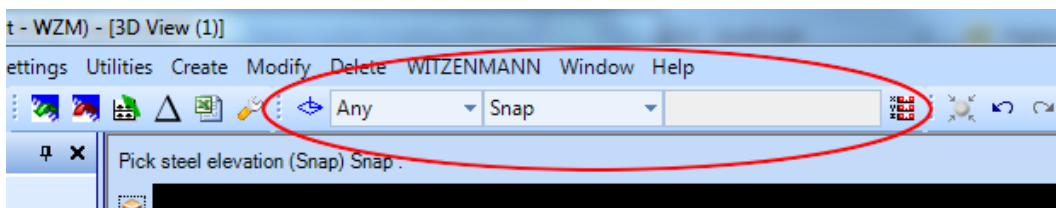



Figure 3.3 – Contents of the status bar during selection mode

If the selection mode is activated, the *Steel height pick* frame will be activated. Additionally, the *Positioning Control* toolbar will become visible and the text *Pick steel elevation* will appear in the PDMS status bar.

The selection itself is similar to Design's distance measuring, which is taking advantage of the *Positioning Control* toolbar, and can be seen in figure 3.4. The *Steel heights pick* frame provides options to change the height you are currently setting. While choosing an appropriate elevation, you will be supported by the *Positioning Control* toolbar and several auxiliary lines in the 3D model. If you activate the *Jump to selected attachment* option, the 3D view will always be centered on your currently selected attachment, which enables you to work even more efficiently.

You quit the selection mode when closing the form, pushing the  button again or pressing the escape key on your keyboard.

³How to define your own axes system is described in appendix C.

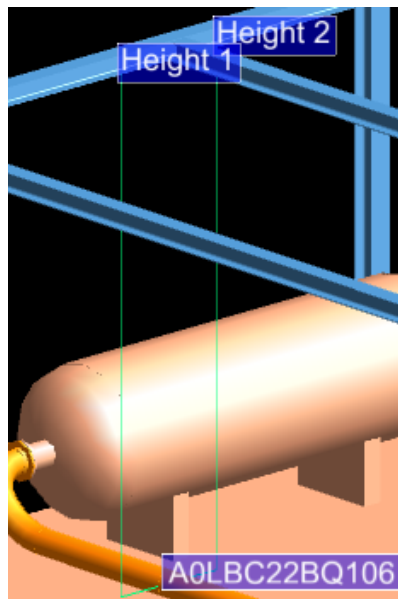


Figure 3.4 – Setting the elevation in the 3D model


3.1.6 Exporting the data

The actual export only requires a click on the OK Button, which will create a single FIN file (<attachment name>.fin) for each list entry and store it in the WMEXPORT directory.

Note that the attachment name should not contain any of the characters \, /, :, *, ?, ", <, > and | since using them in filenames is restricted by Windows.

If any of the created files does already exist in the target directory, you will be asked whether to overwrite the file(s).

3.2 Importing from Flexperte

The Import Tool, which creates the support constructions calculated by Flexperte, is the plugin's main tool regarding the PDMS Design module. While you can use it to create your supports one by one, it is also possible to use the tool for bulk productions. You can start the Import Tool via the WITZENMANN menu and the menu item *Import Flexperte STK files...* or via the  button in the toolbar.

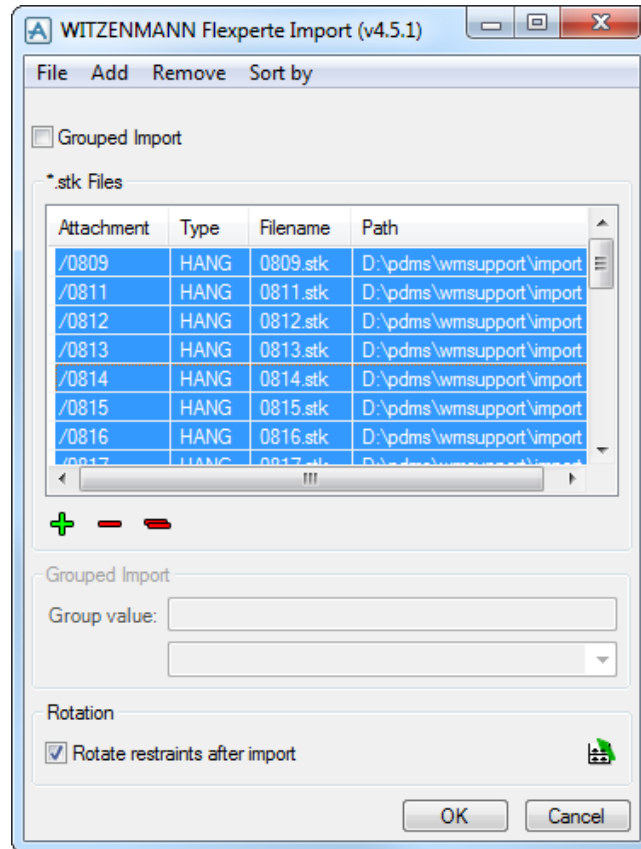


Figure 3.5 – The WITZENMANN Import Tool

3.2.1 Requirements

The Import Tool requires neither any environment variables nor a special configuration to provide its basic functions.

However, to handle STK files for all attachments of a pipe, a zone etc. automatically, at least one of the environment variables `WMIMPORT` and `HANGER`⁴ should be set and point to a valid directory.

Furthermore, we recommended you to set the environment variable `WMOLDIMPORT` since STK files of successfully created support constructions (referred to as STOLD files) will be stored in the directory this variable points to. Storing STOLD files in the `WMOLDIMPORT` directory is required by the Delta Tool (see section 3.5).

⁴In case you only set the environment variable `HANGER`, `%HANGER%\import` should point to a valid directory. For further information see section 2.1.

3.2.2 Adding files

In contrast to the Export Tool (see section 3.1) the Import Tool does not only insert detached attachments into its list, but requires an STK file for each and every attachment and vice versa. The STK file which belongs to a given attachment can be found by simply adding the text `.stk` to the attachment's name. For example the attachment `/A0LBC22BQ106` would be linked to the file `A0LBC22BQ106.stk` (and vice versa). The back link exists as well since STK files contain the name of the attachment they were designed for. While the former relationship is optional, the latter one is mandatory.

You can directly choose files from the file system by pressing the plus button or using the *Add STK Files...* option from the Add menu. If no attachment is found for a given STK file, the file will not be added to the list and an error message will appear, instead.

In case the `WMIMPORT` directory is set, the tool may add all STK files for a selected element automatically. Depending on the Current Element (CE), STK files of single attachments as well as of all attachments of a pipe, zone etc. may be inserted into the list. The files will be searched in the `WMIMPORT` directory and its subdirectories (if existent). Eventually an error message containing those attachments which had no corresponding STK will appear.

Besides this automatic selection of an STK or attachment, you may add a specific attachment-file pair to the list. Use the *Add STK with given attachment...* menu item from the Add menu and specify the desired file and attachment. Keep in mind that STK files contain the name of the attachment they were designed for.

The several list columns indicate the name of the particular attachment, its type (from the attribute `ATYPE`⁵) and the dedicated STK file's name and path respectively.

Sorting the list^{4.2.0}

The *Sort by* menu allows you to sort the file list by any column in ascending or descending order. After the rearrangement, eventually selected list entries will be reselected. New attachment-file pairs, however, will still be appended to the end of the list, which in general destroys the sort sequence. You may only sort by the group column if the *Grouped Import* (see next section) option is checked.

3.2.3 Grouping restraints^{4.2.0}

Grouped import can be en- and disabled with the checkbox of the same name. By enabling this option, another column – the group name – will be added to the file list. Deactivating it causes the column to disappear again. The column displays the values of the User Defined Attribute (UDA) `:WZGROUP` of the contained attachments. The new Import Tool provides a simple way to alter these values.

As shown in figure 3.6, the *Grouped Import* frame displays the groups of the selected attachments. You can change the group value of the selected attachments by entering a new group name into the *Group value* text field or by choosing an existing group from the group list, which is expanded in the figure. The group list will keep its values until you select different entries in the file list. The **unchanged** value in the group list restores the initial grouping. Please note that initial grouping refers to the specific group values at the moment you made your last selection in the file list. Whenever you select different entries in the file list, the group list will be reinitialized with the group values of the corresponding attachments.

You may enter any text as a group name. However, because the group names will later act as zone names for the created restraints, they must not contain spaces.

⁵Typically, only attachments of the type `HANG` are chosen for support constructions. However, you may create constructions for attachments of any type.

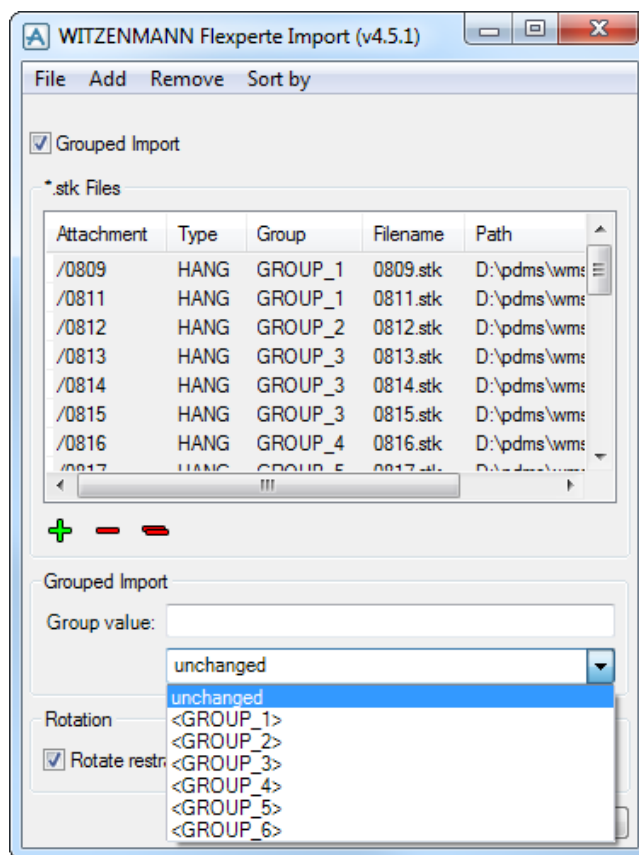


Figure 3.6 – Grouping of selected attachments

3.2.4 Removing files

Both the single minus button and the menu item *Remove selected files* in the Remove menu remove – as the name may suggest – the selected STK files. The double minus button and the *Remove all files* menu item, however, clear the entire list.

3.2.5 Creating the support constructions

When you click the OK button, the listed STK files will be processed sequentially and create their associated restraints. If errors occur while building a restraint, the affected construction will be skipped and a short error message will be printed to the command line. All error messages of the entire run will be collected and displayed in a combined list at the end of the creation process. After all files have been processed, the files which could successfully create their restraint will be removed from the list while the faulty ones will be kept.

Depending on the settings in *wm.ini*, the successfully processed STK files will receive a *.stold file extension and will be moved or copied to the WMOLDIMPORT directory. The directives **restraint_created_action** and **other_restraint_created_action** in section 2.2.2 describe which action will be performed for which file.

If a file of an already existing restraint is processed, a small dialogue allows you to decide whether the respective restraints should be overwritten or kept as they are. In the latter case, however, the tool skips the generation of the new constructions.

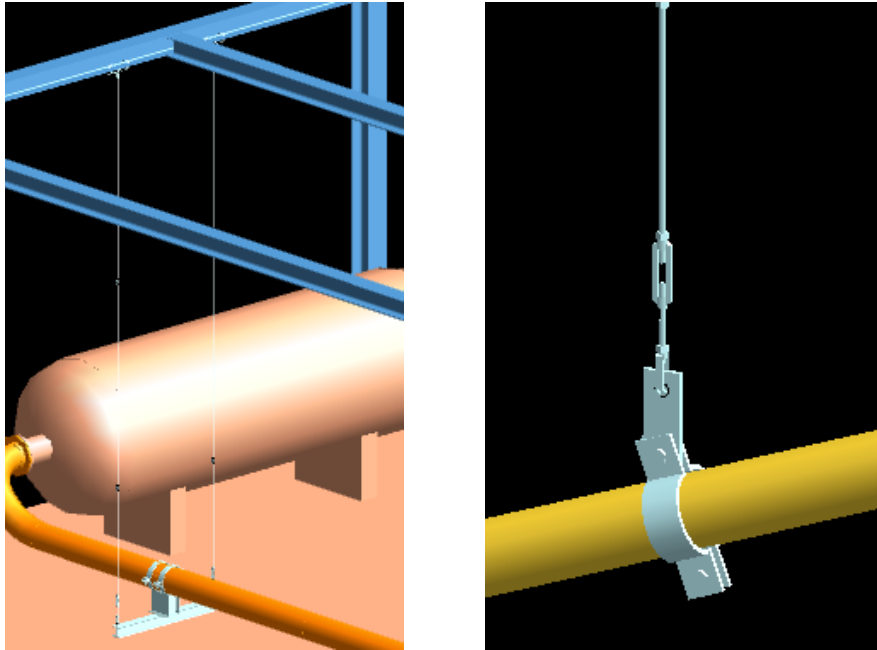


Figure 3.7 – Successfully created support constructions.
The left-hand side shows a double load chain with a traverse, the right-hand side shows a construction on a sloped pipe.

Naming and hierarchy

As stated in the preface, we can directly deduce the restraint name from the attachment name. We associated the attachment `/A0LBC22BQ106` with the file `A0LBC22BQ106.stk`, which in turn

$$\langle \text{restraint name} \rangle = \langle \text{attachment name} \rangle + \text{"RE"}$$

generates the restraint `/A0LBC22BQ106/RE`. However, the restraints will not be created beneath pipe level but in their own zone whose name is calculated differently for “normal” and grouped imports. In the former case the name of the target zone can be retrieved by using the name of the

$$\langle \text{restraint's zone name} \rangle = \langle \text{attachment's zone name} \rangle + \text{"-R+S"}$$

$$\langle \text{restraint's zone name} \rangle = \langle \text{attachment's group name} \rangle + \text{"-R+S"}$$

attachment's zone, in the latter case we simply use the group name of the attachment. In both cases, however, the zones are created in the attachment's site.

3.2.6 Postprocessing

If the option *Rotate restraints after import* is activated during the import, the Rotate Tool, which allows you to orientate the constructions as needed, will open afterwards. The following section describes how to rotate support constructions. Figure 3.8 shows the construction from our previous example. In this case postprocessing is absolutely necessary, since the steel clamp collides with the girder instead of surrounding it.

Furthermore, you may edit the created constructions as you like. To follow the modifications in the Flexperte project files (*.wit) later, each and every change in the 3D model needs to be transferred to the Flexperte project since the project file forms the basis of WITZENMANN's proposal management. The Delta tool, which is described in section 3.5, supports you in data interchange.

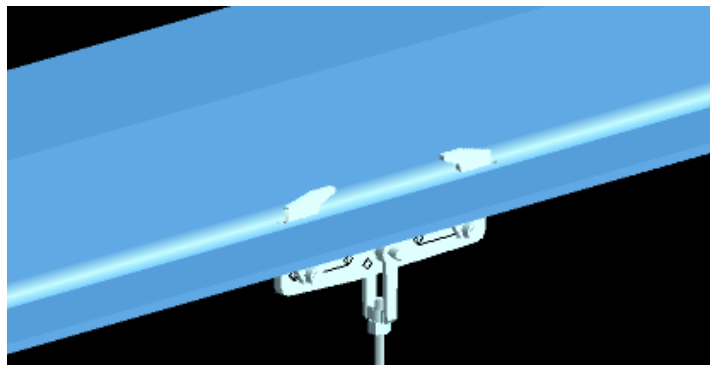




Figure 3.8 – Wrongly aligned steel clamp

3.3 Rotating constructions

The Rotate Tool is used to orientate constructions in the 3D model. It supports you in both creating new support constructions and editing existing ones. Possible use cases include the rotation of a construction about a vertical pipe or the orientation of a steel anchorage point towards a girder. Note that the Z or UP axis respectively is the only rotation axis supported by the tool. The tool can be started via the WITZENMANN menu and the menu item *Rotate restraints...* or via the  button in the toolbar. Possible use cases include the rotation of a construction about a vertical pipe or the orientation of a steel anchorage point towards a girder. Note that the Z or UP axis respectively is the only rotation axis supported by the tool. The tool can be started via the WITZENMANN menu and the menu item *Rotate restraints...* or via the  button in the toolbar.

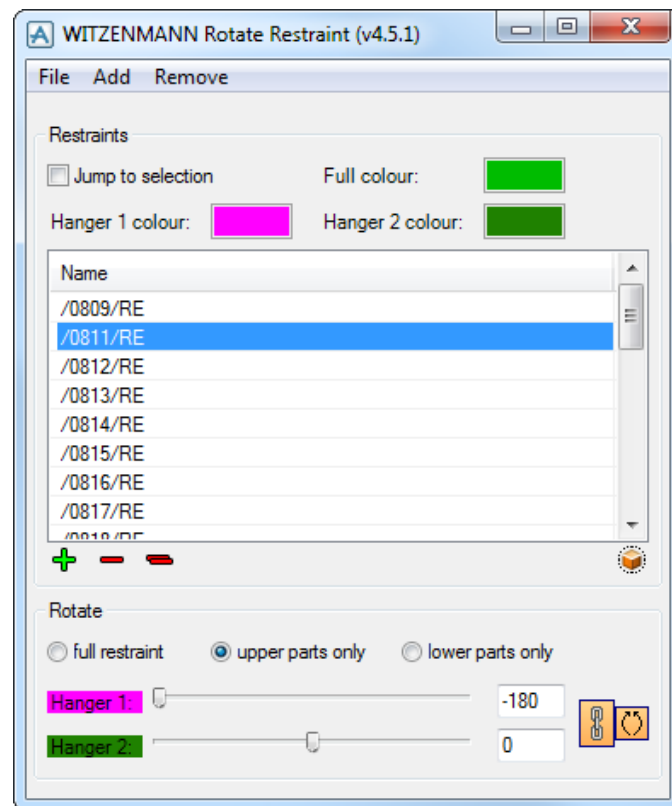


Figure 3.9 – The WITZENMANN Rotate Tool

3.3.1 Requirements

There are no external requirements for using the tool, especially, restraints do not have to be named to be enabled for rotation. However, a certain organization concerning the restraint's internal hierarchy is required. All WITZENMANN load chains are build top down, thus the upper parts of a chain are the first items in the members list of the corresponding hanger.

3.3.2 Adding restraints

If you use the Rotate Tool in combination with the Import Tool, the imported constructions will be inserted into the restraint list automatically. Otherwise restraints may be added, like in the Export and Import Tools, using the plus button and the Add menu respectively.

3.3.3 Removing restraints

Single restraints can be removed from the list by pressing the single minus button or selecting the *Remove selected restraint* option in the Remove menu. In order to quicken editing long lists of restraints, removing an entry from the list will cause the list to select a new one automatically.

As usual, the entire list can be cleared by clicking the double minus button or using the *Remove all restraints* menu item.

3.3.4 The rotation

Selecting a restraint is the first and easiest step. You may use the option *Jump to selection* to center the selected restraint in the 3D view at any time.

The three options in the *Rotate* frame control which parts of a restraint will be rotated. You can either rotate the entire construction or just a leg's upper or lower parts. The parts selected for rotation will be highlighted in the 3D view.

The slider and text gadgets perform the actual rotation. You can move a slider to adjust the angle in 1° steps or enter a specific angle into the text field next to the slider.

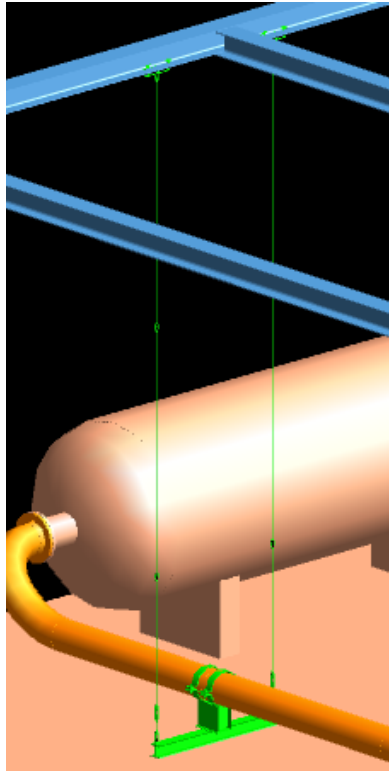






Figure 3.10 – An entirely highlighted support construction

Rotating double load chains is special since you may rotate the single legs together or separately, which can be chosen by the button next to the text fields. Its two states  and  imply a combined and an individual rotation respectively.

In the former case, the second button indicates whether the other leg should be rotated in the same () or the opposite direction ()

In our previous example the steel clamps were aligned wrong and collided with the girder. Figure 3.11 again shows the clamp's malposition and the highlighting of a restraint's upper parts. The right leg is highlighted in the first slider's colour while the colour of the second slider is used for the left one. Hence, the upper slider controls the angle of the right leg, and the lower slider

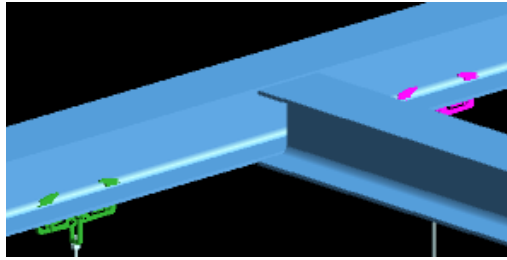


Figure 3.11 – Rotating a restraint's upper parts

the angle of the left one. If we use a combined rotation, the second leg automatically follows the turning when we alter the angle of the first one. Thus, changing the left leg's angle by 90° has the desired effect.

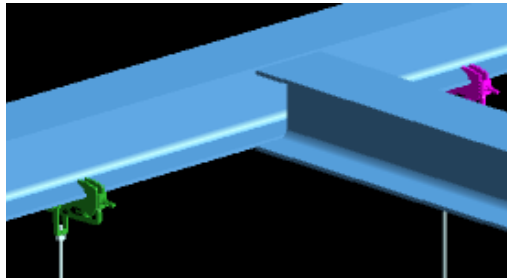



Figure 3.12 – Rotation completed

The three buttons in the form's upper area define the colours the items in the 3D view are highlighted with. When you open the form the first time, the corresponding values from *wm.ini* will be stored in these buttons. If the option **rotate_highlight_paragraphs** (see section 2.2.2) is activated, the texts next to the sliders will be highlighted in the same colour as the corresponding parts in the 3D view.

3.4 Parts list generation^{4.3.0}

Parts list generation used to be possible during drawing production only. The Parts List Tool allows you to directly extract parts lists from the design module. These lists are stored in Microsoft's Spreadsheet XML format, which can be viewed and edited with Microsoft Excel 2003 (and higher) as well as OpenOffice.org 3.0 (and higher). The tool can be started via the WITZENMANN menu and the menu item *Generate parts list...* or the  in the toolbar.

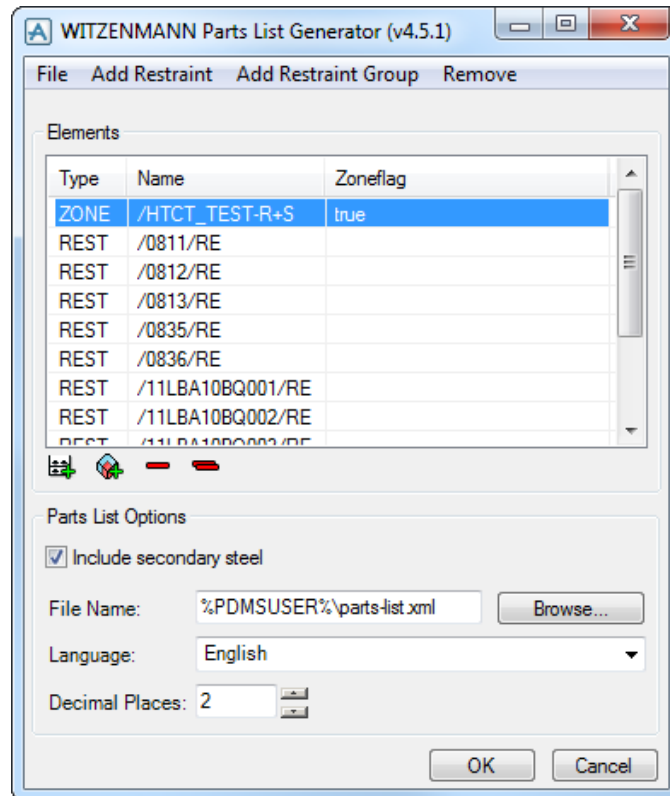




Figure 3.13 – The WITZENMANN Parts List Tool

3.4.1 Requirements

The tool does not require the presence of any environment variable. However, you should ensure that the directive **xml_character_encoding** in *wm.ini* is set appropriately for your needs (see section 2.2.1). This option defines the character set that is used for all generated XML files, which include the parts list files in particular. If you want to extract parts lists with German description texts, we recommend to use the ISO-8859-1 character set, which can be used for the English descriptions, too.

3.4.2 Adding elements

The parts list generator is the first tool of the WITZENMANN Flexperte PDMS Plugin that offers the possibility to handle both single restraints and restraint groups (zones) in a single list. Hence, restraints and zones may be added using the particular menus (*Add Restraint* and *Add Restraint Group* respectively) and buttons ( and  respectively). As usual you can add multiple elements to the list at once.

Duplicates

Do not mix up restraints and zones in the list, except you know what you are doing. Keep in mind which restraints are included in which zones.

Let us assume that a given zone contains the restraints /1/RE, /2/RE and /3/RE, and we add the zone as a restraint group as well as the single restraints /1/RE and /2/RE to the list. The parts of the two single restraints will occur twice in the resulting parts list. There will be a security mechanism in a future release but for the moment we recommend you to be cautious if you do not want doubled part and weight information in your parts lists.

3.4.3 Removing elements

As usual, the single minus button does the same thing as the menu item *Remove selected elements* while the double minus button and the menu item *Remove all elements* clear the entire list.

3.4.4 Options

In the lower part of the form you can decide whether the parts list should contain the secondary steel, too. Moreover, you can change the filename, the language of the description texts, and the precision of decimal fractions.

Secondary steel

If the option *Include secondary steel* is checked, the tool will try to find the corresponding secondary steel elements of every list entry. A single restraint is linked to its secondary steel by the

$$\begin{aligned}\langle \text{restraint name} \rangle &= \langle \text{attachment name} \rangle + \text{"RE"} \\ \langle \text{secondary steel name} \rangle &= \langle \text{attachment name} \rangle + \text{"S"}\end{aligned}$$

corresponding attachment name. The secondary steel name of a restraint group is calculated by using the group name.

$$\begin{aligned}\langle \text{zone name} \rangle &= \langle \text{group name} \rangle + \text{"-R+S"} \\ \langle \text{secondary steel name} \rangle &= \langle \text{group name} \rangle + \text{"S"}\end{aligned}$$

If the generator finds an element with the given steel name, it will search beneath the secondary steel container for elements which can be added to the parts list. Besides girders (SCTN elements), these elements include plates (BOX elements), round profiles (CYLI elements) and any element which has a valid SpReference.

As any other element the steel items receive a description and a material text in the final parts list. However, the tool will only calculate the masses of SCTN, BOX and CYLI elements.⁶ For all other steel elements the tool uses the value from the :USTWEIGHT[1] attribute of the element's SpReference.

Filename

Initially this field receives the value of *wm.ini*'s **parts_list_default_output_filename** directive. When you assign another file extension, consider that both Microsoft Excel and OpenOffice.org can handle *.xml files, but OpenOffice.org chokes a little on *.xls files in Spreadsheet XML format (see section 2.2.2).

⁶BOX elements get the material 1.0038 / S235JRG2, CYLIs receive 1.0533 / S355J0 as their material. The masses of both primitives are calculated with an assumed density of $7.85 \frac{\text{g}}{\text{cm}^3}$.

3.4.5 Creating the parts list

After you have added all constructions and groups that you wanted to extract, simply click the OK button and your parts list will be produced. The tool prints warning messages for missing steel elements to the command line and displays them in a list at the end of the generation process.

3.5 Comparing modifications

The objective of the WITZENMANN Flexperte PDMS Plugin is to support the designer without forcing him to accept everything the plugin does. Thus, every created support construction may be edited subsequently. The Delta Tool helps you to keep consistency between the data in the 3D model and the Flexperte project, and can be started via the WITZENMANN menu using the menu item *Check differences...* or via the Δ button in the toolbar.

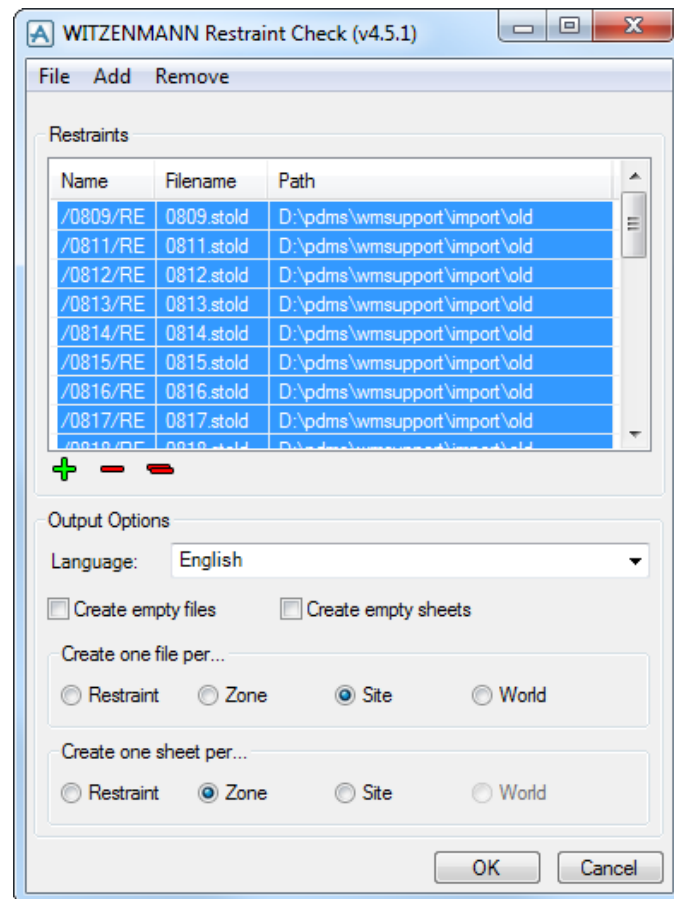


Figure 3.14 – The WITZENMANN Delta Tool

3.5.1 Requirements

To use the Delta Tool, either the environment variables WMOLDIMPORT and WMDELTA, or HANGER⁷ have to be set and point to valid directories. Note that PDMS must be able to create files in the WMDELTA directory while WMOLDIMPORT only requires read-only access. Furthermore, change comparison is restricted to named support constructions because the STOLD file for a given restraint can only be found by the restraint's name.

3.5.2 Adding constructions

As already explained for the three other Design tools, restraints may be inserted into the list using either the plus button or the Add menu. In contrast to the plus button, the Add menu provides

⁷If you only set HANGER, then %HANGER%\import\old and %HANGER%\delta have to point to valid directories. For further information see section 2.1.

options to insert all restraints of a given pipe or zone into the list.

Whenever a restraint is added to the list, the tool scans the WMOLDIMPORT directory and its subfolders searching for an STOLD file that matches the given restraint. Although restraints without a corresponding STOLD file will be added to the list as well, their last two columns will contain the text *<not found>*.

3.5.3 Removing constructions

In order to remove restraints from the list, the Delta Tool, too, provides the two minus buttons and the respective menu items in the Remove menu.

3.5.4 Creating delta files^{4.3.0}

When you click the OK button, the tool processes the listed restraints subsequently and compares them to their corresponding STOLD file.

Since the plugin version 4.3.0 the differences between actual and target state of the support constructions are no longer printed into text files (*.delta) but formatted worksheets. For this purpose the Delta Tool uses Microsoft's Spreadsheet XML format, which can be viewed and edited by both Microsoft Excel 2003 and OpenOffice.org 3.0 (and above).

The radio buttons allow you to control which outputs should go together into a worksheet and which worksheets should be combined to a workbook. Per default the differences of all restraints of a given site will be written into a common file, which contains a worksheet for every zone in this site. The two checkboxes above these radio buttons enable and disable the generation of empty files and worksheets respectively.

3.5.5 Interpreting the results

The only purpose of the Delta Tool is to compare the model's current with its target state. The task of determining how differences between these states arose rests with the designer. He is the one who should recognize that a particular element was just changed (as seen in figure 3.15) and not deleted and recreated.

	A	B	C	D	E	F
1	Fehlende Elemente:	Anzahl	Unit	Restraint	Beschreibung	Materialnummer
2		1	pcs	/0811/RE	Kastenschelle VKK 0400.063.1200.12	624734
3	Total:	1	pcs			
4						
5	Zusätzliche Elemente:	Anzahl	Unit	Restraint	Beschreibung	Materialnummer
6		1	pcs	/0811/RE	Kastenschelle VKK 0400.063.1200.12	
7	Total:	1	pcs			
8						
9						

Figure 3.15 – Output created by the Delta Tool for a changed restraint

Chapter 4

Draft

Like in the Design module, the WITZENMANN Flexperte PDMS Plugin includes itself in the Draft module by creating the WITZENMANN menu and a toolbar, which both can be seen in. The toolbar contains the buttons for each the Drawing Tool, the Backing Sheet Configuration Tool, which can be used in order to create Backing Sheet Configuration Files (BCFs), and the WITZENMANN Configurator (see section 2.3).

4.1 Drawing production

The Drawing Tool is the plugin's main feature in the Draft module. It executes all important steps for drawing generation, thus it is larger than the other tools of the plugin. The tool is divided into three pages, each dealing with a different type of information.

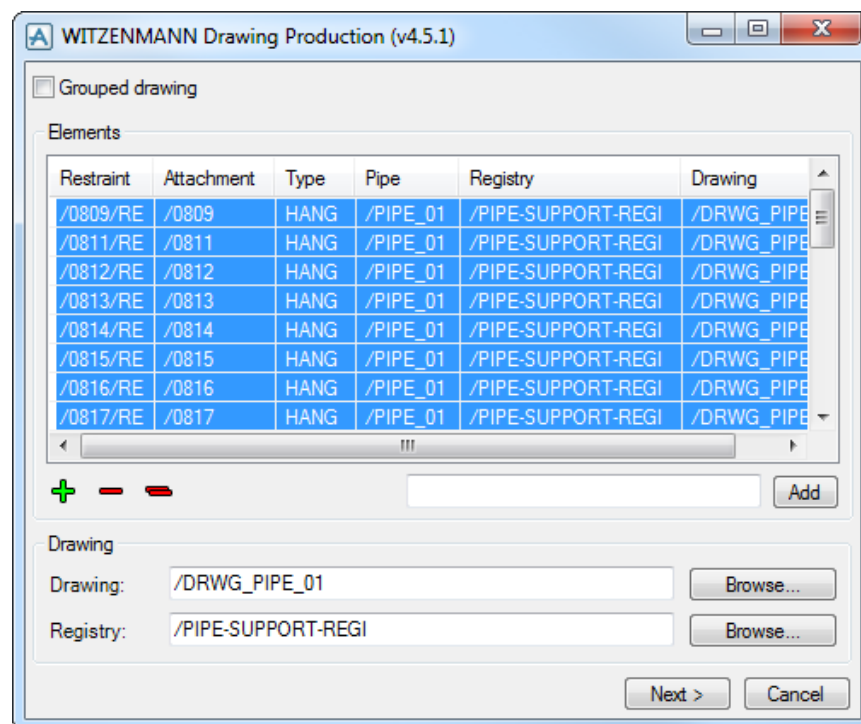



Figure 4.1 – The WITZENMANN Drawing Tool

In particular, these pages request the support points which shall be drawn, a backing sheet

and the size of the generated views, and some additional information. The tool can be started via the WITZENMANN menu and the menu item *Draw restraint...* or via the  button in the toolbar.

4.1.1 Requirements

To use the tool's basic functions, no special requirements have to be fulfilled even though it is recommended that the **bcf_dir** directive is set and points to a valid directory, which enables the tool to automatically select a Backing Sheet Configuration File for a given backing sheet. (See section 2.2.3 for further information on the **bcf_dir** directive.)



4.1.2 Choosing attachments

The tool's first page is used for selecting the support points which shall be drawn. Attachments can be added both by entering their name into the text field and pressing the *Add* button, and by selecting them from the database hierarchy using the plus button. In both cases you have the possibility to insert all attachments of a pipe into the list at once.

Since drawing production also requires a support construction, you may perform the same steps for a restraint. As usual, attachment and restraint will be linked by their name.

$$\langle \text{restraint name} \rangle = \langle \text{attachment name} \rangle + "/\text{RE}"$$

Removing attachments

As usual the  button removes the selected list entries while the  button clears the entire list.

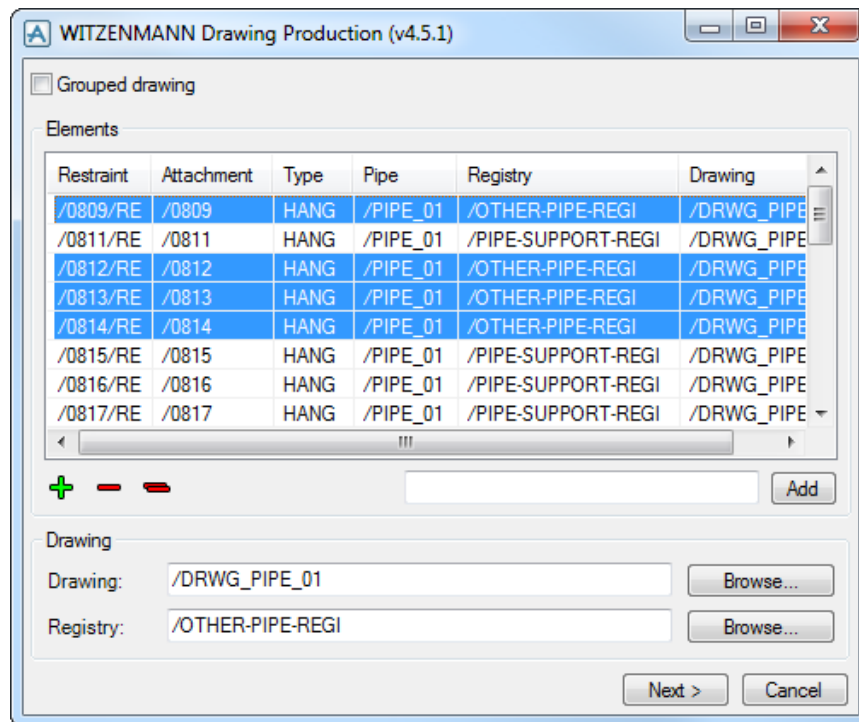


Figure 4.2 – Changing the registry of some support points

Registry and drawing elements

In order to generate the drawings, you have to define a registry and a drawing (both referring to PDMS database elements) in which the drawing sheets can be stored for each and every attachment.

The required data can be specified in the according text fields located at the page's lower area. Again, you may directly enter the drawing's or registry's name or select the corresponding elements from the database hierarchy using the *Browse...* button. However, note that all changes will only be applied to the selected list entries. That is why the lower frame is deactivated if no list item is selected.

While the specified registry has to exist in order to create the drawings, the given drawing element (again referring to the PDMS database element) will eventually be created by the tool.

4.1.3 Choosing elements for a grouped drawing^{4.2.0}

The generation of grouped drawings may be en- and disabled by checking and unchecking the corresponding checkbox respectively. The activation of grouped drawing mode exchanges the attachment list with a zone list while the deactivation brings the attachment list back to top.

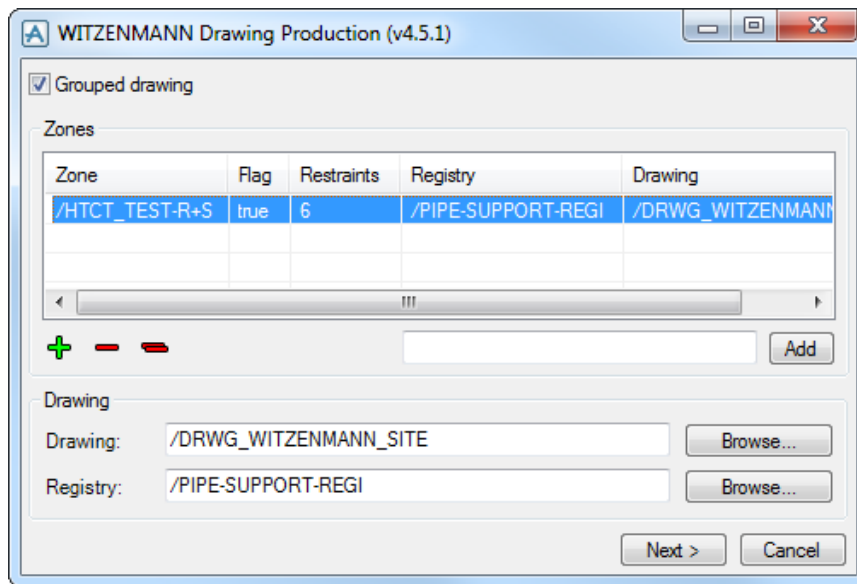


Figure 4.3 – A zone containing six restraints

The list columns now display the name of the zones to be drawn and the number of restraints each zone contains. The registry and drawing columns are kept from the attachment list. However, a *Flag* column was added to the zone list, which denotes whether the zone was created by the Import Tool in grouped import mode (see section 3.2.3). The value of this column is determined by the UDA :WZZONEFLAG of the zone in question, which defaults to false. If you create restraints in grouped import mode, the Import Tool sets the UDA's value to **true**. It is recommended to only create drawings from zones that have this attribute set to **true** – provided that you do not manually change its value. Nevertheless, you have the possibility to generate drawings of any zone.

Please note, however, that each zone must contain at least one restraint. Otherwise you will get an error message when you try to switch to the tool's second page.

Addind and removing zones

Zones may be added to the list by using the well-known plus button or entering their name into the text field below the list (just like attachments were added to their respective list). Now, however, the database browser only allows the selection of zones and sites.

Removing a zone from the list is just the same as removing an attachment, and so is changing the registry and drawing names.

4.1.4 Draw list, backing sheet and views

In the upper area of the form's second page you may specify whether you want to add secondary steel and connected pipes to your drawing's draw list. You may also include the pipes' insulation volume.

If you added the secondary steel to your draw list, you probably want to check the *Fit volume to secondary steel* option. If this option is unchecked, a warning sign ⚠ appears, which should remind you that some parts of the secondary steel might not be visible on the drawing. However, you can always adjust the area displayed on the drawing by changing the *Additional Volume* text field's value.

Furthermore, you can set an additional draw list whose contents will be added to your drawing.

Figure 4.4 – Page 2 of the WITZENMANN Drawing Tool

Backing sheet selection

Figure 4.5 shows the tool which is used to select a backing sheet for your drawing. You may choose between the predefined formats A0 to A3 or enter a custom sheet format by choosing the option *Other*. The tool will offer you list of backing sheets which have the selected format.

By default, the backing sheet list is updated automatically whenever you change to a different format. If the update process takes a lot of time, it may be feasible to uncheck the *Update backing sheet list automatically* option and to press the *Update List* button when required.

A backing sheet can be used for drawing production by double-clicking¹ on it or by selecting the desired backing sheet in the list and pressing the *OK* button.

¹Note that E3D 2.1 doesn't support double-clicks in PML forms anymore.

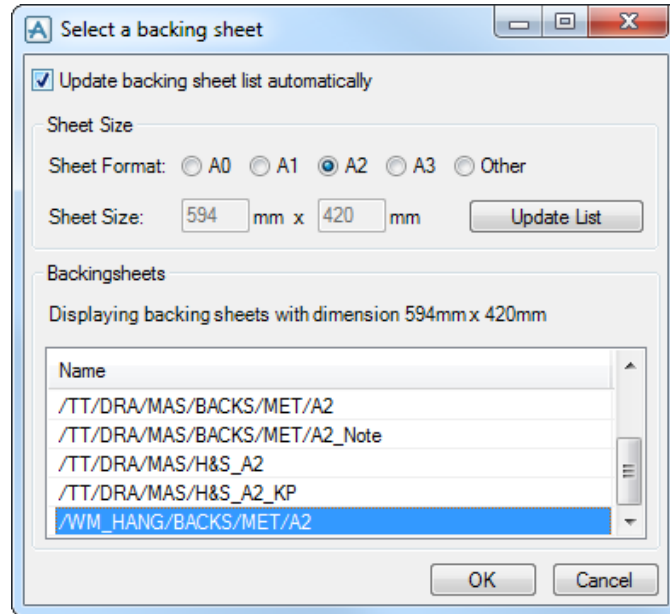


Figure 4.5 – The Backing Sheet Selector

Backing Sheet Configuration File

Every backing sheet needs a Backing Sheet Configuration File (BCF),² which, among others, defines where the views or the parts list shall be positioned and how many views can be created on that backing sheet.

If the **bcf_dir** directive is set and points to a valid directory, the drawing tool will automatically search for a matching BCF when you select a backing sheet. In case the BCF cannot be found, a dialogue window provides options to manually pick an existing BCF or to create a new one. An already chosen BCF may also be edited by pressing the *Create...* or *Edit...* button respectively. The exact process is described in section 4.2.

When you proceed to the form's third page the tool will check whether you selected a valid BCF and whether it belongs to the selected backing sheet. Please note, however, that it is not mandatory, but highly recommended, to select a BCF that was specifically designed for the selected backing sheet.

Views

As figure 4.4 shows, the lower area of the form's second page is used for view definitions. As mentioned above, the BCF specifies the number of views that can be created on a particular drawing. Therefore, you may not be able to edit all four view definitions.

After you have selected the number of views you want to create on your drawing, you can set the viewing direction for these views. The lists contain both absolute (e.g. North, East, ISO 1) and relative directions (e.g. front, side). The latter ones will be evaluated on a per-restraint-basis.

Each view has a label that, by default, shows the direction of that view. When you check the *Use custom view labels* option, you may enter your own label text for each view.

4.1.5 Additional information

The form's last page allows you to enter some additional information. Please note that some options (i.e. parts list, key plan and load table creation) depend on the selected Backing Sheet Configuration File and thus may be disabled.

²The structure of a BCF is described in appendix F.

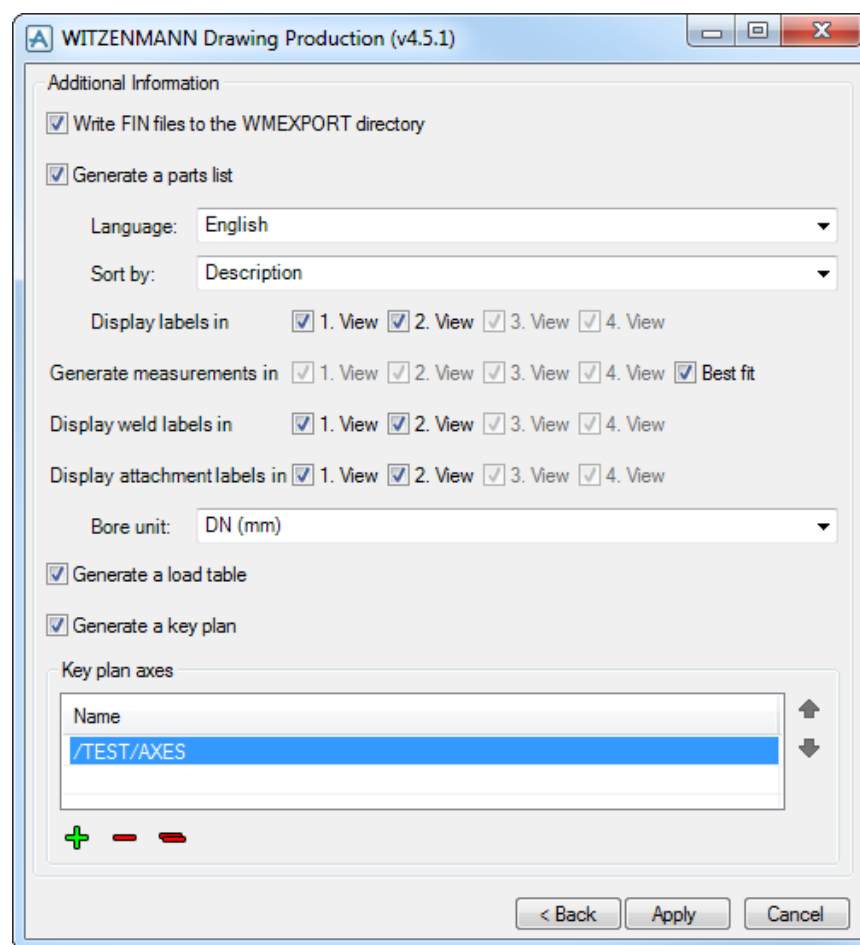


Figure 4.6 – Page 3 of the WITZENMANN Drawing Tool

Exporting the current state of the Design model^{4.5.0} The *Write FIN files to the WMEXPORT directory* checkbox controls whether the drawing tool exports the current state of the Design model to FIN files. These files allow you to easily synchronize your Flexperte project file with the current design model.

In contrast to the Export Tool, the Drawing Tool writes its FIN files to a separate directory below the WMEXPORT directory.

Display in options^{4.4.0} Some options allow you to add information to particular views of the drawing. For example, you may specify that parts list labels will be displayed in the first – and only the first – view of the drawing.

Parts list

The parts list is grouped by items, rods and secondary steel.³

Each line contains an item number, the number of times that particular item is visible on the drawing, a describing text, the WITZENMANN material number, the mass of that single item as well as the total mass of all items of that type. For rods and girders the item length is shown instead of its mass. Each section contains a subtotal mass.

You may set the parts list language to English, German or Polish.

³The parts list will only contain secondary steel elements if you have checked the *Include secondary steel* option on page 2.

Steel elements As already described in section 3.4.4, we determine the steel elements for grouped and single restraints in a slightly different way. While at ungrouped drawings we look for

$$\begin{aligned}\langle \text{restraint name} \rangle &= \langle \text{attachment name} \rangle + "/\text{RE}" \\ \langle \text{secondary steel name} \rangle &= \langle \text{attachment name} \rangle + "/\text{S}"\end{aligned}$$

secondary steel on a per restraint basis, grouped drawings only get one steelwork per group.

$$\begin{aligned}\langle \text{zone name} \rangle &= \langle \text{group name} \rangle + "-\text{R}+\text{S}" \\ \langle \text{secondary steel name} \rangle &= \langle \text{group name} \rangle + "/\text{S}"\end{aligned}$$

In both cases, however, the tool collects all elements which are either girders (SCTN), plates (BOX), or round profiles (CYLI) or have a valid SpReference, adds them to the parts list including their description and material text. It only calculates the masses of SCTN, BOX and CYLI elements, though.⁴ For all other steel elements the tool uses the value from the :USTWEIGHT[1] attribute of the element's SpReference.

Sorting^{4.3.0} The list below language setting can be used to control by which column the parts list will be sorted. You may choose between a sorting by description, material number and no sorting at all. Independently of your selection, the parts list will still be divided into groups of parts, rods and secondary steel.

Item numbers in views^{4.4.0} You may specify the views that shall contain labels with item numbers that point to the corresponding item.

Chain dimensions

The *Measurement* options allow you to specify the views that chain dimensions will be created in. If you have checked the *Best fit* option, the drawing tool tries to determine the best possible view. In almost every case this will be the view whose viewing direction is closest to the front view. However, restraints on a sloped pipe will be dimensioned in a view that is closest to the side view.

If you uncheck the *Best fit* option, the drawing tool will create chain dimensions in every view you selected.

Support point labels

You may also specify the views that attachment labels will be created in. These labels contain the names of the respective attachment and its pipe as well as its nominal bore.

Since version 4.4.0 you can select whether bores will be printed in DN (milli meters) or NPS (inches).

Load table



The load table shows the acting forces, movements and environmental conditions of the pipe. This is, of course, only possible if the according values have been calculated and transfered from Flexperte to PDMS. Note that the Drawing Tool does not calculate any forces or movements et cetera, but simply prints them in the load table.⁵

⁴BOX elements get the material 1.0038 / S235JRG2, CYLIs receive 1.0533 / S355J0 as their material. The masses of both primitives are calculated with an assumed density of $7.85 \frac{\text{g}}{\text{cm}^3}$.

⁵Appendix E contains a sample layout of a load table.

Key plan

The key plan shows the drawn support point with respect to one or more axes systems,⁶ which can be selected in the respective frame. As figure 4.7 shows, the axes closest to the attachment are labeled and the distances to these axes are shown.

If one or more views will contain chain dimensions, the shown elevations will be with respect to the first axes system in the list. You may change the order of the axes systems using the  and  buttons.

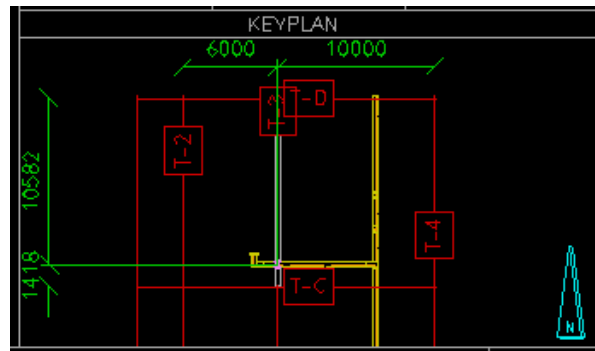



Figure 4.7 – Key plan of a sample drawing

4.1.6 Creating the drawings

When you click the *Apply* button, all data will be checked again and the tool will finally create the drawings. If a particular drawing does already exist, you will have the possibility to decide whether it should be overwritten or stay untouched.

⁶Appendix C describes what can be selected as an axes system.

4.2 Backing sheet configuration

Drawing production used to be strongly dependent on the used backing sheet. At times, drawing production even required the use of predefined backing sheets and formats. In order to provide you with a drawing generation that is independent of a special type of backing sheet, the used sheets have to be configured *once*. This happens with the Backing Sheet Configuration Tool which is shown in figure 4.8. The tool can be started via the WITZENMANN menu and the menu item *Configure backing sheet...* or via the  button in the toolbar. The tool works on so-called Backing Sheet Configuration Files⁷ (BCF). The plugin already contains four of these files, which can be used for the default WITZENMANN backing sheets.

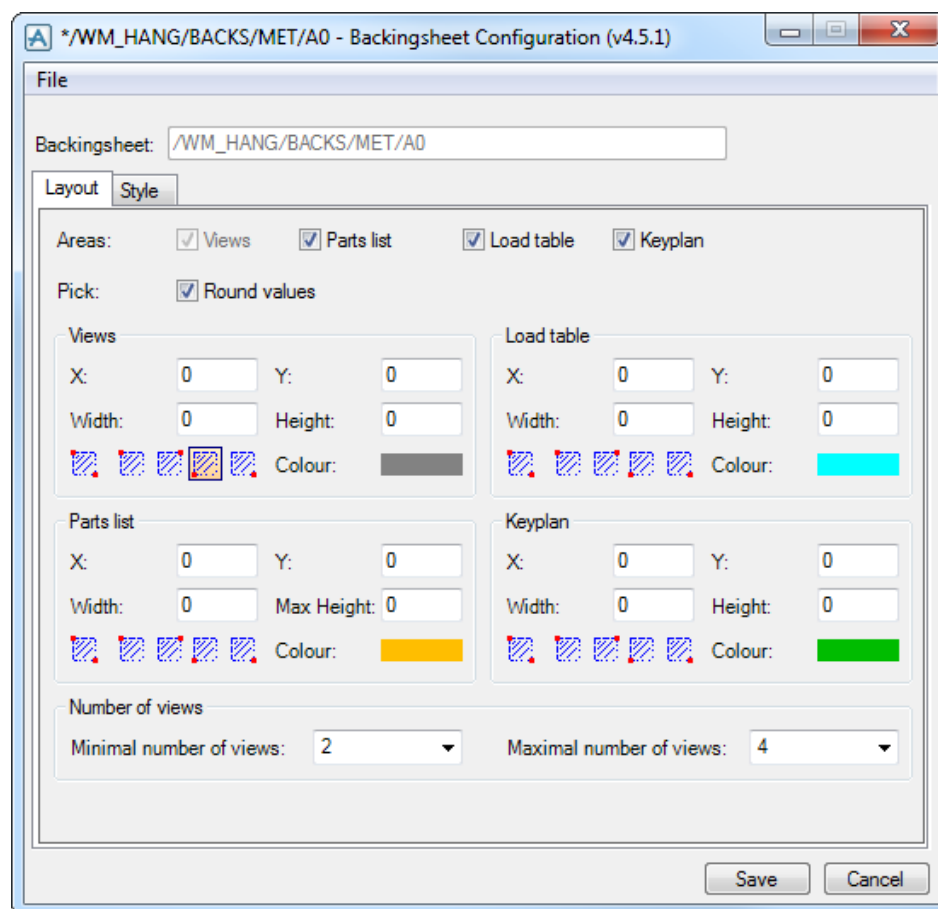


Figure 4.8 – The WITZENMANN Backing Sheet Configuration Tool

4.2.1 Requirements

The Configuration of backing sheets requires a temporary drawing, which will be created automatically. Thus, the tool's basic function requires a drawing element (DRWG) you have write access on. Additionally, we recommend you to set the directives **bcf_dir** and **bcf_default_style** in the *wm.ini* file. The former points to the directory the BCFs are stored in and loaded from while the latter refers to a BCF that contains your company-owned style definitions.

⁷appendix F describes the layout of a BCF in detail.

4.2.2 Create and open

To create a new BCF, you can either start the tool while standing on a backing sheet element (BACK) or choose the *New...* menu item from the tool's File menu. The latter, in turn, requires you to pick the backing sheet you want to configure from the database hierarchy. In both cases, the tool creates a temporary drawing having the chosen backing sheet as an underlay, and activates the Layout and Style tabs.

To open an existing BCF, choose the menu item *Open...* from the File menu. When you pick the particular BCF, again a temporary drawing will be created and filled with the chosen BCF's data.

If an BCF was already open at this time, you will be asked whether you want to save the changes made to this BCF.

4.2.3 Defining the areas

On every backing sheet, space for upto four areas can be defined. The four checkboxes control which of the areas should be allowed on the backing sheet.

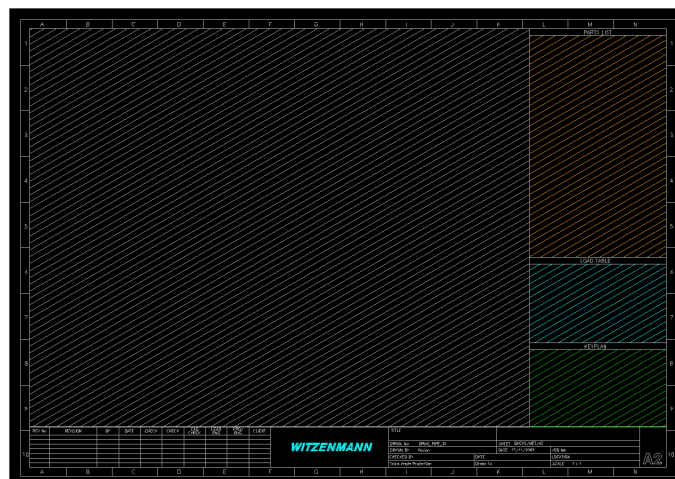


Figure 4.9 – Area definitions in a sample BCF

Size of the areas

To set an area's size, you can either manually enter its coordinates into the according text fields or pick them directly from the drawing using the buttons. An area always starts at its top left corner and extends to the bottom right.

With the respectively first button an entire area can be redefined, which requires two clicks that will be interpreted as two opposite corners of a rectangle. Each of the following four buttons allows you to reset a the position of a particular corner. Note that negative widths and heights are possible.

If the option *Round values*, which is positioned in the upper part of the form, is set, the coordinates of a click will be round to integer values. This can be undesirable if you want to position a rectangle's corner exactly on to a line. If you switch the option off, the actual coordinates of your click will be used.

Regardless of the mode you changed the areas, every redefinition causes the tool to redraw the affected rectangle on the sheet. The colours in which the rectangles are highlighted can be found on the form.

Views

This area acts as a container for the views of the drawing. Thus, its definition is mandatory. Note that the size of a single view must be at least 10 mm × 10 mm. Hence, this area should be defined big enough.

Parts List

This area will contain the later drawing's parts list. Note that the parts list always fits the area's width but not necessarily its height.

Load Table

Just like the parts list, the load table fits in the area's with but in general not its height.

Key plan

The key plan fills all of its available space. Note that the key plan is realized by a view. Thus, this area again has to have a minimum size of 10 mm × 10 mm.

4.2.4 Minimum and maximum view count

At the bottom of the form you may define the minimum and the maximum number of views that can be positioned on the sheet. These values must lie in between two and four and, naturally, the former should be smaller than the latter. Using this feature, BCFs that allow three views and use the remaining space for parts list, load table and key plan can be created – as seen in figure 4.10. Note, however, that both minimum and maximum view count have to be set to 3 in order to achieve a similar result. If you allow two or four views on the sheet, the space will be used for a view which then will overlap parts list, load table and key plan.

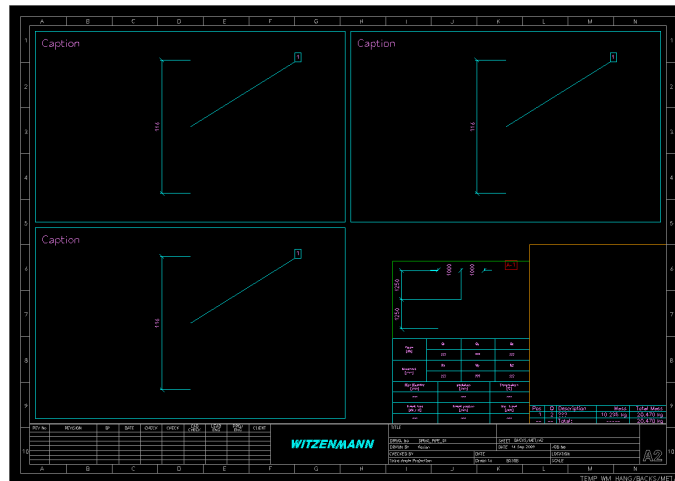


Figure 4.10 – Backing sheet with three views

4.2.5 Editing style definitions

When you finished setting the available areas' size, you may change the backing sheet's appearance in the Style tab. In case you created a new Backing Sheet Configuration File, the tab will initially be filled with the default values as shown in figure 4.11. These will be filled with the data from the file which is given in the **bcf_default_style** directive or the plugin's default values.

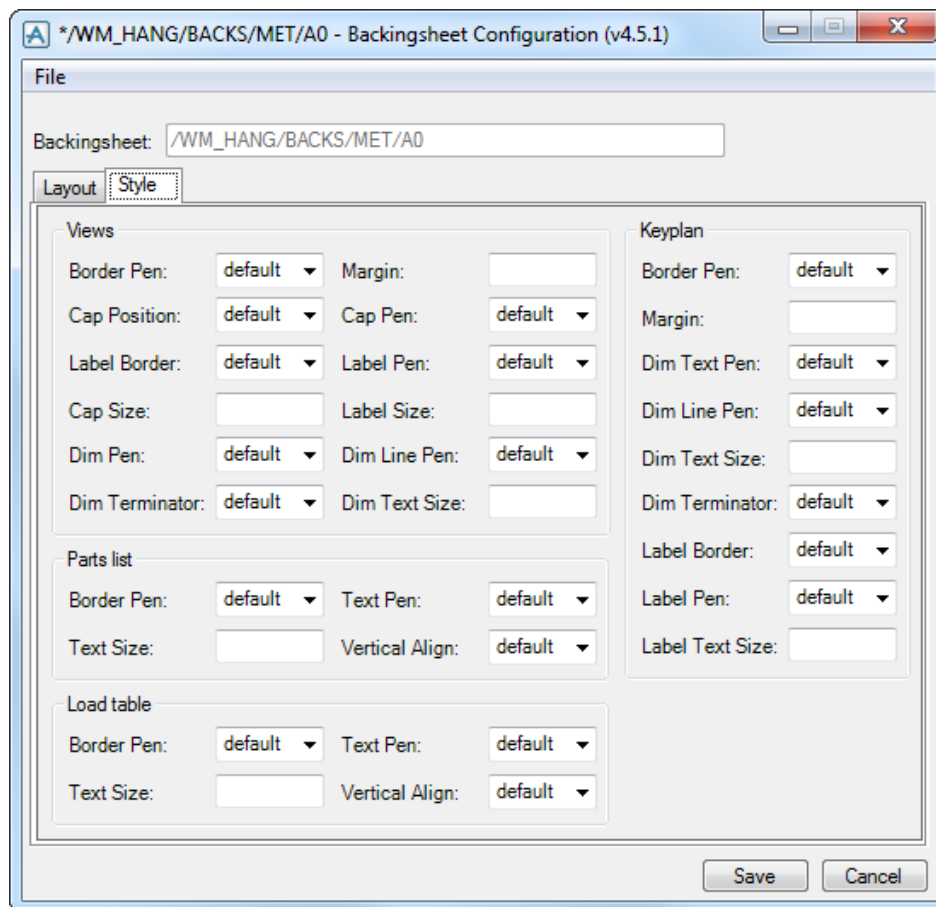


Figure 4.11 – Editing a BCF’s style definitions

The properties are spread over the four well-known areas since every area uses its own style definitions. For example, there is a border colour property, which may be applied to the views and the parts list as well. The value of a particular property may be changed independently of any other property. However, most properties have alike effects:

All pens – namely the border, label, text and dim pens – astonishingly define the pens the respective elements will be drawn with. While texts are only affected by a pen’s colour, all other pens additionally control the line’s style. Thus dotted, dashed and no border lines at all can be produced.

Margin defines the gap between two views and between a view and the view area’s bounds in millimeters. The respective text fields expect numbers between 0 and 255.

Size sets the according text’s font size in millimeters. Again, values between 0 and 255 are expected. Note that the font size of parts list and load table cannot be set to any value since it will be reduced as far as required in order to fit the given bounds.

Dim Terminator this refers to the dimension terminators. You can choose between none, arrows, obliques and dots.

As mentioned above, every property has a default value, which is either determined from the `bcf_default_style` file or set by the plugin itself. To restore a default value, simply select the `default` option in a list or clear the according text field. Note that default values will not be written into a BCF but will be determined dynamically at runtime. Thus, the appearance of

several backing sheets can be changed at once by editing the default definitions. If you want to have the default values stored in a particular BCF, you may use the tool's Import function and import the style definitions from the desired file.

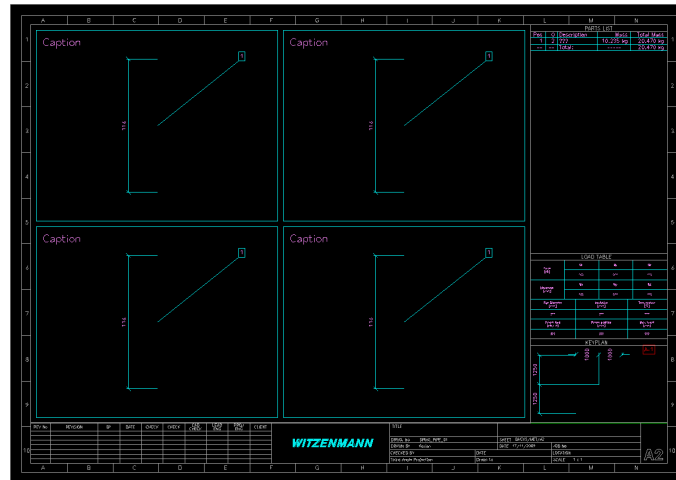


Figure 4.12 – Appearance of the default styles

4.2.6 Import

The File menu provides functions to import area and style definitions from other BCFs into the current one. While you may import area and style definitions at once, you may also import only one of them.

4.2.7 Saving

If the **bcf_dir** directive is set in *wm.ini*, the edited BCF can be saved by clicking the *Save* button. It will automatically be stored in the **bcf_dir** where the Drawing Tool can find it.

In case the directive is not set or does not point to a valid directory, you will be prompted to save the BCF manually.

Chapter 5

General tools

This chapter describes supporting tools which cannot be started directly. Instead, they are called by the several other tools.

5.1 The database browser

The database browser was designed to support you in selecting elements from the PDMS database hierarchy. It combines functionality of the Members List and the Design Explorer since it makes use of the next and previous buttons and shows the siblings of an element's parent elements. This is shown in figure 5.1.

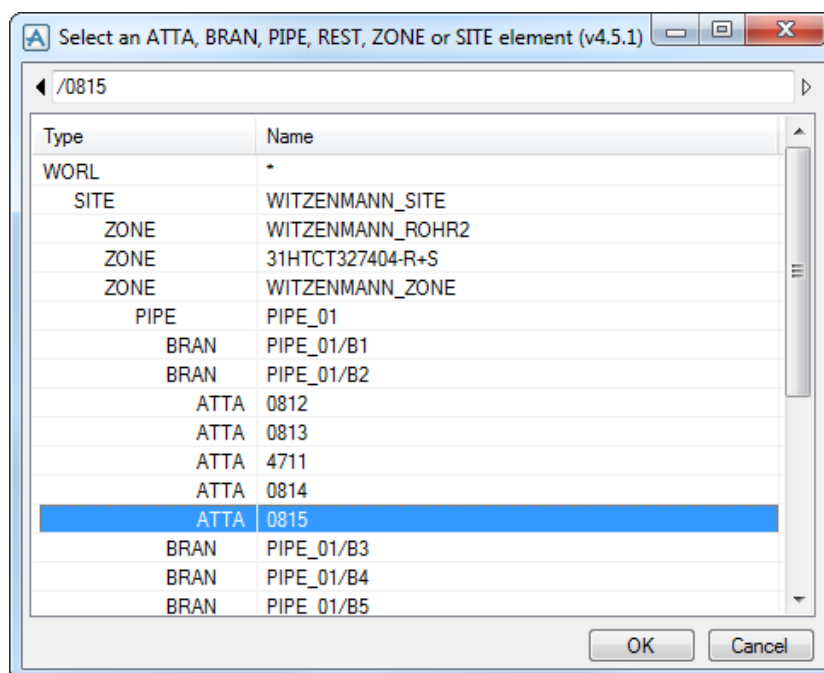


Figure 5.1 – The database browser

5.1.1 Navigation and selecting elements

Just like in the Members List, you can move through the element tree by single clicks. Furthermore, you may select the previous or the next element on the current level by pressing the ◀ or ▶ button

respectively. Additionally, an element can be selected by entering a name in the text field.

Note that navigating through the elements with the database browser does not affect the Current Element (CE).

To confirm the selection of an element, you only need to press the *OK* button. Alternatively, you can double-click¹ the element you want to select.

¹Note that E3D 2.1 doesn't support double-clicks in PML forms anymore.

Appendix

A Combined E3D 1.1 Customization File

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <UICustomizationSet xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <DefaultIcon>AvevaSharedIcons:ID_WARNING</DefaultIcon>
4   <UICustomizationFiles>
5     <CustomizationFile Name="Module" Path="design.uic" />
6     <CustomizationFile Name="SchematicExplorerAddin"
7       Path="CoreSchematicMenu.uic" />
8     <CustomizationFile Name="Project" Path="$1.uic" Optional="true" />
9     <CustomizationFile Name="SVGCompare" Path="SVGCompare.uic" />
10    <CustomizationFile Name="Cabling" Path="AVEVA.design.cabling.uic" />
11    <CustomizationFile Name="Hvac" Path="AVEVA.design.hvac.uic" />
12    <CustomizationFile Name="Supports" Path="AVEVA.design.MDS.uic" />
13    <CustomizationFile Name="Piping" Path="AVEVA.design.piping.uic" />
14    <CustomizationFile Name="Steelwork"
15      Path="AVEVA.design.steelwork.uic" />
16    <CustomizationFile Name="MessageAddin"
17      Path="MessageWindowCoreMenus.uic" />
18    <CustomizationFile Name="Laser" Path="AVEVA.design.laser.uic" />
19    <CustomizationFile Name="Integrator" Path="Integrator.uic" />
20    <CustomizationFile Name="DiagramViewer" Path="DiagramViewer.uic" />
21    <CustomizationFile Name="InstrumentationImportAddin"
22      Path="InstrumentationImportAddin.uic" />
23    <CustomizationFile Name="FlexperteDesignAddin"
24      Path="..\wittenmann\uic\FlexperteDesignAddin.uic" />
25    <CustomizationFile Name="OtherDesignAddin"
26      Path="..\other\uic\OtherDesignAddin.uic" />
27   </UICustomizationFiles>
28 </UICustomizationSet>
```

Figure A.1 – Combined *DesignCustomization.xml* for two plugins in E3D 1.1. For E3D to load the UIC files of the Flexperte and third party plugins, the environment variable “CAF_UIC_PATH” has to be set to E:\e3d-plugins\common as mentioned in section 1.5.3. Alternatively, you can also use absolute paths in lines 19 and 20 to refer to those UIC files.

B Application codes in PDMS

The following tables list the application codes available in the PDMS default installation.

B.1 Design

Table B.1 – PDMS application codes in the Design module

Code	Application
GEN	General Application
EQUI	Equipment Application
PIPE	Pipework Application
CABL	Cable Trays Application
HVACADV	HVAC Designer Application
STLWRK	Beams & Columns Application
PANEL	Panels & Plates Application
CIVIL	Walls & Floors Application
ASL	ASL Modeller Application
DTMP	Design Templates Application
CABLINGSYSTEM	Cabling System Application

B.2 Draft

Table B.2 – PDMS application codes in the Draft module

Code	Submodule	Application
GEN	User	General
ADPCORE	User	Automatic Drawing Production
EDITOR	User	AutoDRAFT
LIBMGEN	Admin	General Admin
LIBMSHEE	Admin	Sheet Administration
LIBMSYM	Admin	Symbol Administration
LIBMISO	Admin	Isodraft Symbol Administration
LIBMLAB	Admin	Label Administration
EDITORADMIN	Admin	AutoDRAFT Administration
LIBMSTY	Admin	Representation Style Administration
LIBMREP	Admin	Representation Rule Administration
LIBMDRL	Admin	Drawlist Administration
LIBMTAG	Admin	Tagging Administration

C Axes systems

The Export and Drawing tools allow the selection of axes systems which shall be used in order to set a specific attachment in a spatial context. For this purpose, the plugin searches for a site named /AXES and offers all of its zones for selection as a possible axes system.

C.1 Inner structure of an axes system

The particular axes systems can be divided in equipments and boxes, which represent the actual axes. Figure C.1 shows the axes system of the test project.

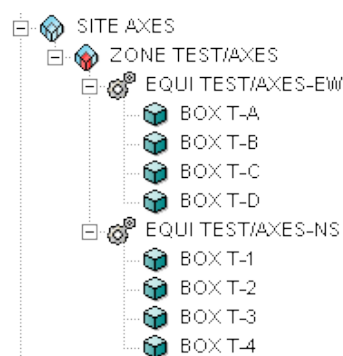


Figure C.1 – Hierarchy of the test project's axes system

Note that it is neither required to separate east-west and north-south axes by individual equipments nor to apply a special naming convention to zone, equipments or boxes. (Although the names of the boxes will be shown in the keyplan and used in the FIN files.)

C.2 Determining the closest axes

In order to determine the closest axes of a given attachment, only those axes that align with one of the main coordinate axes will be inspected. Skew axes or those that are parallel to the x-y-plane will be ignored.

In the best case, exactly one box length is less than 100 mm while the two remaining lengths are considerably greater. (This is the case for the boxes of the test axes system.)

D UDAs required for data interchange

Data Interchange between PDMS and Flexperte (and vice versa) requires User Defined Attributes (UDAs) at attachment, pipe and restraint level. They are listed in the tables [D.1](#) to [D.3](#).

Only the UDAs mentioned in these tables will be recognized at drawing production and at export and import of data.

Table D.1 – UDAs required on attachment level

Name	Type	Description
:USTDISTROD	Number	distance between single legs of a double load chain
:USTDYNLOADS	Array	dynamic loads at support point in x-, y- and z-direction
:USTEREPOR	String	
:USTFORCES	Array	static loads at support point in x-, y- and z-direction
:USTHCOMP	String	type of variable spring hanger
:USTHTYP	String	hanger type proposed by designer
:USTLDYNLOADS	Array	dynamic loads at support point in x-, y- and z-direction with respect to the point's local coordinate system
:USTLFORCES	Array	static loads at support point in x-, y- and z-direction with respect to the point's local coordinate system
:USTLMOVEMENT	Array	movements in x-, y- and z-direction, positive and negative dynamic movements at support point with respect to the point's local coordinate system
:USTMAXTRAVEL	Number	maximum movement of hanger / support
:USTMOVEMENT	Array	movements in x-, y- and z-direction, positive and negative dynamic movements at support point
:USTSCHTYP	String	calculated hanger type
:USTSRATE	Array	spring rate
:USTSTATLOAD	Number	maximum static load in z-direction
:USTSTEELELEV	Array	elevation of steel connection (one value per leg)
:USTTESTLOAD	Number	hydro test load

Table D.2 – UDAs required on pipe level

Name	Type	Description
:USTDTEMP	Number	design temperature

Table D.3 – UDAs required on restraint level

Name	Type	Description
:wzmBlockPos	Number	calculated blocking position
:wzmDTemp	Number	design temperature
:wzmDynLoads	Array	dynamic loads at support point in x-, y- and z-direction
:wzmHangType	String	hanger type
:wzmLoadType	Character	load type “C” for cold “H” for hot
:wzmMaxForces	Array	maximum forces at support point in positive and negative x-, y- and z-direction. (Sequence in the array is +x, +y, +z, -x, -y, -z.)
:wzmMaxMoves	Array	maximum movements at support point in positive and negative x-, y- and z-direction. (Sequence in the array is +x, +y, +z, -x, -y, -z.)
:wzmMaxTravel	Number	maximum movement of hanger / support
:wzmOpForces	Array	operating forces at support point in positive and negative x-, y- and z-direction. (Sequence in the array is +x, +y, +z, -x, -y, -z.)
:wzmOpMoves	Array	operating movements at support point in positive and negative x-, y- and z-direction. (Sequence in the array is +x, +y, +z, -x, -y, -z.)
:wzmSkewPull	Array	skewed pull in x- and y-direction
:wzmSRate	Array	spring rate per HANG element (upto two)
:wzmStatLoad	Number	maximum static load in z-direction
:wzmTestLoad	Number	hydro test load

E Load table layout

The plugin version 4.2.1 introduced the generation of load tables in grouped drawings. In order to achieve this, a new load table layout was designed, which can be used for both single and grouped drawings.

Used UDAs^{4.3.2}

Because of the new UDAs, there is no distinction between the POW and UST attributes, anymore. Instead, we only print the WZM attributes, which are now available on restraint level. However, we differentiate whether the attributes `:wzmOpForces` and `:wzmOpMoves` are set or not.

E.1 The load table's body

The values of the UDAs listed below are queried from the restraint. Only the pipe diameter and insulation thickness are provided by the attachment.

Table E.1 – The load table's body

Restraint	Force [kN]			Hydrotest load [kN]
	Qx	Qy	Qz	
Restraint 1	<x-Force>	<y-Force>	<z-Force>	:wzmTestLoad
Restraint 2	<x-Force>	<y-Force>	<z-Force>	:wzmTestLoad
	Movement [mm]			Skewed Pull X / Y [°]
	Wx	Wy	Wz	
Restraint 1	<x-Movement>	<y-Movement>	<z-Movement>	:wzmSkewPull
Restraint 2	<x-Movement>	<y-Movement>	<z-Movement>	:wzmSkewPull
	Pipe Diameter [mm]	Insulation [mm]	Temperature [°C]	
Restraint 1	PTOD	INTHICKNESS	:wzmDTemp	
Restraint 2	PTOD	INTHICKNESS	:wzmDTemp	
	Preset load [kN]	Preset position [mm]	Max. travel [mm]	Spring rate [N/mm]
Restraint 1	:wzmStatLoad	:wzmBlockPos	:wzmMaxTravel	:wzmSRate
Restraint 2	:wzmStatLoad	:wzmBlockPos	:wzmMaxTravel	:wzmSRate

As already mentioned, the Drawing Tool does not check whether the POW or UST attributes are present during drawing production, anymore. Instead, the final layout of the load table depends on whether the operating forces (`:wzmOpForces`) and movements (`:wzmOpMoves`) are set. Note that the output of the forces does not affect the output of the movements, so the forces may be formatted in the old UST style while the movements are printed the way the POW attributes used to be.

Forces

If the attribute `:wzmOpForces` is empty, **Qx**, **Qy** and **Qz** all show the respective force with the largest absolute value in positive and negative x-, y- and z-direction respectively. The result is identical to the former UST load table.

Force	Assignment
Qx	$\max(\text{abs}(\text{:wzmMaxForces}[1]), \text{abs}(\text{:wzmMaxForces}[4]))$
Qy	$\max(\text{abs}(\text{:wzmMaxForces}[2]), \text{abs}(\text{:wzmMaxForces}[5]))$
Qz	$\max(\text{abs}(\text{:wzmMaxForces}[3]), \text{abs}(\text{:wzmMaxForces}[6]))$

If the attribute contains some data, the Drawing Tool constructs the values for **Qx**, **Qy** and **Qz** using the former ALSTOM semantics and prints the maxima from both **:wzmOpForces** and **:wzmMaxForces**. Note that it is possible that index x_1 differs from x_2 and the load table displays the operating force in positive x-direction (index 1) together with the maximum force in negative x-direction (index 4). (This of course applies to the forces in y- and z-direction as well.)

Force	Assignment
Qx	$\text{:wzmOpForces}[x_1] (\text{:wzmMaxForces}[x_2])$
Qy	$\text{:wzmOpForces}[y_1] (\text{:wzmMaxForces}[y_2])$
Qz	$\text{:wzmOpForces}[z_1] (\text{:wzmMaxForces}[z_2])$

Movements

The movements are processed in a similar way to the forces. If the UDA **:wzmOpMoves** is empty, the values of **Wx** upto **Wz** are assembled from the respective maximum movements in positive and negative x-, y- and z-direction respectively.

Movement	Assignment
Wx	$\max(\text{abs}(\text{:wzmMaxMoves}[1]), \text{abs}(\text{:wzmMaxMoves}[4]))$
Wy	$\max(\text{abs}(\text{:wzmMaxMoves}[2]), \text{abs}(\text{:wzmMaxMoves}[5]))$
Wz	$\max(\text{abs}(\text{:wzmMaxMoves}[3]), \text{abs}(\text{:wzmMaxMoves}[6]))$

However, if **:wzmOpMoves** contains any data, the load table shows all movements.

Movement	Assignment
Wx	$\text{:wzmOpMoves}[1] (\text{:wzmMaxMoves}[1])$ $\text{:wzmOpMoves}[4] (\text{:wzmMaxMoves}[4])$
Wy	$\text{:wzmOpMoves}[2] (\text{:wzmMaxMoves}[2])$ $\text{:wzmOpMoves}[5] (\text{:wzmMaxMoves}[5])$
Wz	$\text{:wzmOpMoves}[3] (\text{:wzmMaxMoves}[3])$ $\text{:wzmOpMoves}[6] (\text{:wzmMaxMoves}[6])$

F Layout of Backing Sheet Configuration Files

Backing Sheet Configuration Files (BCF) are files which contain the used areas' size, the backing sheet referenced by the BCF and the style definitions.

BCFs are written in the Backing Sheet Markup Language (BackML), an extension of XML. XML (eXtensible Markup Language) itself is a hierarchically structured language which is used for data interchange between different programmes. Data in XML documents is stored as plain text and can be viewed and edited with a simple text editor. Figure F.1 shows one of the BCF delivered with the plugin. A BCF consists of a **sheetConfiguration** element which contains a **backingSheet** element and the four area elements **view**, **keyplan**, **loadtable** and **partslist** (case sensitive). Additionally, the root element has two attributes **minViews** and **maxViews** that control the minimum and maximum number of views allowed by this BCF. The attributes' value must lie in between two and four.

The **backingSheet** element specifies the used backing sheet's name. Note that an element with the given name must exist in the PDMS database.

Each of the four area elements, of which only the **view** element is mandatory, is cut into an **area** and a **style** element. The former describes the coordinates of the area's top left corner and its width and height. All this information must be present and contain nonnegative numbers. However, the **style** element is optional. It contains a semicolon-separated list of key-value pairs which describe the according area's appearance. The style definitions are described in detail in appendix F.2.

Note that the order in which the elements appear is irrelevant.

F.1 XML Schema

Another way to describe a XML document is using a so-called XML Schema which primarily specifies the complex elements' structure and the simple elements' value range. Moreover, with the declaration of an XML Schema or a Document Type Definition (in general a formal grammar) a *well-formed* XML document becomes *valid*. Figure F.2 shows the XML Schema the Backing Sheet Configuration Files are based on.

The BCFs generated by the Backing Sheet Configuration Tool are *valid* XML documents.

Figure F.1 – Sample Backing Sheet Configuration File

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <sheetConfiguration
3     xmlns="http://www.ib-werk.de/backml"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://www.ib-werk.de/backml
6         http://www.ib-werk.de/backml/bcf.xsd"
7     minViews="2" maxViews="4">
8     <backingSheet><![CDATA[
9         /WM_HANG/BACKS/MET/A0
10    ]]></backingSheet>
11     <view>
12         <area>
13             <x>28.0011</x>
14             <y>813</y>
15             <width>1006</width>
16             <height>750</height>
17         </area>
18     </view>
19     <keyplan>
20         <area>
21             <x>1034</x>
22             <y>131</y>
23             <width>127</width>
24             <height>68</height>
25         </area>
26     </keyplan>
27     <loadtable>
28         <area>
29             <x>1034</x>
30             <y>204.3</y>
31             <width>127</width>
32             <height>67.298</height>
33         </area>
34         <style><![CDATA[
35             border-pen : 1;
36             pen : 41;
37         ]]></style>
38     </loadtable>
39     <partslist>
40         <area>
41             <x>1034</x>
42             <y>807.063</y>
43             <width>127</width>
44             <height>596.769</height>
45         </area>
46     </partslist>
47 </sheetConfiguration>

```

Figure F.2 – XML Schema for BCFs

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <xsd:schema
3     targetNamespace="http://www.ib-werk.de/backml"
4     xmlns="http://www.ib-werk.de/backml"
5     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6     elementFormDefault="qualified">
7     <xsd:element name="sheetConfiguration">
8         <xsd:complexType>
9             <xsd:all>
10                 <xsd:element name="backingSheet" type="xsd:string" />
11                 <xsd:element name="view" type="container" />
12                 <xsd:element name="partslist" type="container" minOccurs="0" />
13                 <xsd:element name="loadtable" type="container" minOccurs="0" />
14                 <xsd:element name="keyplan" type="container" minOccurs="0" />
15             </xsd:all>
16             <xsd:attribute name="minViews" type="viewCount" />
17             <xsd:attribute name="maxViews" type="viewCount" />
18         </xsd:complexType>
19     </xsd:element>
20
21     <xsd:complexType name="container">
22         <xsd:all>
23             <xsd:element name="area" type="area" />
24             <xsd:element name="style" type="xsd:string" minOccurs="0" />
25         </xsd:all>
26     </xsd:complexType>
27
28     <xsd:complexType name="area">
29         <xsd:all>
30             <xsd:element name="x" type="length" />
31             <xsd:element name="y" type="length" />
32             <xsd:element name="width" type="length" />
33             <xsd:element name="height" type="length" />
34         </xsd:all>
35     </xsd:complexType>
36
37     <xsd:simpleType name="viewCount">
38         <xsd:restriction base="xsd:integer">
39             <xsd:minInclusive value="2" />
40             <xsd:maxInclusive value="4" />
41         </xsd:restriction>
42     </xsd:simpleType>
43
44     <xsd:simpleType name="length">
45         <xsd:restriction base="xsd:decimal">
46             <xsd:minInclusive value="0.0" />
47         </xsd:restriction>
48     </xsd:simpleType>
49 </xsd:schema>
```

F.2 Style definitions in a BCF

The appearance of the several areas can be set in their **style** elements. The syntax of style definitions follows the Cascading Style Sheets (CSS) of web design and has the aim: Separation of a document's content from its representation. The style definitions' syntax is as follows:

<property> : <value>

If multiple definitions are present in one **style** element, they have to be separated by semicola.

The supported properties and their effects are listed in the tables F.1 to F.4. Unsupported properties will be ignored. Values which do not lie in their property's value range will be replaced with the property's default value.

Table F.1 – Supported style properties in the **keyplan** element

Properties	Value range (default value)	Description
border-pen	0 - 255 (1)	Sets the pen the keyplan's border is drawn with. Pens can be taken from the range of 1 to 255. A value of 0 denotes that no border shall be drawn.
dim-font-size	1 - 255 (4)	Sets the text height for the linear dimensions in millimeter.
dim-line-pen	1 - 255 (1)	Defines the pen the linear dimensions are drawn with.
dim-pen	1 - 255 (1)	Defines the pen the linear dimensions' texts are drawn with.
dim-terminator	off, arrows, dots or obliques (obliques)	Sets the terminator between the single components of a linear dimension.
label-border-pen	0 - 255 (1)	Sets the pen that is used to draw a border around the labels. Pens can be taken from the range of 1 to 255. A value of 0 denotes that no border shall be drawn.
label-font-size	1 - 255 (4)	Defines the labels' text height in millimeter.
label-pen	1 - 255 (1)	Defines the pen the labels' text is drawn with.
margin	1 - 255 (0)	The keyplan's margin to the edges of its area in millimeter.

Table F.2 – Supported style properties in the `loadtable` element

Property	Value range (default value)	Description
<code>border-pen</code>	0 - 255 (1)	Sets the pen the loadtable's lines are drawn with. Pens can be taken from the range of 1 to 255. A value of 0 denotes that no lines shall be drawn at all.
<code>font-size</code>	1 - 255 (4)	Defines the loadtable's text height in millimeter. Note that this is not necessarily the actual, but the maximum text height, since the text height will be reduced so the text will fit the loadtable's size.
<code>pen</code>	1 - 255 (1)	Defines the pen the loadtable's text is drawn with.
<code>vertical-align</code> ^{4.2.1}	<code>bottom</code> or <code>top</code> (<code>top</code>)	Sets the vertical align of the load table. <code>bottom</code> leads to a load table positioned at the bottom of its area while <code>top</code> works the other way round.

Table F.3 – Supported style properties in the `partslist` element

Property	Value range (default value)	Description
<code>border-pen</code>	0 - 255 (1)	Sets the pen the parts list's lines are drawn with. Pens can be taken from the range of 1 to 255. A value of 0 denotes that no lines shall be drawn at all.
<code>font-size</code>	0 - 255 (4)	Defines the parts list's text height in millimeter. Note that this is not necessarily the actual, but the maximum text height, since the text height will be reduced so the text will fit the parts list's size.
<code>pen</code>	1 - 255 (1)	Defines the pen the parts list's text is drawn with.
<code>vertical-align</code>	<code>bottom</code> or <code>top</code> (<code>top</code>)	Sets the vertical align of the parts list. <code>bottom</code> leads to a parts list positioned at the bottom of its area while <code>top</code> works the other way round.

Table F.4 – Supported style properties in the **view** element

Property	Value range (default value)	Description
border-pen	0 - 255 (1)	Sets the pen the border of a single view is drawn with. Pens can be taken from the range of 1 to 255. A value of 0 denotes that no lines shall be drawn at all.
caption-align	left , center or right (center)	Defines the align of the caption containing a view's viewing direction.
caption-font-size	1 - 255 (4)	Sets the captions' text height in millimeter.
caption-pen	1 - 255 (1)	Defines the pen the captions' text is drawn with.
caption-vertical-align	bottom or top (top)	Defines the vertical align of the views' caption.
dim-font-size ^{4.3.0}	1 - 255 (4)	Defines the text height for linear and angular dimensions.
dim-line-pen	1 - 255 (1)	Sets the pen the linear and angular dimensions are drawn with.
dim-pen	1 - 255 (1)	Sets the pen the linear and angular dimensions' texts are drawn with.
dim-terminator	off , arrows , dots or obliques (obliques)	Sets the terminator between the single components of a linear dimension.
label-border-pen	0 - 255 (1)	Defines the pen that is used to draw a border around the labels. Pens can be taken from the range of 1 to 255. A value of 0 denotes that no border shall be drawn.
label-font-size	1 - 255 (4)	Defines the labels' and the linear dimensions' text height in millimeter.
label-pen	1 - 255 (1)	Sets the pen the labels' text is drawn with.
margin	0 - 255 (0)	The views' margin to the edges of their area and to the other views in millimeter.