SIEMENS

BACnet CL

Integration with OIS MP 3.0

Configuration

A6V101121669
2023-11-08

Smart Infrastructure

Copyright

Cyber Security Disclaimer "Legal Ts&Cs"

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art security concept. Siemens' portfolio only forms one element of such a concept.

You are responsible for preventing unauthorized access to your plants, systems, machines and networks which should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place. Additionally, Siemens' guidance on appropriate security measures should be taken into account. For additional information, please contact your Siemens sales representative or visit http://www.siemens.com/industrial-security.

Siemens' portfolio undergoes continuous development to make it more secure. Siemens strongly recommends that updates are applied as soon as they are available and that the latest versions are used. Use of versions that are no longer supported, and failure to apply the latest updates may increase your exposure to cyber threats. Siemens strongly recommends to comply with security advisories on the latest security threats, patches and other related measures, published, among others, under http://www.siemens.com/cert/en/cert-security-advisories.htm.

About this Document

This document cannot deal with every possible application. Depending on the OIS version, the management system or subsystem installed, the functional scope described in this document may vary or not be available at all. Further useful product information as well as application examples and supplementary information to the manuals can be found on our intranet website: https://psp.sbt.siemens.com/ (Product Support Platform).

Trademarks

Siveillance™ is a registered trademark of Siemens Switzerland Ltd.

All other product or company names mentioned in this document are trademarks or registered trademarks of their respective owners and are used only for purposes of identification or description.

Contact

If you have questions or suggestions regarding the product or this documentation, please contact our Field Support Center:

Intranet:	Siveillance Technical Support
Internet:	Global Customer Support
Email:	I BT Siveillance Support

Training Courses

Siemens Building Technologies provides training courses for all products. For Siemens employees, please see SBT Academy. For external customers, please contact your local Siemens product representative.

Edition: 2023-11-08

Document ID: A6V101121669

Table of Contents

1 Safety

1.1 Target Readers

This document provides instructions for the following target groups:

Target readers	Qualification	Activity	Condition of the product
Installer	Technical training for electrical installations.	Installs the product, individual components of the product or replacement parts.	Components of the product are not yet installed or need to be replaced or modified.
Operational startup personnel	Technical training for electrical installations. Training on the product is necessary.	Places the product into operation for the first time or changes the existing configuration.	The product is installed but not yet configured, or the existing configuration is to be changed.
Service personnel	Technical training for electrical installations and software installation.	Checks the product at regular intervals to ensure good working order, to service the device or system and to repair or expand and upgrade the system.	The product is already in use and requires servicing.

1.2 Work Safety Information

- Follow all instructions in this guide.
- Keep this document for reference.
- Always include this document with the product.
- Make sure to back up all data before updating the software to avoid data loss.

1.3 Meaning of the Symbols

Tips and information

NOTICE	
	Malfunction or data loss may result

2 Purpose of the Document

This document describes how to configure the BACnet CL with OIS 2.6.1.

The configuration with BACnet in OIS Tools Web is explained on the basis of the connection to GMA Manager and Siveillance Control. If another type of management system is used, the values may differ.

3 History

Version	Changes
0.0.8	New callback function shell_command
0.0.9	New functionality: COV subscriptions
0.1.1	See chapter Example of Creating a Configuration.
0.1.2	See chapter *Example of how to Use a Timer in the Event Callback Function [➙ 82]*.
0.1.3	Detailed information on configuration using OIS Tools Web. See chapter Configuration Using OIS Tools Web.
ois-cubacnet1.12.5.12	New parameter support_date_time

4 Introduction

The Python package for the CL is provided in the ois *clbacnet* package.

The application-specific configuration of the csv files for this CL has to be done in the workspace. The csv files in the workspace describe the available point types, their events and orders. These configuration files can (and should) be reused for other projects using the same type of BACnet devices.

The *DpList.xls* file defines the available data points and the mapping of the BACnet data points from the BACnet Cu-Handler to the data points of the management system.

Special Features

- **Orders with priority:**
  Some BACnet objects support a priority array for orders. In that case, the priority of the order needs to be determined. See chapter *Order Definition (order_def.csv)* [➙ 45].
- **Definition of several order reactions for one order:**
  It is possible to set several BACnet properties at different BACnet objects as a result of one order. A delay between the setting of the individual properties is supported. See chapter *Order Definition (order_def.csv)* [➙ 45].
- **Reading additional BACnet properties for the event**:
  Additional BACnet properties can be read in the event handler using the callback function *get_unit_attribute*. See chapter *get_unit_attribute* [➙ 53].
- **Execution of a shell command as a result of an event:**
  Shell commands can be executed in the event handler using the callback function *shell_command*. See chapter *shell_command* [➙ 53].
- **Conversion of state numbers into corresponding state names:**
  The state value of a multi-state object can be converted into the corresponding text using the callback function *get_state_text*. See chapter *get_state_text* [➙ 54].
- **Support of own callback functions:**
  The event handler can call your own event callback functions. See chapter *Example of the Implementation of an Event Callback Function* [➙ 81].
- **Example of how to use a timer:**
  Timers can be used to execute a function after a given delay, for example to send an event after a delay. See chapter *Example of how to Use a Timer in the Event Callback Function* [➙ 82].
- **COV support:**
  Performs cyclic COV subscriptions and handles COV notifications. See chapter *COV Subscriptions* [➙ 39].
- **Support of the same CL for several Cu-Handlers:**
  The same CL can be used for several Cu-Handlers. For GMA Manager 8.x the point names must be unique (for example, use a prefix or suffix in the point name for each Cu-Handler). This restriction does not exist for other management systems.

5 Location of the Configuration Files

The BACnet-specific configuration files are located in a subfolder of the **plc** unit in the workspace.

Example:

Name of the BACnet unit = **BACNETEXAMPLE**

Name of the CL = **ois_clbacnet_example**

The following files are the only files to be adapted for an application-specific BACnet CL.

If you want to connect only with a GMA Manager 8.x, the folders *vualoxerv* and *vuoisexplorer* can be deleted.

```
BACNET_EXAMPLE/etc/import/DPList.xls
BACNET_EXAMPLE/etc/subscribe_cov.csv
plc/etc/ois_clbacnet_example/__init__.py
plc/etc/ois_clbacnet_example/configuration/order_def.csv
plc/etc/ois_clbacnet_example/configuration/
point_type_def.csv
plc/etc/ois_clbacnet_example/configuration/
text_def_order_names.csv
plc/etc/ois_clbacnet_example/configuration/
text_def_vu_point_types.csv
plc/etc/ois_clbacnet_example/connectors/vualoxerv/
event_def.csv
plc/etc/ois_clbacnet_example/connectors/vualoxerv/
order_map.csv
plc/etc/ois_clbacnet_example/connectors/vualoxerv/
point_type_map.csv
plc/etc/ois_clbacnet_example/connectors/vualoxerv/
vu_point_type_map.csv
plc/etc/ois_clbacnet_example/connectors/vuoisexplorer/
vu_point_type_map.csv
plc/etc/ois_clbacnet_example/connectors/vusis/event_def.csv
plc/etc/ois_clbacnet_example/connectors/vusis/order_map.csv
plc/etc/ois_clbacnet_example/connectors/vusis/
point_type_map.csv
plc/etc/ois_clbacnet_example/connectors/vusis/
vu_point_type_map.csv
```

NOTICE	
!	**Encoding for German umlauts** All csv configuration files have to be encoded in ISO8859-1 if German umlauts are used.

The values of the attributes defined in the configuration files can use the string formatting described in chapter *String Formatting* [→ 56].

The *__init__.py* file starts the CL.

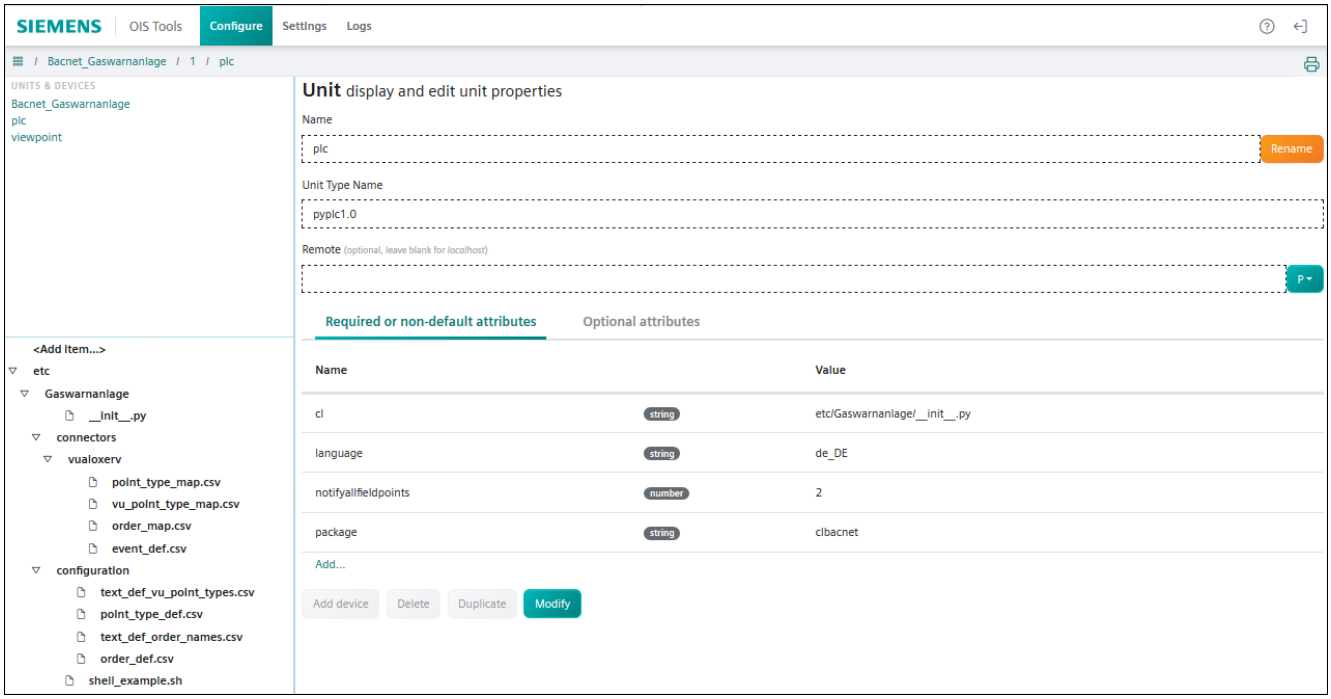
The subsystem configuration file *DPList.xls* defines the available OIS data points and is located in the */etc/import* directory of the BACnet Cu-Handler unit.

The *subscribe_cov.csv* file is only required if **Change of Value** reporting (COV) is used to define the required subscriptions.

The configuration files that are common for all connectors are located in the *configuration* directory. In general, these are the following files:

- *order_def.csv*
- *text_def_vu_point_types.csv*

- *point_type_def.csv*
- *text_def_order_names.csv*



The configuration files that are specific for a connector (such as **vusis** or **vualoxerv**) are located in the subdirectory with the same name as the connector.

For example, the files for connector **vusis** are located in the *connectors/vusis* directory. In general, these are the following files:

- *vu_point_type_map.csv*
- *order_map.csv*
- *event_def.csv*
- *point_type_map.csv*

The *event_def.csv* file for connector **vusis** is often reused by other connectors. The **vualoxerv** and **vuvantage** connectors support a default mapping of the message codes to the Vu-Handler dependent attributes for this purpose.

To simplify matters, you can copy the files to a computer on which MS Excel is installed.

6 Configuring the Subsystem Using OIS Tools

6.1 Configuring OIS Tools

6.1.1 Setting Up OIS Tools

1. Start the web browser on the OIS server. Use a compatible web browser such as Mozilla Firefox or Google Chrome.

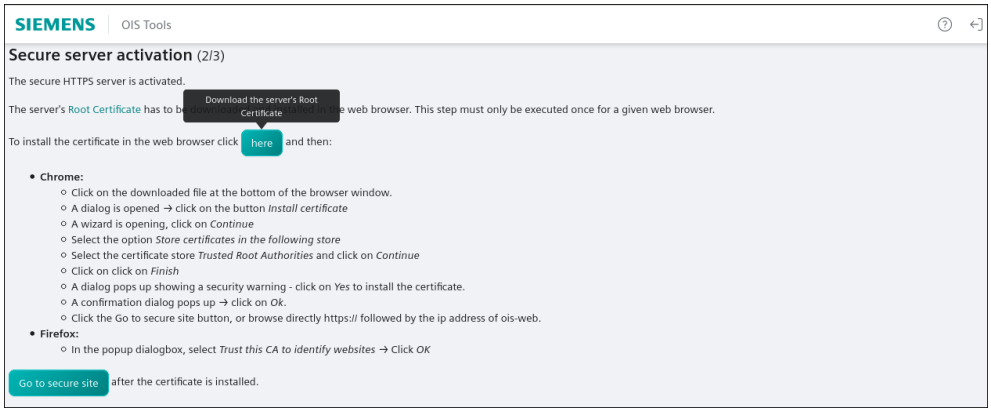
The graphics below were created using the Mozilla Firefox web browser.

2. Enter the host name in the address bar or via the local loopback adapter **127.0.0.1**.

   ⇨ The following web browser window opens:

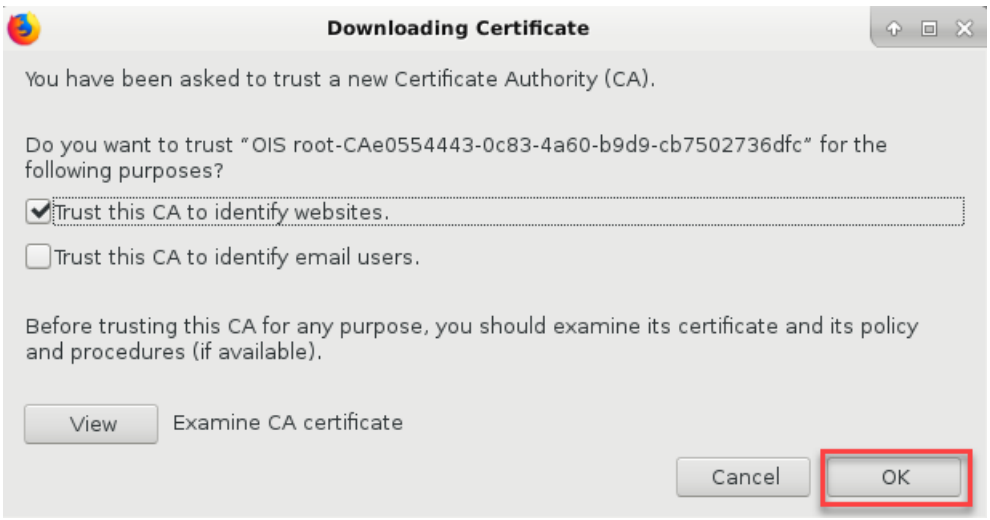
3. Click **Generate**.

   ⇨ The following web browser window opens:

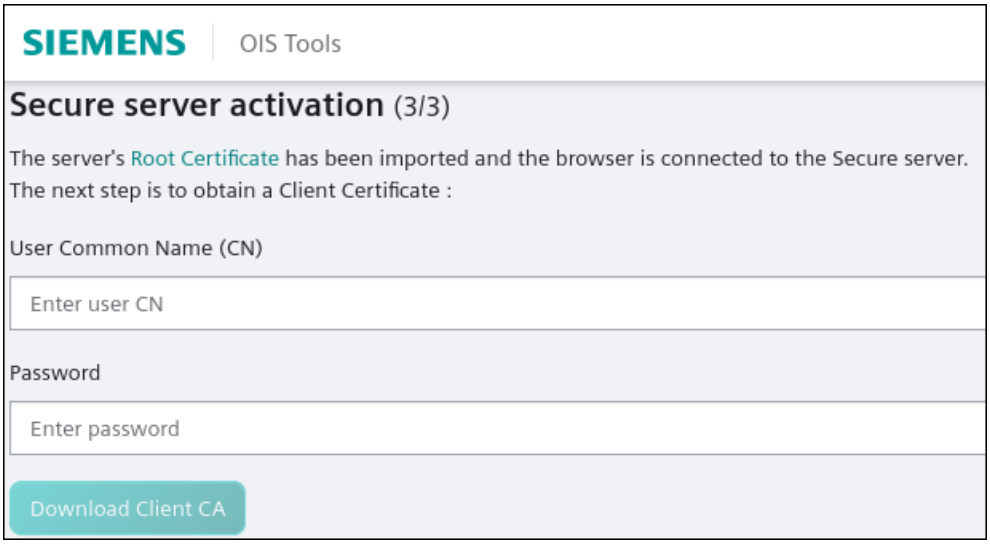


4. Click **here**.

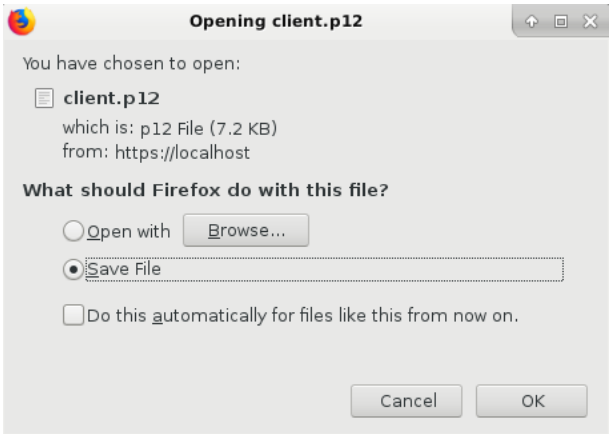
   ⇨ The following dialog box opens:



5. Mark the **Trust this CA to identify websites** check box.
6. Click **OK**.
7. Click **Go to secure site**.
8. Enter the OIS user name in the **User Common Name (CN)** text field.
9. Enter a password in the **Password** text field.

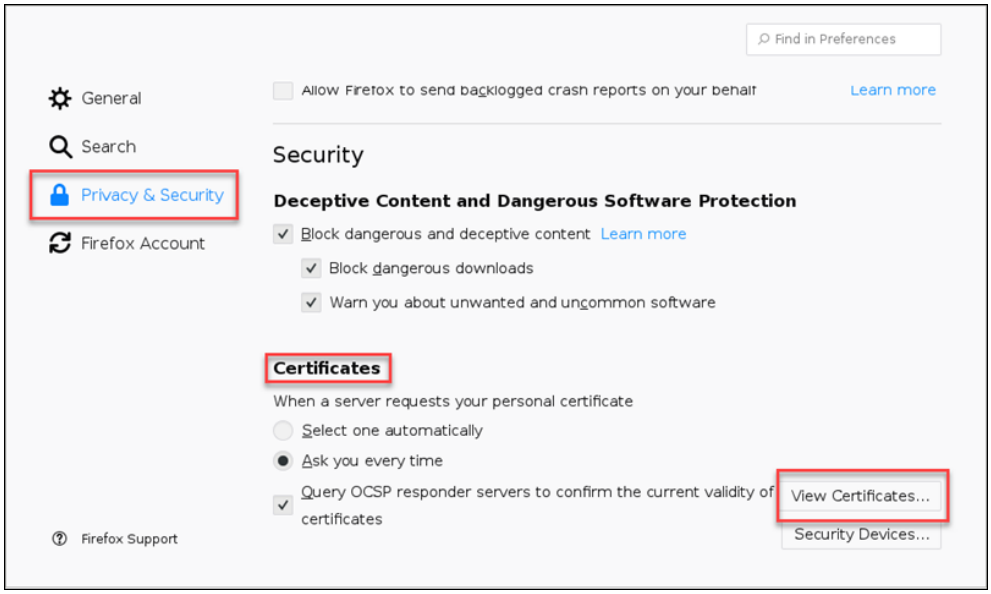


10. Click **Download Client CA**.
    ⇨ The following dialog box opens:

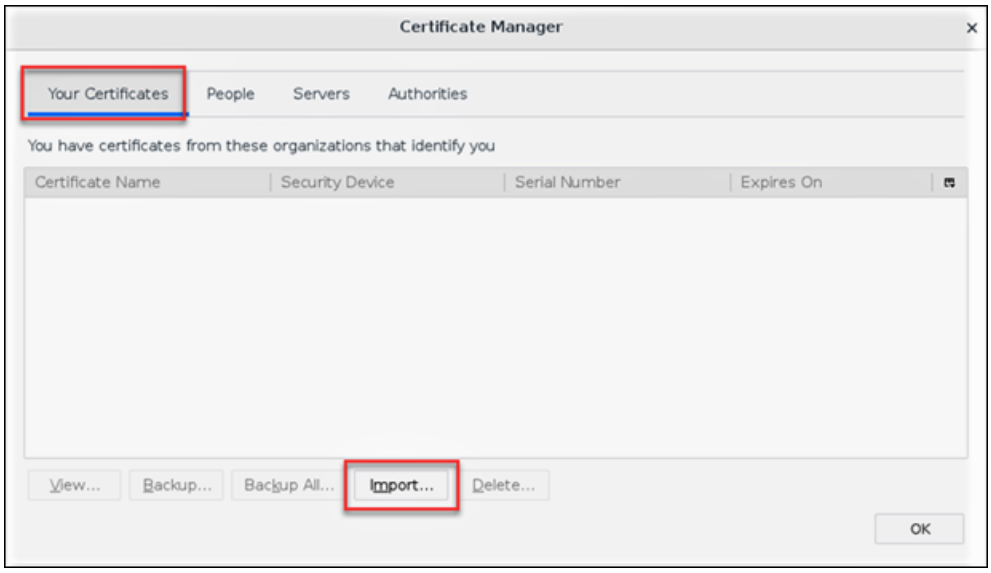


11. Mark the **Save File** option.

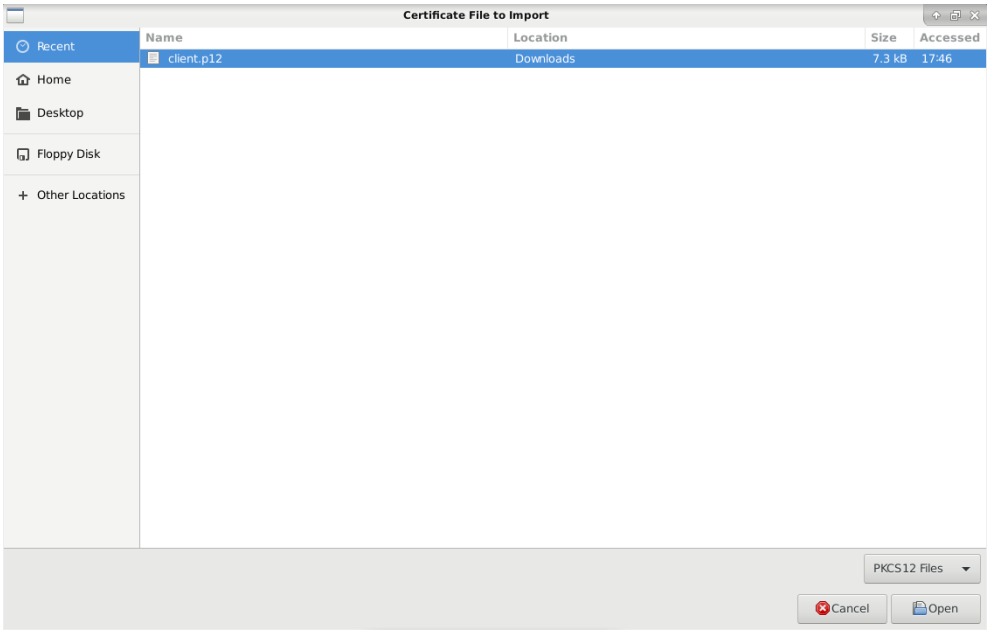
12. Click **OK**.
    ⇨ The **client.p12** certificate is saved.
13. In the web browser toolbar, click the **Menu** icon.
14. Select **Options**.
15. Click **Privacy & Security**.



16. Scroll down to the **Certificates** section.
17. Click **View Certificates**.
    ⇨ The following dialog box opens:



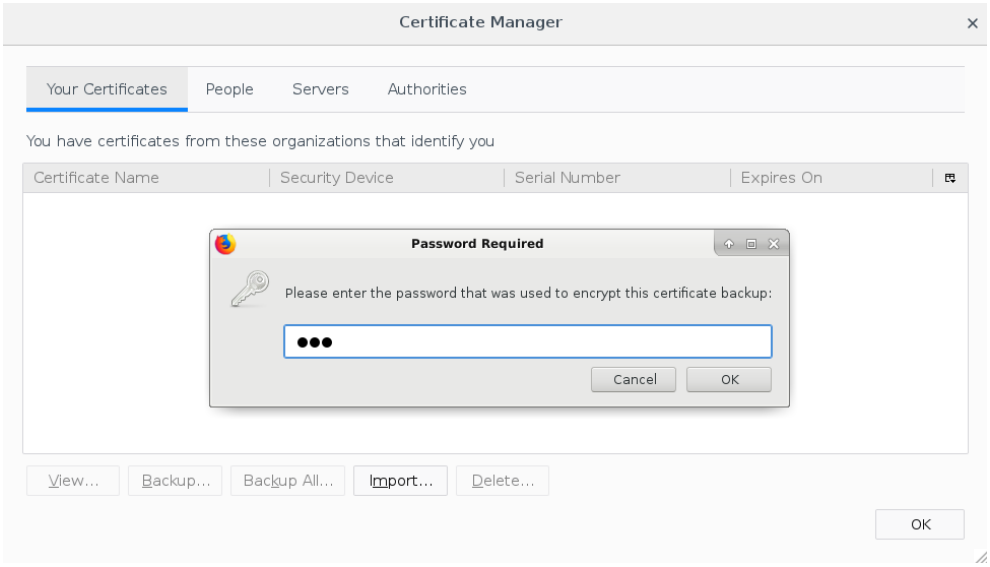
18. Select the **Your Certificates** tab and click **Import...**.
    ⇨ The following dialog box opens:



19. Select the saved **client.p12** certificate and click **Open**.

Use the password you defined for the user when creating the certificate.

20. Enter the password and click **OK**.



21. Click **OK** again.

The certificates cannot be loaded until the browser is restarted.

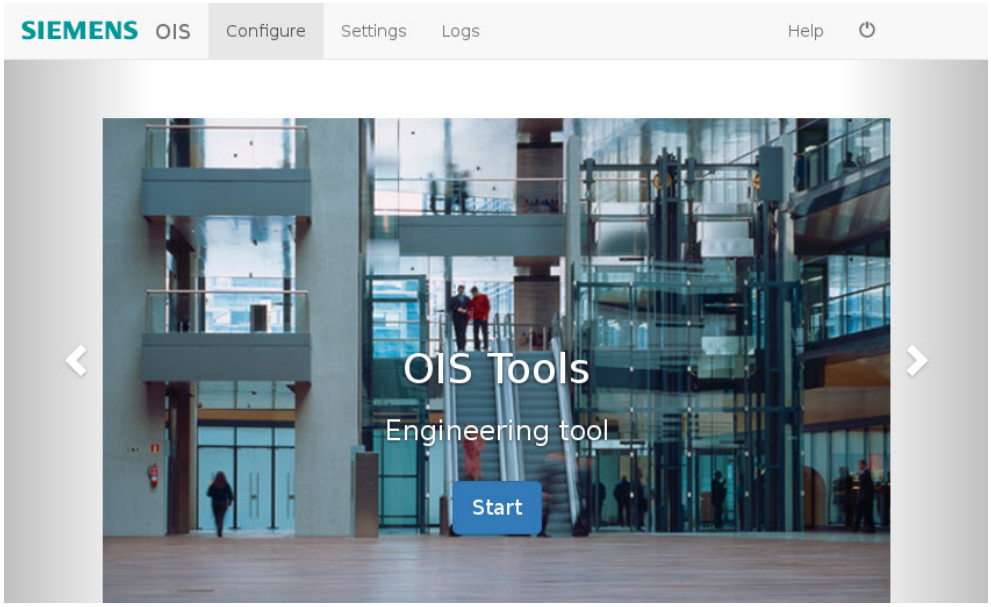
22. Close the web browser and restart it.

6.1.2 Starting OIS Tools

▷ You have set up **OIS Tools**.

1. Launch the web browser.

2. If you launch the web browser on the OIS server, enter the following IP address in the address bar: **[hostname]/#/home**
3. If you connect to OIS from a remote computer, enter the IP address of the target computer in the address bar: **https://[hostname]/#/home**
4. Click **OK** and **Go to secure site**.
   ⇨ The following web browser window opens:



5. Click **Start**.

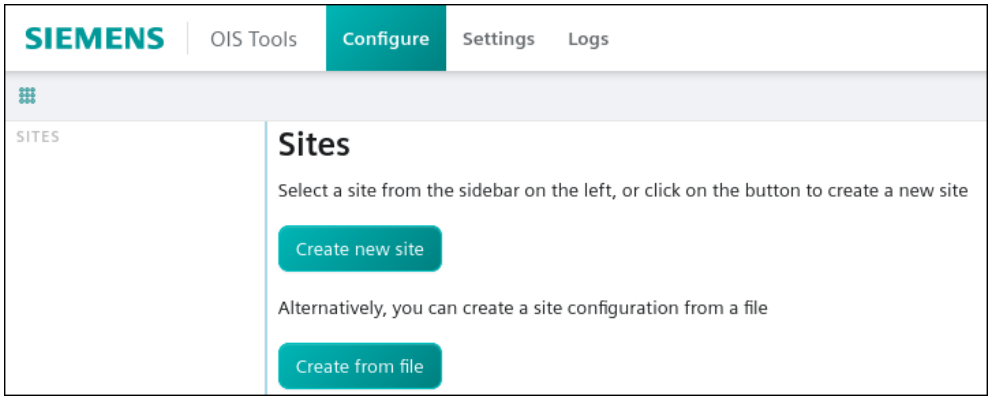
⇨ This website is now always the entry point in the OIS configuration.

6.2 Configuring Sites

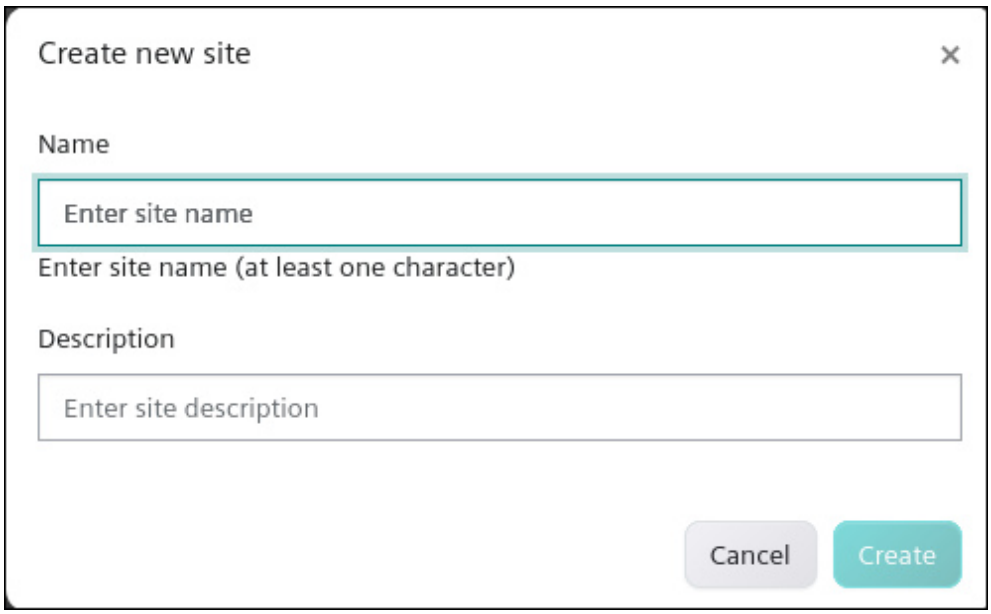
6.2.1 Creating a Site

Proceed as follows to create a new site:

1. Select the **Configure** tab.
   ⇨ The following dialog box opens:



2. Click **Create new site**.
   ⇨ The following dialog box opens:



3. Enter a name in the **Name** text field (This can be the name of the customer or location, for example BACnet_Gaswarnanlage) and click **Create**.

The site name must not contain spaces. Use hyphens or underscores instead of spaces.

4. Click **Submit**.
5. Continue with the upload of the license file.

6.2.2 Migrating the Site

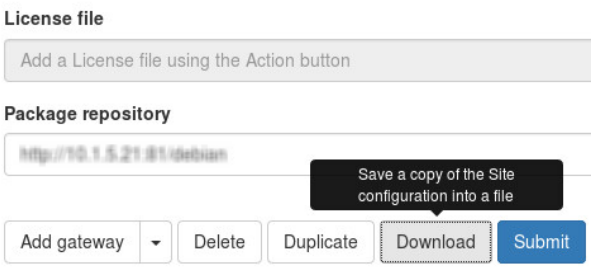
You can create the site alternatively using a configuration file. Perform the following steps to migrate the configuration file from the previous OIS version to OIS MP 3.0.x :

- Back up the site in OIS.
  See chapter *Backing Up the Site* [→ 20].
- Upload the site backup to OIS MP 3.0.x using OIS Tools.
  See chapter *Uploading the Site Backup* [→ 21].

6.2.2.1 Backing Up the Site

Proceed as follows to back up the site:

1. Click the **Site** tab in OIS Tools.



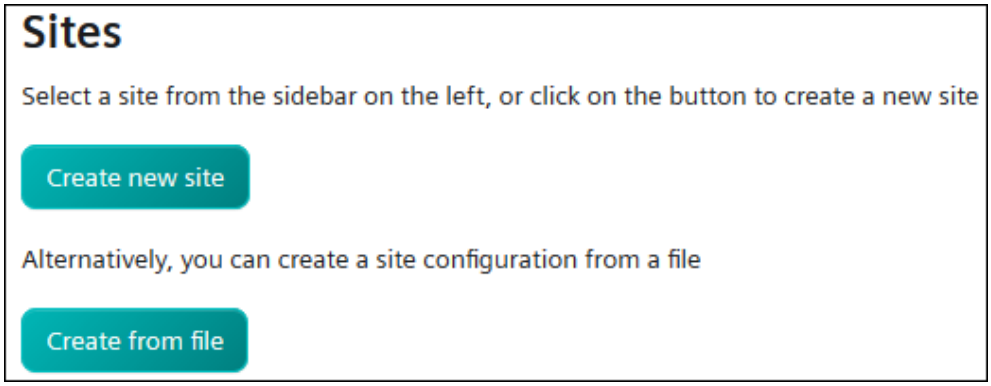
2. Select **Download** to save the site.
   ⇨ The file is saved in **Sitename.tgz** format.
3. Copy this file to a data medium.

⇨ The backup is ended.

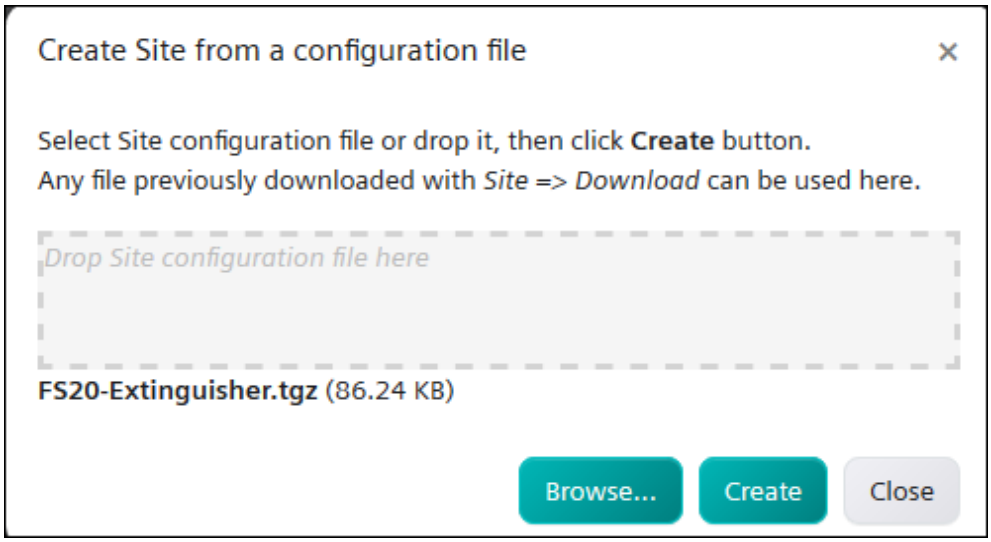
6.2.2.2 Uploading the Site Backup

Proceed as follows to upload the saved site to OIS MP 3.0.x:

1. Copy the saved **Sitename.tgz** file to the OIS MP 3.0.x computer from a console. The most appropriate target directory is **/home/ois**.
2. Change to OIS Tools.
3. Select the **Configure** tab.
   ⇨ The following dialog box opens:



4. Click **Create from file**.



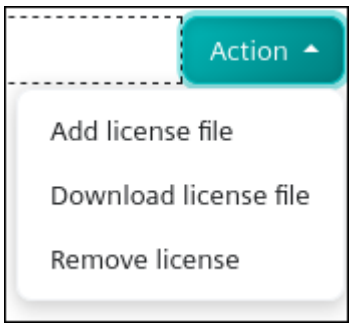
5. Click **Browse…** to select the required file.
6. Click **Create** to upload the selected file.
   ⇨ The file is uploaded.
7. Click **Close**.
   ⇨ The site has been created.
8. Check the data and files uploaded (*.xml configuration data) to make sure they are complete.

6.3 Uploading the License File

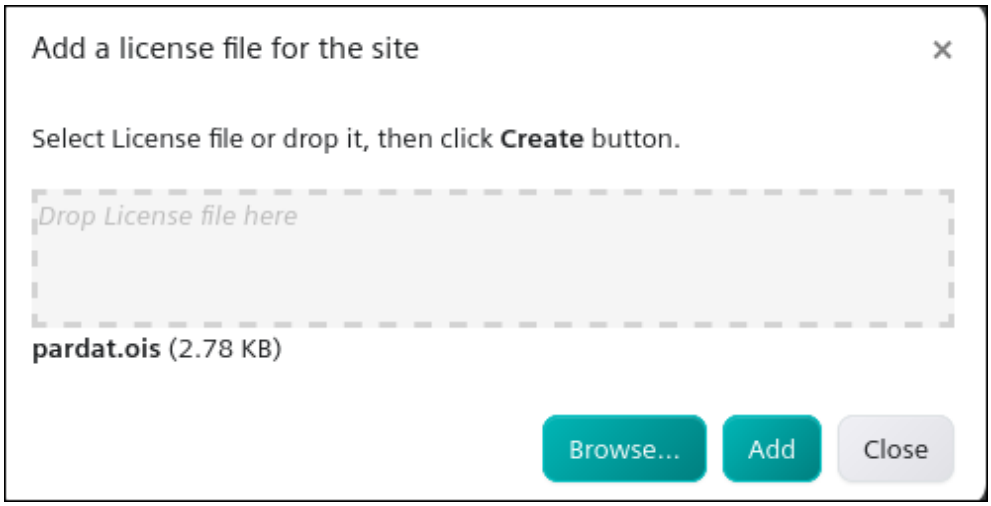
▷ The **pardat.ois** license file is available.

▷ The gateway name is configured in the **pardat.ois** license file.

1. Select **Add license file** in the **Action** drop-down list under **License file** in the newly created site.



2. Click **Browse a**nd select the **pardat.ois** license file.



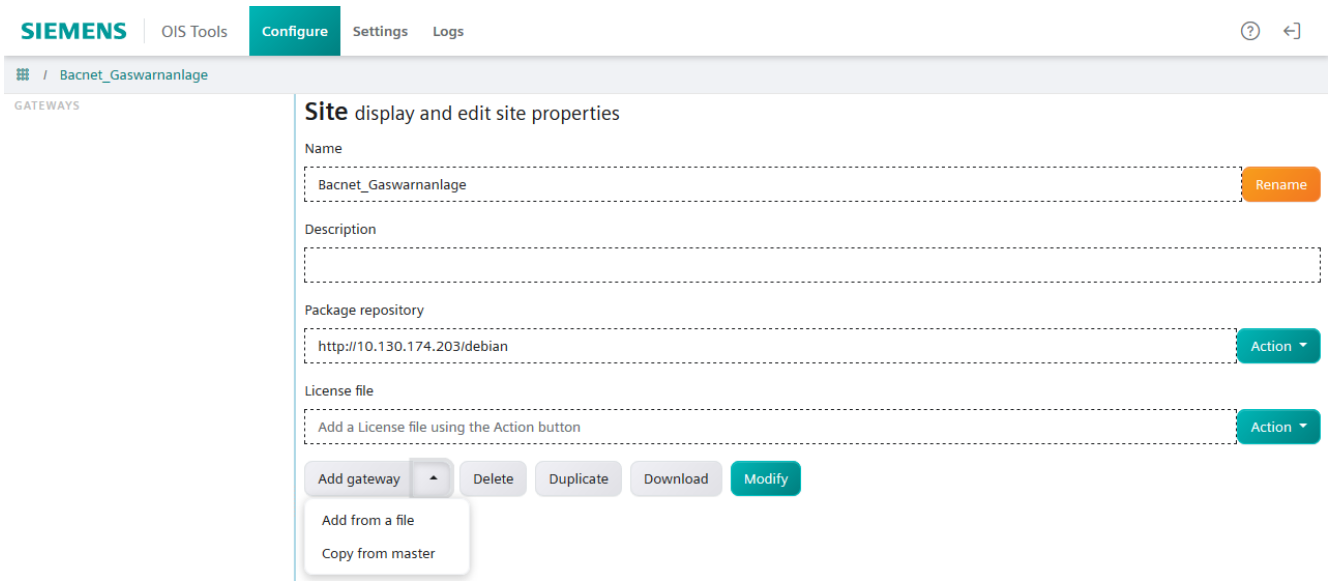
3. Click **Add**.
4. Wait until the **pardat.ois** license file has been added.
5. Click **Close**.
6. If required, enter a description in the **Description** text field.

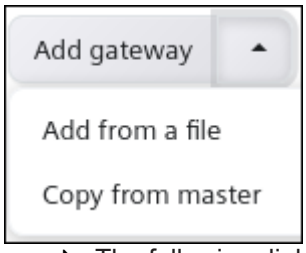
⇨ The **pardat.ois** license file is uploaded.

6.4 Configuring a Gateway

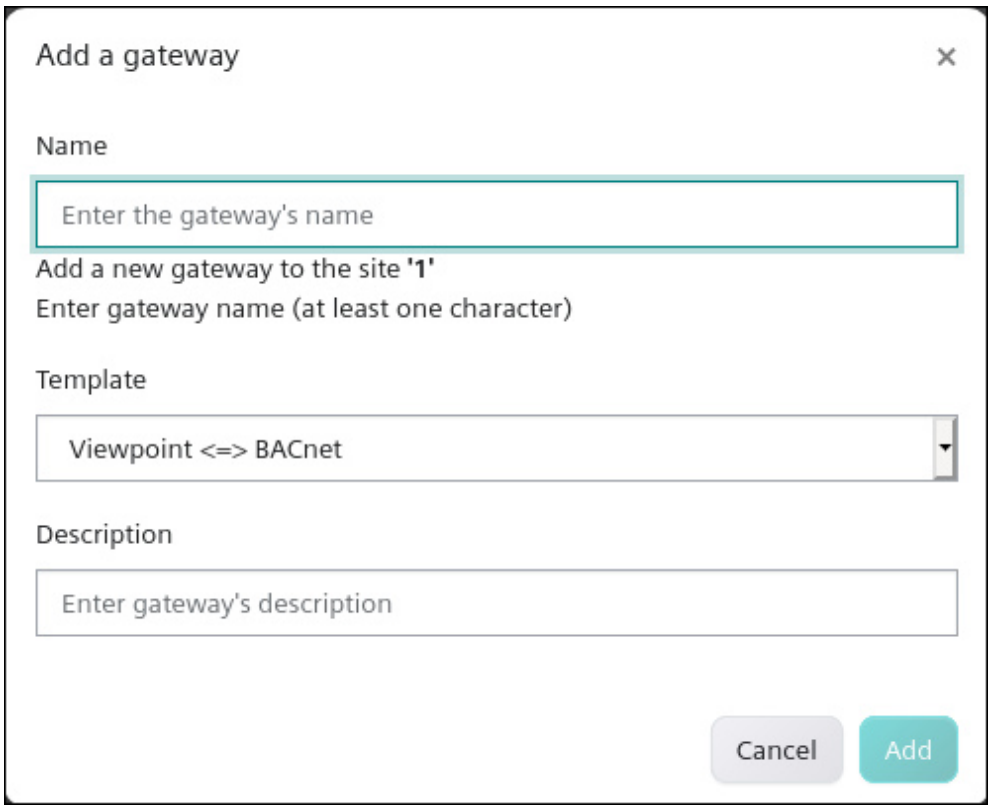
6.4.1 Creating a Gateway

1. Select **Add gateway** on the newly created site to add a gateway.





➪ The following dialog box opens:



2. Enter a number in the **Name** text field (for example 1 for MA1).
3. In the **Template** drop-down list, select Viewpoint <=> BACnet .
4. Click **Create**.
5. Enter a description in the **Description** field.

6. Click **Submit**.

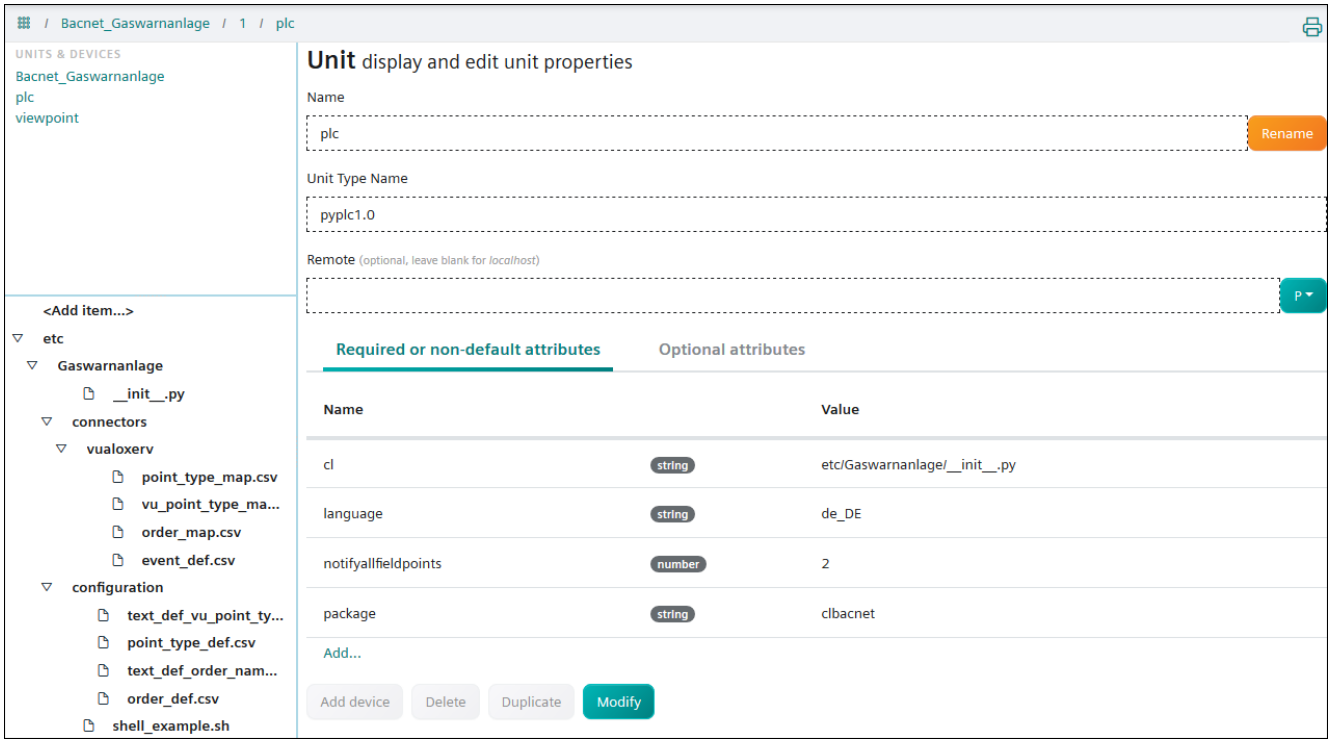
⇨ The gateway with all necessary units has been created.

6.4.2 Configuring the PLC Unit

PLC is the abbreviation of Programmable Logic Controller. An OIS gateway contains only one PLC. The PLC distributes messages between all the units involved based on the logical configuration of the OIS gateway.

1. In the left pane, click the **plc** unit you have created.

⇨ The following dialog box opens:



2. Check the configuration of the **plc** unit.

Field	Description
Required or non-default attributes	Overview of all attributes which are either required or optional and set to a value other than default value.
Optional attributes	Overview of all optional attributes

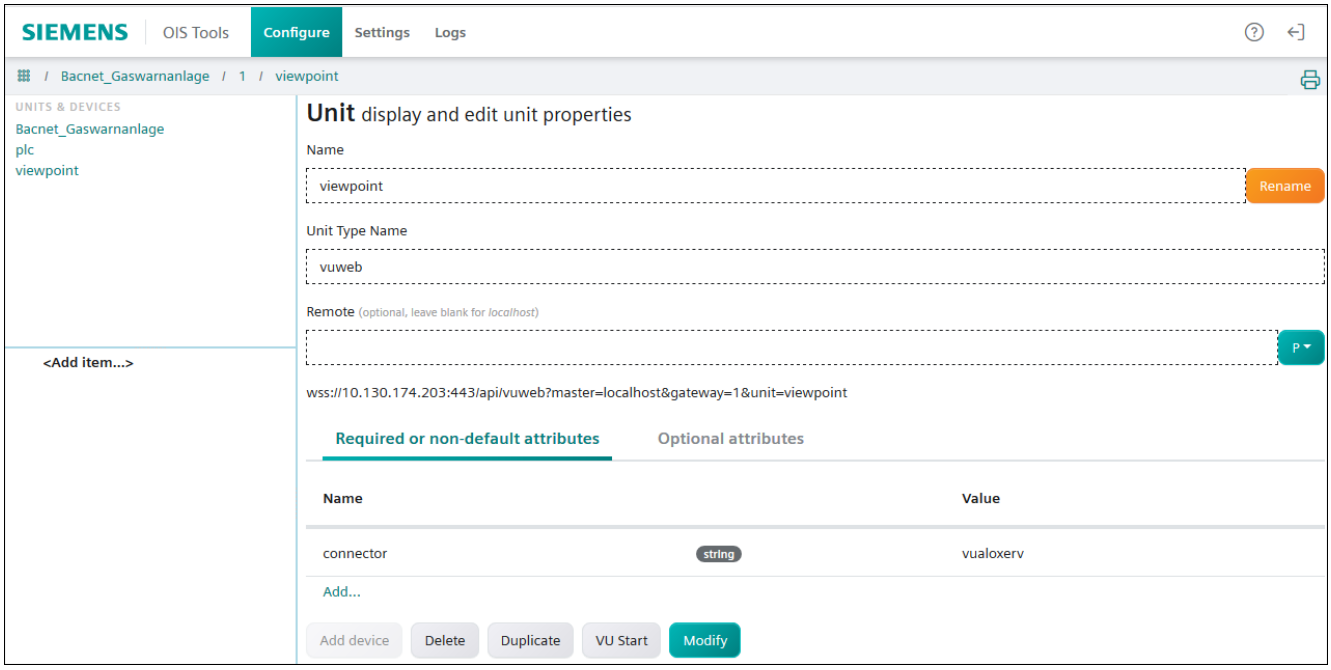
3. If necessary, change the attribute for **language** by clicking **Value**.

Attribute	Description
package	The **package** attribute defines the list of CL packages to be started. The clbacnet package is used for the BACnet Cu-Handler.
language	The **language** attribute defines the language setting for the language-dependent elements. The following languages are available: **de_DE (default)** German in Germany **en_GB** English in the UK Other languages are not available.

Attribute	Description
cl	Start file of the application-specific BACnet CL relative to the directory of the **plc** unit. For example, enter *etc/Gaswarnanlage/__init__.py* for a CL in a subdirectory named *Gaswarnanlage*.

6.4.3 Configuring the Vu-Handler Unit

1. In the left pane, click viewpoint unit you have created.



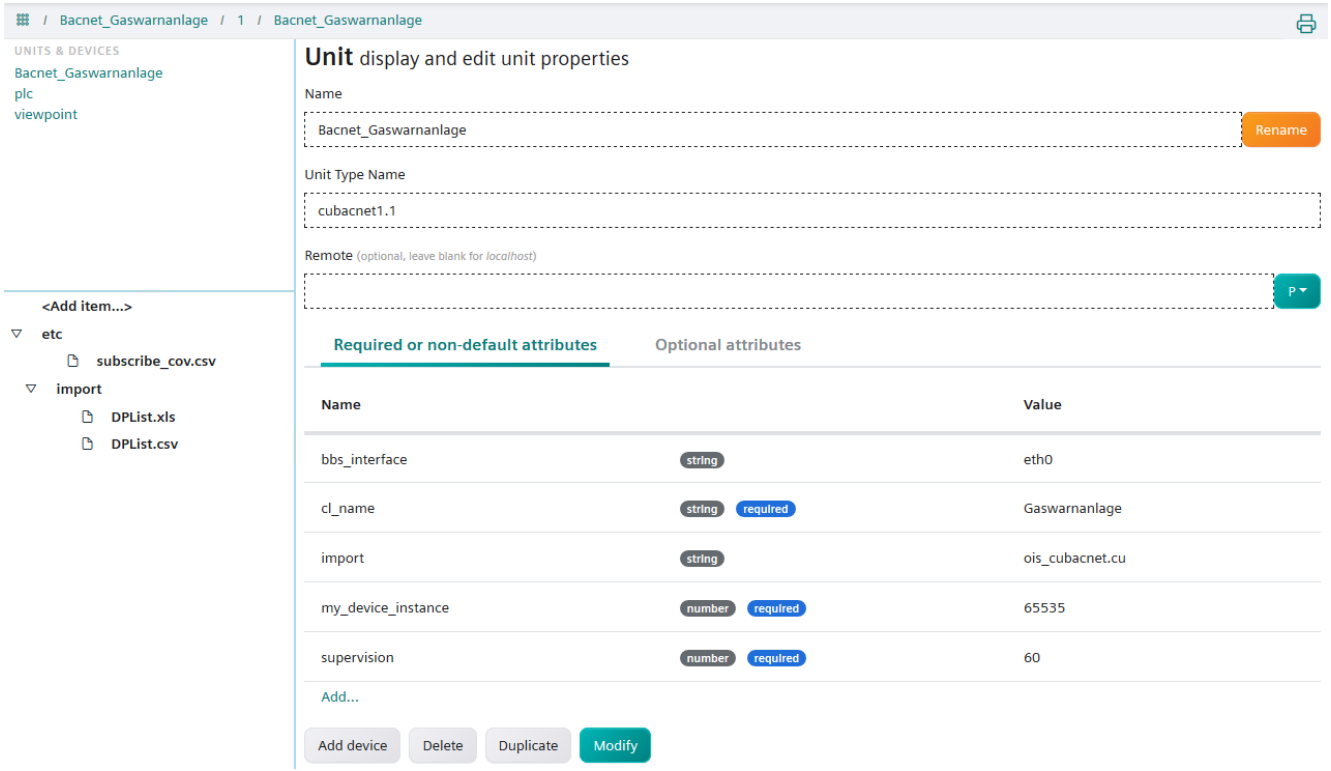
2. Use the following table to check if the necessary attributes are configured according to requirements.

Field	Description
Name	The name of the unit
Unit Type Name	The name of the unit type
Remote	Enter the `<IP address:port>` of the target computer here if the Vu-Handler is to be installed on a separate PC. For a distributed environment, update the IP address with the Vu-Handler host IP. This is the SSH URL for connecting to a remote host on which the unit is running. The SSH URL is a character string in the format [user@] host [:port], where: • user = the user name (optional, default is ois) • host = the IP address or hostname of the remote host • port = the TCP port (optional, default is 22)
Required or non-default attributes	Overview of the required attributes
Optional attributes	Overview of the optional attributes

3. The **encoding** parameter is initialized with **iso-8859-1** for the correct display of German umlauts. It is not necessary to change the attributes of Vu-Handler **sisvu**.

6.4.4 Configuring the Cu-Handler Unit

1. In the left pane, click the Cu handler unit name you have created (BACNETGASWARNANLAGE).

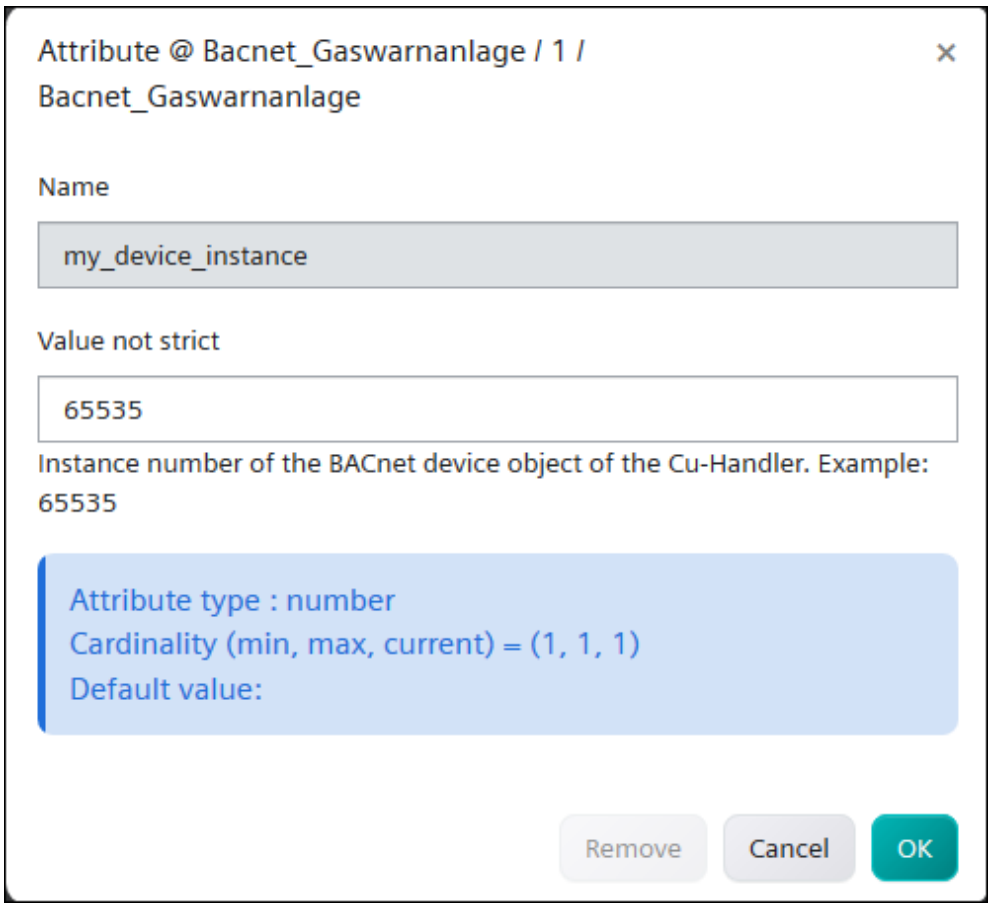


2. Use the following table to check if the necessary attributes are configured according to requirements. **NOTICE! Click “string” or “number” to configure the attribute.**

Field	Description
Name	The name of the unit
Unit Type Name	The name of the unit type
Remote	Enter the **[IP address]:[port]** of the target computer here if the Cu handler is to be installed on a separate PC.
Required or non-default attributes	Overview of all attributes which are either required or optional and set to a value other than the default value.
Optional attributes	Overview of all optional attributes

3. To change the **my_device_instance** attribute, click **number** in the line of the **my_device_instance** attribute in the **Attribute** group field.

   ⇨ The following dialog box opens:



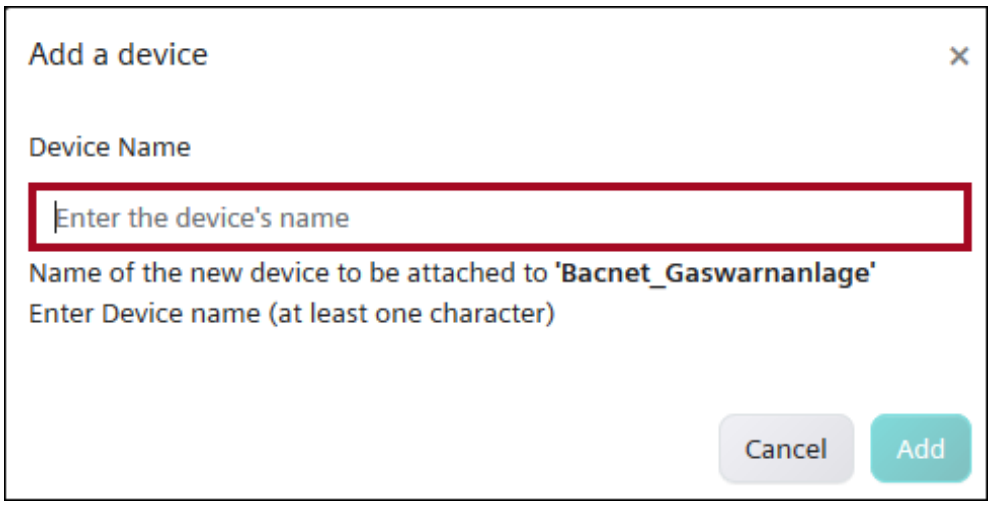
4. Enter the value of the **my_device_instance** attribute in the **Value not strict** text field.
5. Click **OK**.

⇨ The unit has been configured.

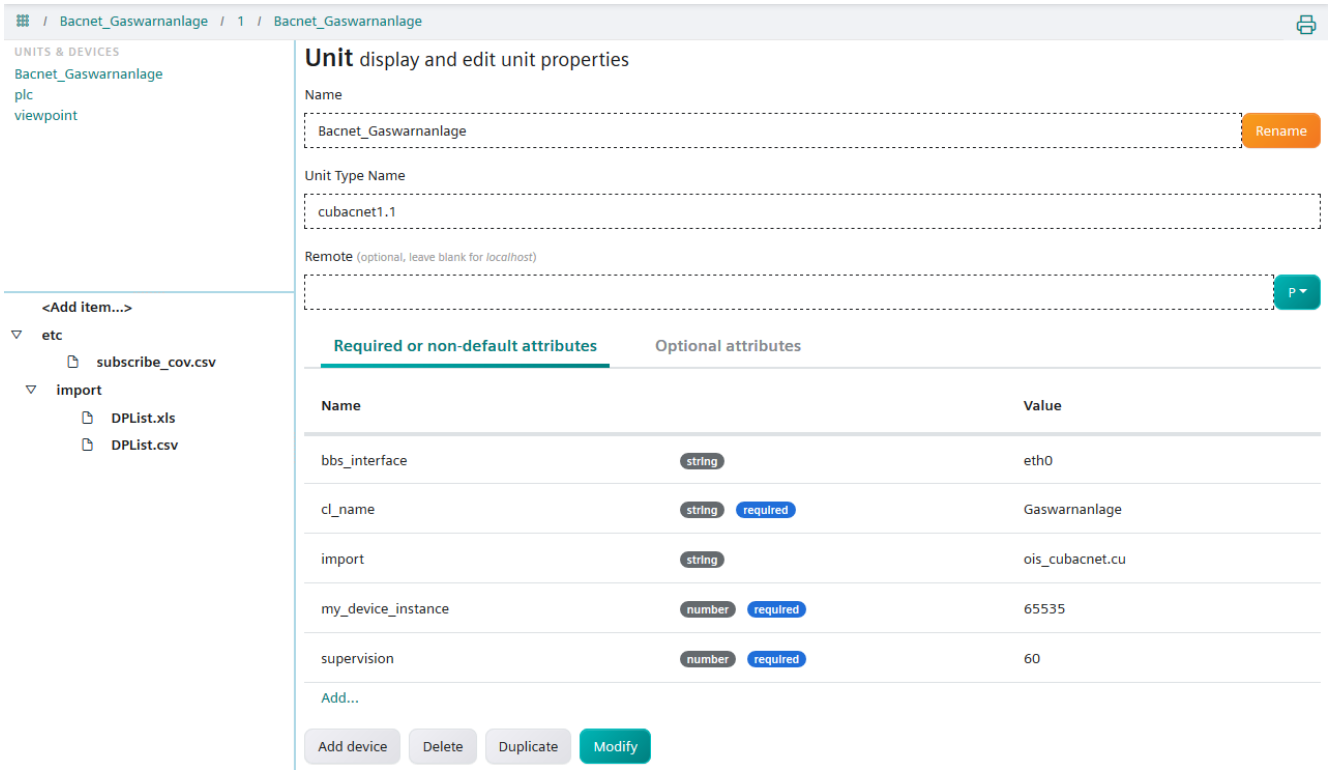
6.4.5 Configuring the Cu-Handler Device Time Server

NOTICE	
!	Only the BACnet device that is configured as a time server needs to be considered here, for example, device identifier of the time server: 1. This is only required if the BACnet device supports the time server functionality.

1. Click **Add device** in the window for configuring the Cu-Handler.



2. Enter a name in the **Device Name** text field (for example 1).
3. Click **Add**.
4. Click the device you have created (for example 1).
   ⇨ The following dialog box opens:

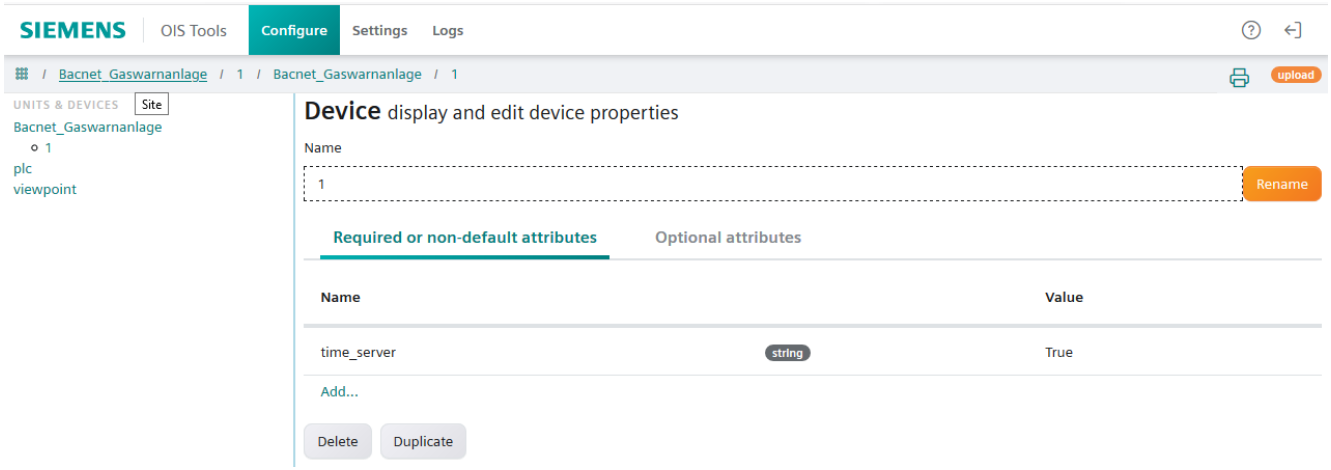


5. Set the value of the **time_server** attribute to **True**.
   See chapter BACnet Configuration.
6. Click **Submit**.

6.4.6 Configuring the Cu-Handler Device

1. In the left pane, click the Cu-Handler Device you have created (1).

NOTICE	
!	If multiple Transliner control panels are to be connected, an additional device must be added for each further control panel. To do this, click **Add device** in the Cu-Handler unit.



2. Use the following table to check if the necessary attributes are configured according to requirements.

Field	Description
Name	The name of the unit
Required or non-default attributes	Overview of all attributes which are either required or optional and set to a value other than default value.
Optional attributes	Overview of all optional attributes

3. Enter values for the following required attributes in the **Required or non-default attributes** tab.

Attribute Name	Description
ip	The IP address of the alarm panel
public_certificate	The public key (the *public lines*) of the SSH server certificate stored on the alarm panel. See chapter Copying the Public Lines of the Server Certificate.
ssh_user	The SSH user name configured on the alarm panel. See chapter Configuring SSH Users in NoxConfig.

4. Edit further optional attributes if required in the **Optional attributes** tab.
5. To save the changes, click **Submit**.

6.4.7 Importing the Configuration File

OIS Tools requires the subsystem configuration file with the data point definitions. See chapter *Subsystem Configuration File (DPList.xls)* [➙ 37].

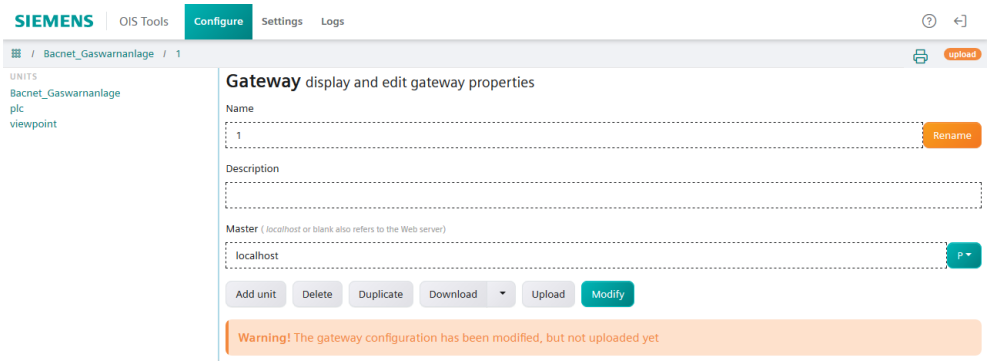
The name of the DPList.xls / DPList.csv must not contain spaces or umlauts.

1. Click the gateway you have created.
2. Select the required Cu handler.

3. In the navigation tree, click the **import** directory.
   ⇨ The following dialog box opens:
4. Select the **Upload** tab.
   ⇨ The following dialog box opens:
5. Click **Browse**.
6. Select the required file.
   – Or...
7. Drag the file into the upload window.
8. Click **Open**.
9. Click **Upload**.
   ⇨ The selected file is loaded and the progress is displayed.
10. Click **Close**.
    ⇨ The file has been uploaded.
11. Click **Submit**.
    ⇨ The Cu-handler configuration is saved.

6.4.8 Uploading the Gateway Configuration

1. Click the gateway you have created (1).
   ⇨ A warning message and the orange upload symbol indicate that an upload is scheduled.



2. Click **Upload**.
3. Click **Start**.
   ⇨ The upload is performed.

If no or an insufficient license file is available, OIS will run for 96 hours and then be automatically exited.

   ⇨ OIS checks the licenses on the basis of the configured units (Cu-Handler and Vu-Handler).
   ⇨ If one or more configured files are not registered in the license file, a warning message is displayed indicating the missing license(s). OIS will run for 96 hours and then stop automatically. OIS has to be started again manually and will run for another 96 hours until a proper license has been obtained.
   ⇨ If the upload was performed correctly, a **Success** message is displayed.

4. Click **Close**.

⇨ All necessary software packages are installed during initial start-up.

All the files required for the BACnet integration are created through the data import.

- In the first step, the OIS data points are created based on the subsystem configuration file and mapped with the Vu data points. See chapter *Subsystem Configuration File (DPList.xls)* [➙ 37]. The files that were created are in the directory of the Cu unit: */var/lib/ois/localhost/<gateway name>/<unit name>/etc*.
- In the second step, the data point lists for the Vu-Handler are created. The files that were created are in the *etc* directory of the Vu unit.

Step 1

The first step is dependent on the Vu-Handler type.

The XML file with the OIS data points that was created is useful for troubleshooting. Here you can check how the Cu, OIS, and Vu data points are mapped to one another.

The files that were created are in the directory of the unit: */var/lib/ois/localhost/<gateway name>/<unit name>/etc*.

For gateway 1 unit BACNET_GASWARNANLAGE, this is the */var/lib/ois/localhost/1/BACNET_GASWARNANLAGE/etc* directory.

With Vu-Handler SIS

- The subsystem configuration file is read.
- The *ois_dp_<unit_name>_vusis.xml* file is created.

With Vu-Handler Siveillance Control

- The subsystem configuration file is read.
- The *ois_dp_<unit_name>_vualoxerv.xml* file is created.

With Vu-Handler Siveillance Control Pro

- The subsystem configuration file is read.
- The *ois_dp_<unit_name>_vuvantage.xml* file is created.

Step 2

This applies to all Vu-Handlers

- The *vu_dp-<gateway_name>-<component_name>-<vu_unit_name>-<unit_name>.xml* file with the Vu data points is created for each component. This is the default format of a file that is imported into a management system.

  Example of the file name:

  – Vu-Handler SIS: *vu_dp-1-sisvu-sisvu-BACNET_GASWARNANLAGE.xml*
  – Vu-Handler Siveillance Control, Siveillance Control Pro: *vu_dp-1-C_BACNET_GASWARNANLAGE_2-VU_1-BACNET_GASWARNANLAGE.xml*
  – The files that were created are in the directory of the Vu-Handler unit: */var/lib/ois/localhost/<gateway name>/<vu_unit name>/etc*.
  – For gateway 1 Vu unit **sisvu**, this is the */var/lib/ois/localhost/1/sisvu/etc* directory.

With Vu-Handler SIS

GMA Manager 8.x requires a specific format for the Vu data point list.

The files that were created are in the directory of the Vu-Handler unit.

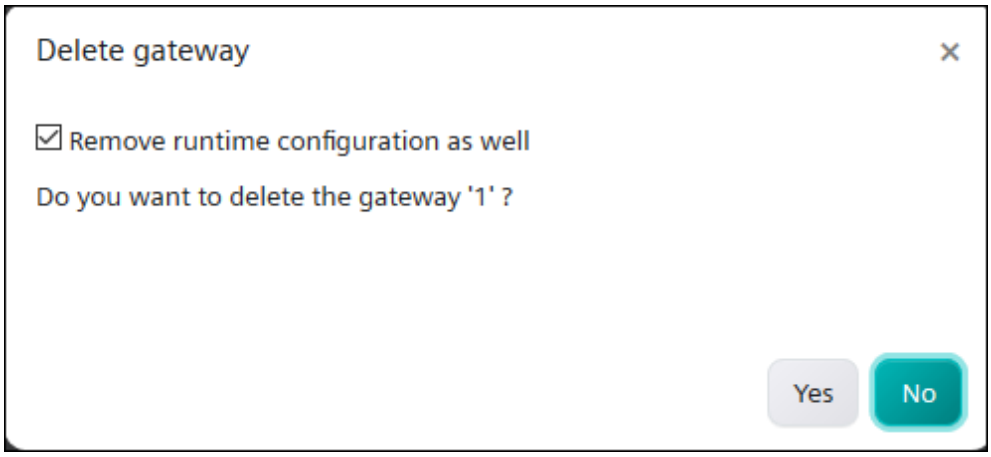
The *vu_dp_<vu_unit_name>_BAC.at* file for GMA Manager 8.x is created with all data points of the BACnet system.

6.4.9 Deleting the Gateway Configuration

The gateway configuration can be deleted in OIS Tools and the corresponding runtime folder.

1. Click the required gateway (for example 1).

2. Click **Delete**.
   ⇨ The **Remove runtime configuration as well** check box is marked by default. If the runtime folder is to be maintained or is no longer available (for example because the computer is no longer in use), then the **Remove runtime configuration as well** option should be deselected.



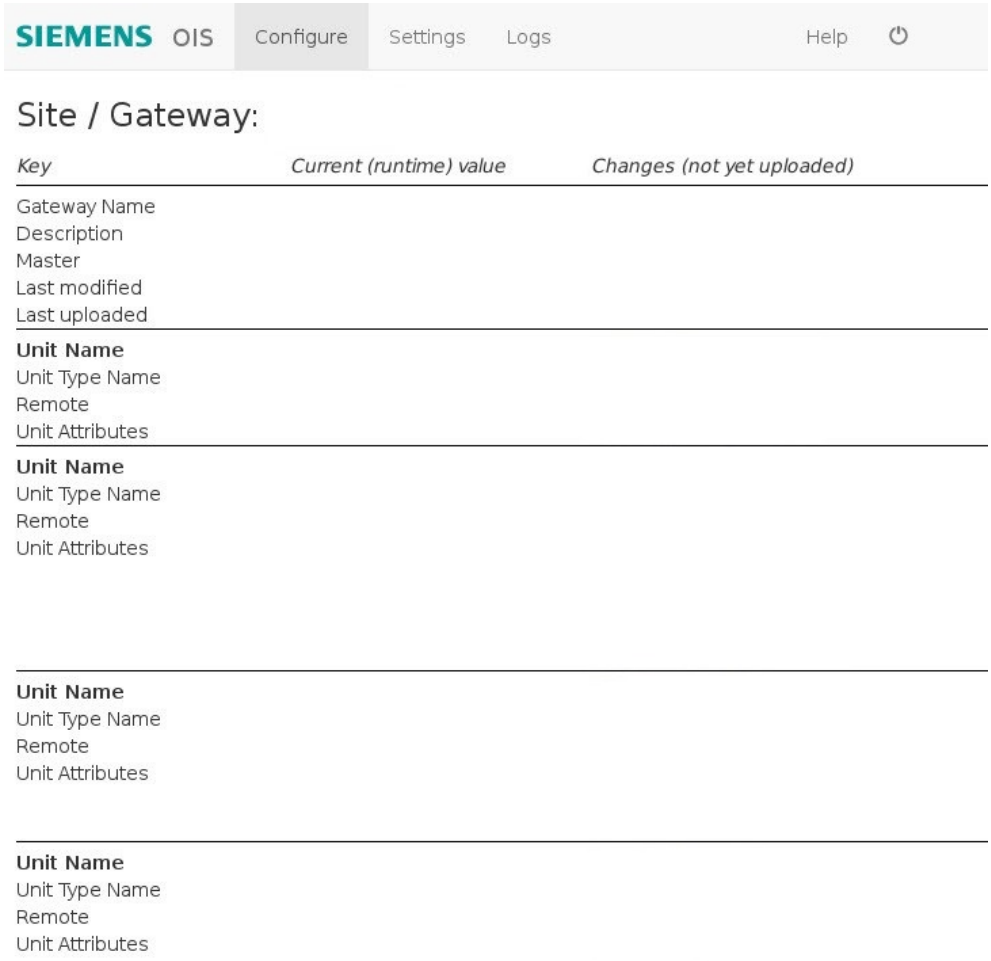
3. Click **Yes**.

⇨ The gateway configuration is deleted.

6.4.10 Printing the Gateway Configuration

After you have created the gateway, the print symbol is displayed in the top right-hand corner.

- Click the **Print gateway** symbol.
  ⇨ An overview of the gateway configuration is displayed.



Key	Name of the attribute/element
Current (runtime) value	Value of the attribute / element that was configured. This value may differ from the value that was uploaded.
Changes (not yet uploaded)	Value of the attribute / element since the last time it was uploaded. This value may differ from the configured value if the changes have not yet been uploaded.

You can directly call up the gateway configuration in the web browser and print it out: **https://[host]/#/print/[site name]/[gateway]**.

6.5 Creating the Data Points File

Once the configuration has been uploaded, OIS does not automatically create the subsystem data points file for integration in Siveillance Control.

Perform the following steps to generate the subsystem data points file:

▷ **OIS Tools** is open.

1. Click the Vu-Handler in the left-hand window pane (viewpoint).
2. Click **VU Start**.
3. Wait until the **UPDATE_END** appears.
4. Click **Stop**.

⇨ The subsystem data points file is generated.

6.6 Transferring Data Points File from OIS to Windows Computer

Perform the following steps to transfer the subsystem data points file from OIS to the Windows computer.

▷ The subsystem data points file is available.

▷ If there are multiple subsystem data points files with the same name, ensure they have a unique name before importing them into Siveillance Control.

1. Navigate to the following location of the OIS machine:
   /var/lib/ois/localhost/[gateway name]/[vu-handler unit name]/etc
2. Copy the file using **WinSCP** or USB stick to the Siveillance Control client computer.

7 Configuration Files in the Workspace

7.1 Relationship between the Config Files

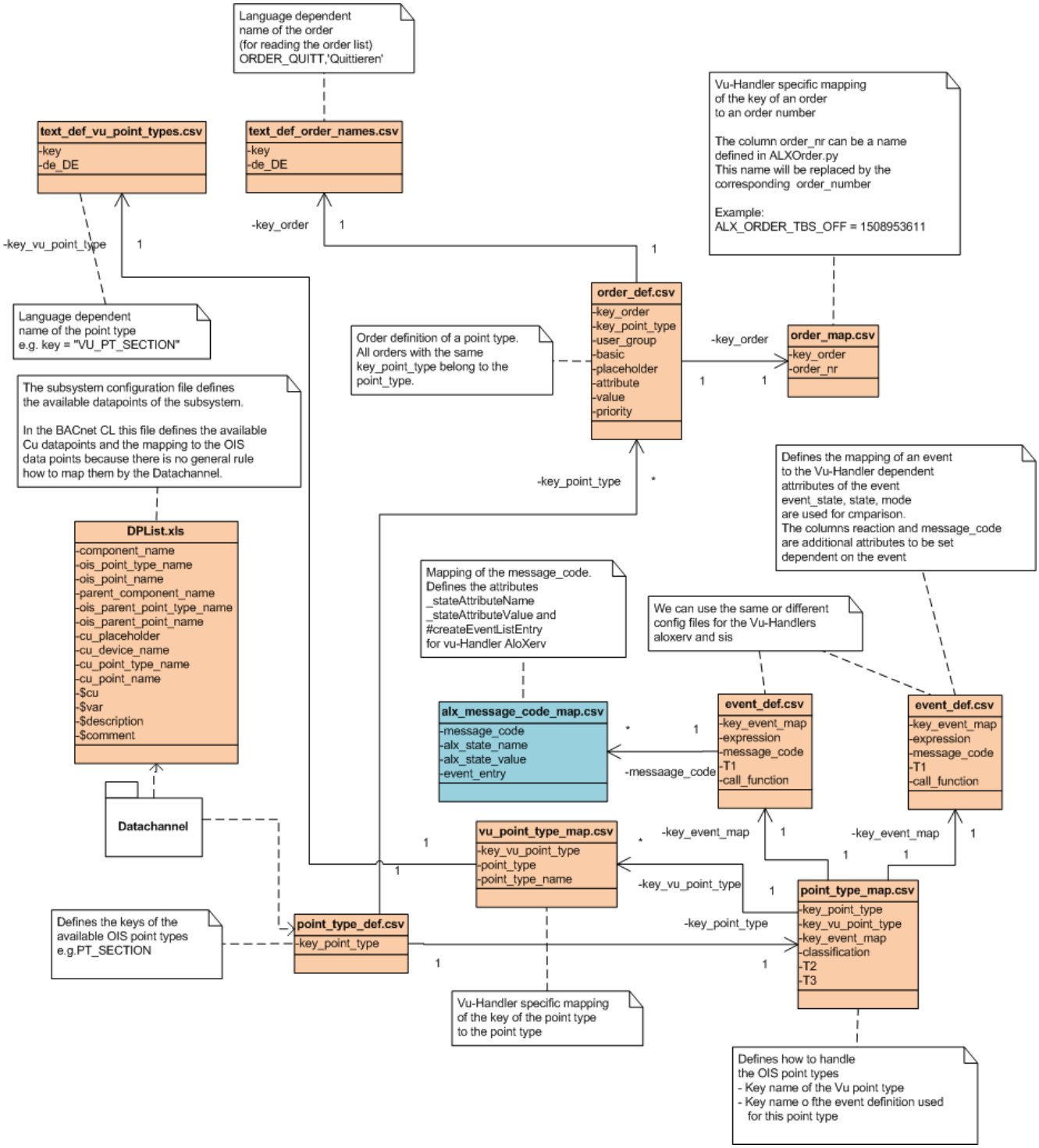


Fig. 1: CL: Configuration files

7.2 Content of the Configuration Files

7.2.1 Data Point Configuration File

File Name	Description
DPList.xls or DpList.csv	This file defines the data points and their relationship with the Cu data points. See chapter *Subsystem Configuration File (DPList.xls) [➙ 37]*.

7.2.2 Event Handler Extensions

File Name	Description
__init__.py	This file starts the CL and defines extension functions for the event handler, such as application-specific callback functions referenced by *event_def.csv*.

7.2.3 Point Type Configuration Files

The following csv configuration files are required to define the point types, their orders and the handling of their events.

File Name	Description
point_type_def.csv	This file defines the list of available OIS point types. See chapter *Point Type Definition (point_type_def.csv) [➙ 41]*.
point_type_map.csv	This file defines the following: ● The mapping of the OIS point types to the related Vu point type ● The event definition used for this point type. See chapter *Point Type Mapping (point_type_map.csv) [➙ 42]*. ● The application-specific attributes and their values
vu_point_type_map.csv	This file defines the mapping of the key name of the Vu point type to the Vu point type for the Vu-Handler. See chapter *Vu Point Type Mapping (vu_point_type_map.csv) [➙ 43]*.
text_def_vu_point_types.csv	This file defines the text of the point types that is visible in the traces and at GMA Manager 8.x. See chapter *Name of the Vu Point Type (text_def_vu_point_types.csv) [➙ 44]*.
order_def.csv	This file defines the available orders and their mapping to the attributes of the BACnet object. See chapter *Order Definition (order_def.csv) [➙ 45]*.
order_map.csv	This file defines the mapping of the orders to the order numbers used by the management station. See chapter *Order Mapping (order_map.csv) [➙ 47]*.
text_def_order_names.csv	This file defines the text of the orders that is visible in the log files and the order list at GMA Manager 8.x. See chapter *Order Names (text_def_order_names.csv) [➙ 49]*.
event_def.csv	This file defines the handling of the events. It defines the reaction sent to the Vu-Handler for the different events. See chapter *Event Definition (event_def.csv) [➙ 49]*.

7.2.4 Connectors (Vu-Handlers)

A connector defines the specific part of a CL that connects to a Vu-Handler of a specific type. The connector-specific configuration files are located in the directory of the connector.

Currently the following connectors are available:

Connector	Description
vusis	Used to connect to GMA Manager 8.x
vualoxerv	Used to connect to Siveillance Control (AloXerv)

Connector	Description
vuvantage	Used to connect to Siveillance Control Pro
vuoisexplorer	This connector is only used for the OIS explorer to show the raw data from the Cu-Handler.

7.3 Subsystem Configuration File (DPList.xls)

The CL template provides an example of the subsystem configuration file in the *etc/import* folder for the unit of the BACnet Cu-Handler. You can use either the Excel file *DPList.xls* or the csv file *DPList.csv*. The XLS file is more user-friendly when editing it (drop-down list). If the csv file is used, however, the data point types and component names cannot be selected from drop-down lists. Set the **configfile** parameter to the file of your choice. See chapter Configuring the Cu-Handler Unit.

This file defines the following:

- The OIS data points and the BACnet objects mapped to them
- The parent data point for the navigation tree at the management station
- The attributes and their values related to the OIS or BACnet data points
  Example of the configuration of the *DPList.xls* file:

	A	B	C	D	E	F	G	H	I	J	K	L	
1	component_name	ois_point_type_name	ois_point_name	paren	o	ois	cu_placeholder	cu_device	cu_point_t	cu	$cu	$var	$description
2	DEVICE_3000	PT_COMMUNICATION	COMMUNICATION00				COMMUNICATION	3000	binary-input	1	text_on="On" text_off="Off"		Communication PT-Interface
3	DEVICE_3000	PT_LINIE	01-0013-0006 01				ANSTEUERUNG	3000	binary-value	1	text_on="On" text_off="Off"		Linie:001 Reserve Raum:
4	DEVICE_3000	PT_LINIE	01-0013-0007 01				ANSTEUERUNG	3000	binary-value	2	text_on="On" text_off="Off"		Linie:002 Reserve Raum:

NOTICE	
!	Excel files with the ending **.xlsx** are only supported by OIS version 2.6.1.

For a site with the name **BACnet_Gaswarnanlage**, a gateway with the name **1**, and a unit with the name **BACNETGASWARNANLAGE**, the *DPList.** file is located in the following directory:

/var/lib/ois/.web/sites/BACnet_Gaswarnanlage/1/BACNET_GASWARNANLAGE/etc/import.

You can access this directory from your Windows PC, for example using **WinSCP**, if you have a network connection to the OIS computer. By double-clicking the *DPList.xls* file, Excel will open with this file on your Windows PC. For this purpose, the following configurations need to be made in **WinSCP** under **Editors** in the **Settings** dialog: **Mask = *.xls** and **Editor = Associated application**.

There is usually no subsystem configuration file from BACnet. Reading the BACnet data point list does not provide the information required to create the list of OIS data points and their neighbors.

The OIS data points have to be defined in an excel file in a **worksheet** with the name **OISDPList**.

The available BACnet objects are usually provided in an EDE file (Engineering Data Exchange). Unfortunately this file does not provide any information on which of these BACnet objects will be merged together in one BACnet data point. In general, the BACnet object type and its instance number can be used to define a rule for how these BACnet objects will be merged together in the OIS data points. This rule is generally application-specific.

Note for the example file DPList.xls:

The "Drop Downs" in the cells, for example to select the point type name, are only defined for 10,000 lines due to performance issues when reading the file. If more lines are required, the line 9999 must be copied to the following lines. From line 10,000 on, Excel cells no longer contain combo boxes for selecting values.

The configuration file provides the following information:

- The name and type of the OIS data point
- Cu Neighbors of the OIS data point from which events can be received and to which orders can be sent.
  If an OIS data point has several neighbors, then additional neighbors are defined in a new line, leaving the **component_name** column and the columns starting with **ois_** empty. In this case, a unique name needs to be defined for the neighbor in the **cu_placeholder** column. This name then has to be used when defining the destination of an order in the *order_def.csv* file.
- The name of the component when Vu-Handler SIS is not used (for example, Vu-Handler Siveillance Control).
  In general, the data points of the individual devices should be assigned to a separate component. (UnitFailed of a component is only False when all devices of this unit are connected.)
- Attributes for the CL related to the OIS or Cu data points
- Attributes in the **$cu** column that are specific to a BACnet data point
- Attributes in the **$var** column and other columns starting with the character **$** that are specific to OIS data points

Column	Description	Example	Required
component_name	Name of the component of the data point This column is only required if data points from different devices are merged together. By default, the data points of different devices are located in different components.	BACNET1	No
ois_point_type_name	Key name of the point type of the OIS data point (value of the **key_point_type** column of the *point_type_def.csv* file)	SECTION	Yes
ois_point_name	Name of the OIS data point and Vu data point visible to the management station	Room 101 00	Yes
parent_component_name	Name of the component of the parent data point This column is not required for Vu-Handler SIS.	BACNET	No
ois_parent_point_type_name	Key name of the point type of the parent OIS data point (from the **ois_point_type_name** column of another data point)	ZONE	No
ois_parent_point_name	Name of the parent OIS data point (from the **ois_point__name** column of another data point)	Building 12 00	No
cu_placeholder	Placeholder for the Cu data point default = CU This column is only required if an OIS data point has more than one Cu neighbor.	NEIGHBOR_LS	No
cu_device_name	Name of the BACnet device This is the value of the instance number of the BACnet device object. Parsing of the table stops if this column is empty.	1	Yes

Column	Description	Example	Required
cu_point_type_name	Name of the point type of the Cu data point This is the type name of the BACnet object (`BACnetObjectType`). The following values are available: ● analog-input ● analog-output ● analog-value ● binary-input ● binary-output ● binary-value ● life-safety-zone ● life-safety-point ● multi-state-input ● multi-state-output ● multi-state-value	life-safety-zone	Yes
cu_point_name	Name of the Cu data point This is the instance number of the BACnet object.	123	Yes
$cu	Attribute definitions related to the given Cu neighbor Use <Alt>+<Enter> in Excel to create a new line for a second attribute. Alternatively, the attributes can be separated by commas.	text_on="Door open" text_off="Door closed"	No
$var	Attribute definitions related to the OIS data point. These attributes are available for all events related to this data point. In general, this column is only used if not all OIS data points provide these attributes.	T2="%(message_text)s %(description)s"	No
$instanceName	Defines the **instanceName** attribute of a data point in the file with the Vu data points. The value of the **ois_point_name** column will be used if the value of this column is not given. This column is only required for connector **aloxerv**.	ÜE 1	No
$comment	Optional Defines the **comment** attribute of a data point in the file with the Vu data points This column is only required for connector **aloxer**.	Handfeuermelder EG Trph. Geb. 4	
$<attr_name> Example $T2	Application-specific attributes: The name of the attribute is defined by the name of the column without the preceding "$". The value of the attribute is defined by the value of the column. This is only an alternative to the **$var** column for defining attributes. In general, this column is only used if all OIS data points provide this attribute.	AVC Fire ALARM	Yes

7.4 COV Subscriptions

The BACnet Service **SubscribeCOV** is used for the registration and removal of subscriptions at the BACnet device. The COV subscriptions are generally only required if the BACnet device does not support intrinsic reporting.

Events from COV reporting cannot be acknowledged by the management station because they never have a unique number. (The BACnet service **AcknowledgeAlarm** is only available for intrinsic reporting.)

7.4.1 Configuring COV Subscriptions (subscribe_cov.csv)

The *subscribe_cov.csv* file defines the COV subscriptions to be sent periodically to the BACnet device.

Registration takes place as follows:

- During connection setup to the BACnet device.
- Periodically in the interval configured with the **subscribe_cov_cycle** parameter. The re-subscriptions are evenly distributed in this interval.

The subscriptions are removed when the Cu-Handler is shut down.

For the configuration of the **subscribe_cov** and **subscribe_cov_cycle** parameters, see chapter *Configuring the BACnet Client (Cu-Handler)* [➙ 74].

Column	Description	Example(s)
device_instance	Instance number of the device object	2171905
object_type	Object type of the BACnet object Supported object types (others are not tested): ● binary-value ● binary-input ● binary-output ● multi-state-value ● multi-state-input ● multi-state-output ● analog-value ● analog-input ● analog-output ● life-safety-point ● life-safety-zone	binary-value
object_instance	Instance number of the BACnet object. The subscription will only be done if an object with such an instance number exists (available in the object list). The configuration can be simplified by using the definition of ranges. Ranges: ● * All available BACnet objects of the given type ● **[<min>:<max>]** All available BACnet objects with the object instance in the given range (including min and max) ● **[<instance1>,<instance2>,..]** List of object instances	* [1:1000] [1,3,5,7] 1
Confirmed	Optional column: ● **True** Notification with **confirmed COV notification** ● **False** Notification with **unconfirmed COV notification** default = True	True

Example of the subscribe_cov.csv File:

```
device_instance;object_type;object_instance;confirmed
# All binary-value objects
2171905;binary-value;*;

# A concrete binary-value objects with instanc number 1
2171905;binary-value;1;

# All binary-
value objects where the object instance is in the range 100
```

```
..10000
2171905;binary-input;[100:10000];

# Only life-safety-
point objects with the object instance 10,20 and 25
2171905; life-safety-point;[10,20,25];
```

7.4.2 Configuring COV Event Definitions (event_def.csv)

The handling of events of the **COV notification** type can be configured in the *event_def.csv* file.

The attributes received with a COV notification can be used in the event definition. See also Attributes from the Cu-Handler COV Notifications for the attributes that are available for the CL from COV notifications.

To simplify the detection and configuration of the COV notifications in the CL, the values of `group` and `entry` are set to 100 for COV notifications.

Example of the Configuration of a Binary-Value Object in event_def.csv for Connector Vusis:

```
key_event_map;placeholder;group;entry;expression;transmit;m
essage_code;T1
EVDEF_MELDERGRUPPE;BV;100;100;value=='active';True;set
alarm;'Window open';
EVDEF_MELDERGRUPPE;BV;100;100;value=='inactive';True;set
message;'Window closed';
```

7.5 Start File for the CL (__init__.py)

This file must be referenced with the **cl** attribute in the **plc** unit. The directory has to be set relative to the *plc* folder in the workspace.

In our example, the **cl** parameter must be set to **etc/ois_clbacnet_example/__init__.py**.

The *event_def.csv* file can reference application-specific functions to be called for an event. See also chapter *Event Definition (event_def.csv)* [➙ 49]. The application-specific functions must be implemented in the *__init__.py* file, for example in the class **BACPointTypeApplication**. See chapter *Example of the Implementation of an Event Callback Function* [➙ 81] for an example of how to implement the callback function.

The address format of the data point name for GMA Manager 8.x needs to be defined in the **on_init** function in the *__init__.py* file. The address length (20, 2) is set by default.

7.6 Point Type Configuration

7.6.1 Point Type Definition (point_type_def.csv)

This file defines the key names **key_point_type** of the OIS data points. These names are the key names of the point type and will be used to reference this point type in other configuration files.

Example:

```
key_point_type
# Definition of the available point types of the OIS-
data points
# key_point_type     - Key name of the OIS point type.
#                      (used for the point-type-
```

```
name of the OIS data point)
#
PT_SECTION
PT_ZONE
PT_BINARY_INPUT
PT_BINARY_OUTPUT
PT_MULTI_STATE_VALUE
```

In this example, all values of the **key_point_type** column start with **PT_**. This allows searching for all definitions related to a point type.

7.6.2 Point Type Mapping (point_type_map.csv)

This configuration file defines the key names of the Vu point type definition and the key name of the event definition related to the point types.

You can additionally define attributes that are specific to the OIS data point types. These attributes are used for defining the attributes of the event which are identical for all data points of this type.

Example of Vu-Handler SIS:

```
key_point_type;key_vu_point_type;key_event_map;classificati
on;T3;T2;state_def
# Definition of the point types for the Vu-Handler-SIS
#
# key_point_type     - Key name of the OIS point type
# key_vu_point_type  - Key name of the VU point type
# use_point_type     -
 Optional column. Set to False if the point type is not use
d
#                      In our example we use all point type
s, so we have deleted
#                      this column.
# key_event_map      -
 Key name of the according event mapping definition defined
#                      in event_def.csv. Leave this column
empty if the point type
#                      does not support events.
# classification     - classification of the data point
#                       (This text is used for @MA in the at
 file)
#                      AREA, SECTION ZONE, SENSOR, OTHER, .
..)
# T3                 -
 Application specific column to define T3 for Vu-
Handler sis
#                      %
(type_name)s: Type name of the point type as defined in
#                                        text_def_vu_point_typ
es.csv
#                      %
(point_name)s: Name of the data point
# T2                 -
 Application specific column to define T2 for Vu-
Handler sis
#                      %
(message_text)s: Message text from the Cu-Handler event
#                      %
(description)s: Value of the attribute description from
#                                         the subsystem confi
guration file
# state_def          -
 Dictionary with the mapping of the state to the state name
```

```
#                       for multistate objects

PT_SECTION;VU_PT_SECTION;EVDEF_SECTION;SECTION;"%
(type_name)s*%(point_name)s";"%(message_text)s %
(description)s"
PT_ZONE;VU_PT_ZONE;EVDEF_ZONE;ZONE;"%(type_name)s*%
(point_name)s";"%(message_text)s %(description)s"
PT_BINARY_INPUT;VU_PT_SENSOR;EVDEF_BINARY_INPUT;SENSOR;"%
(type_name)s*%(point_name)s";"%(description)s"
PT_BINARY_OUTPUT;VU_PT_SENSOR;EVDEF_BINARY_OUTPUT;SENSOR;"%
(type_name)s*%(point_name)s";"%(description)s"
PT_MULTI_STATE_VALUE;VU_PT_SENSOR;EVDEF_MULTI_STATE;SENSOR;
"%(type_name)s*%(point_name)s";"%(description)s";{0:'State
0', 1:'State 1', 2:'State 2'}
```

Example of Vu-Handler Siveillance Control (AloXerv):

The values of the **mot** column need to be clarified with the Siveillance Control experts.

```
key_point_type;key_vu_point_type;key_event_map;mot;state_de
f
# Definition of the Point types for the Vu-Handler-AloXerv
#
# key_point_type     - Key name of the OIS point type
# key_vu_point_type  - Key name of the VU point type
# use_point_type     -
 Optional column. Set to False if the point type is not use
d
#                       In our example we use all point type
s, so have deleted
#                       this column.
# key_event_map      -
 Key name of the according event mapping definition defined
#                       in event_def.csv
# mot                -
 MOT of the data point required for the list of Vu
#                       data points for Siveillance Control
# state_def          -
 Dictionary with the mapping of the state to the state name
#                       for multistate objects

PT_SECTION;VU_PT_SECTION;EVDEF_SECTION;ALX_MOT_FIRE_ZONE
PT_ZONE;VU_PT_ZONE;EVDEF_ZONE;ALX_MOT_INTRUSION_ZONE
PT_BINARY_INPUT;VU_PT_SENSOR;EVDEF_BINARY_INPUT;ALX_MOT_TBS
_BINARY_INPUT
PT_BINARY_OUTPUT;VU_PT_SENSOR;EVDEF_BINARY_OUTPUT;ALX_MOT_T
BS_BINARY_OUTPUT
PT_MULTI_STATE_VALUE;VU_PT_SENSOR;EVDEF_MULTI_STATE;ALX_MOT
_TBS_BINARY_INPUT;{0:'State 0', 1:'State 1', 2:'State 2'}
```

7.6.3 Vu Point Type Mapping (vu_point_type_map.csv)

This file defines the point type of the Vu data point. It is allowed to map several OIS point types to one Vu point type if these data points have different names.

Example of Vu-Handler SIS:

```
key_vu_point_type;point_type
# Definition of the point types for the Vu-Handler-SIS
#
# key_vu_point_type  - Key name of the Vu point type
# point_type         -
```

```
 Point type (int value) of the Vu data point

VU_PT_SECTION;6
VU_PT_ZONE;17
# It is allowed to map several point types to the same Vu p
oint type if the
# point names are unique
VU_PT_SENSOR;17
```

Example of Vu-Handler Siveillance Control (AloXerv):

NOTICE	
!	**point_type_name** is currently not supported by Siveillance Control.

```
key_vu_point_type;point_type
# Definition of the Point types for the Vu-Handler-AloXerv
#
# key_vu_point_type  - Key name of the Vu point type
# point_type         -
 Point type (int value) of the Vu datapoint

VU_PT_SECTION;6
VU_PT_ZONE;17
VU_PT_SENSOR;17
```

7.6.4 Name of the Vu Point Type (text_def_vu_point_types.csv)

This file defines the language-dependent names of the Vu point types. These names are visible at GMA Manager 8.x.

The language to be used is defined by the **language** parameter of the **plc** unit.

```
key;de_DE
# The file must be encoded in ISO-8859-1 if you use German
umlaute.
# This file defines the language dependent names of the poi
nt types.
# Set the parameter language in the plc unit to select the
desired language.
# All point types mapped to the same VU point type must hav
e the same name
# This text is used for traces and for the result of the ge
t_datapoint_types
# request from the management system.
#
# key   - Key name of the point type
# de_DE - German text for the point type
# en_GB - Optional English text for the point types

# Definition of the type_name of the Point Types;
# Here we use the same name for VU_PT_ZONE and VU_PT_SENSOR
 due to mapping
# to the same Vu point type.
VU_PT_SECTION;Bereich/Abschnitt;Area/Section
VU_PT_OBJECT;Objekt;Object
VU_PT_ZONE;Meldergruppe/Melder;Detector Group/Detector
VU_PT_SENSOR;Meldergruppe/Melder;Detector Group/Detector
```

7.6.5 Order Definition (order_def.csv)

The order definition defines the available orders and the attribute to be set.

An additional **placeholder** column is required in the *order_def.csv* file if an OIS data point has more than one Cu neighbor.

You can define several values to be set by one order. Leave the **user_group** and **basic** columns empty for further values.

Column	Description	Example(s)
key_point_type	Key name of the point type of the OIS data point or Definition of the templates if the key name starts with **Template_**	• SECTION • Template_LifeSafety
key_order	Key name of the order or a list of template references separated by space	• ENABLED • Template_LifeSafety
user_group	The user group the order is assigned to (used for getting the order list of a DP)	4
basic	Basic flag of the order for message processing (Level 1) in GMA Manager (used for getting the order list of a DP)	• 0 • 1
placeholder	Placeholder for the Cu neighbor default = CU The placeholder must be one of the placeholders defined in the subsystem configuration file	• cu_1 • RESET
attribute	Name of the BACnet property to be set: *LSOP* for LifeSafetyOperation to a station *LSOE* for LifeSafetyOperation to a BACnet object	• mode • present-value • LSOP • LSOE
value	Value of the attribute to be set. The values depend on the type of the BACnet property. Example: A BACnet property of the type **BACnetBinaryPV** supports the values **inactive** and **active**.	Attribute mode, Type BACnetLifeSafetyMode: • 'enabled' • 'manned' Attribute present-value, type *BACnetBinaryPV* • 'inactive' • 'active' Attribute LSOP • 'reset'
priority	Value for the priority when the BACnet object supports priority arrays The orders from a management station generally have priority 8. Writing the value **NULL** to the property releases the entry in the priority array with the given priority.	8
delay	Delay before the order is executed Optional delay for an order (10 seconds max.) **Important:** During the delay, it is not possible to send other orders to the same device.	1

Example:

```
key_point_type;key_order;user_group;basic;placeholder;attri
bute;value
# Definition of the available orders for the point types
# See also text_def_order_names.csv and order_map.csv
#
# key_point_type -
 Key name of the point type of the OIS data point or name o
```

```
f the
#                   template
# key_order      -
 Key name of the order or a list of template references
#                   separated by space
# user_group     -
 User group where the order belongs to (used for getting th
e
#                   order list of a DP)
# basic          - Basic flag of the order
# placeholder    -
 Placeholder of the neighbor where the command has to be se
nt to
# attribute      - Name of the attribute to be set
# value          - Value of the attribute to be set
# priority       -
 optional value for the priority when the BACnet object sup
ports
#                   a priority array for the property


# Template definition (Common orders)
# Key names starting with Template_ are template definition
s
Template_LifeSafety;ORDER_RESET_STATION;1;0;NEIGHBOR_LS;LSO
P;'reset'

# >>>> Orders of a section <<<<

# Example for using a template definition
PT_SECTION;Template_LifeSafety

# Example for orders with 2 reactions
PT_SECTION;ORDER_OFF;1;0;NEIGHBOR_BO;present-value;0
PT_SECTION;ORDER_OFF;;;NEIGHBOR_LS;mode;'off'

PT_SECTION;ORDER_ON;1;0;NEIGHBOR_BO;present-value;1
PT_SECTION;ORDER_ON;1;0;NEIGHBOR_LS;mode;'on'

# >>>> Orders of a zone <<<<
PT_ZONE;Template_LifeSafety

# >>>> Orders of a Binary-Output <<<<
PT_BINARY_OUTPUT;ORDER_OFF;1;0;NEIGHBOR_BO;present-value;0
PT_BINARY_OUTPUT;ORDER_ON;1;0;NEIGHBOR_BO;present-value;1
```

Example: Order with Priority 8

Priority 8 is generally used for OIS.

```
key_point_type;key_order;user_group;basic;placeholder;attri
bute;value;priority
PT_MELDERGRUPPE;ORDER_OFF;1;0;BV;present-value;'inactive';8
PT_MELDERGRUPPE;ORDER_ON;1;0;BV;present-value;'active';8
# Release the entry in the priority array for lower
priorities
PT_MELDERGRUPPE;ORDER_RESET_PRIO_8;1;0;BV;present-
value;'NULL';8
```

Example: Order with Two Reactions:

Here the present value is set to 1 and then immediately back to 0.

The **user_group** and **basic** columns may only be defined in the first reaction of the order.

```
key_point_type;key_order;user_group;basic;placeholder;attri
bute;value
PT_SECTION;ORDER_OFF;1;0;NEIGHBOR_BO;present-value;1
PT_SECTION;ORDER_OFF;;;NEIGHBOR_BO;present-value;0
```

7.6.6 Order Mapping (order_map.csv)

This file defines the mapping of the orders to the order numbers of the management station.

Example of Vu-Handler SIS:

The order number must be greater than 10

```
key_order;order_nr
# Defines the order numbers for the Vu-Handler-Sis
# Mapping of the key name of an order to the order number r
eceived from
# the Vu-Handler.
# See also the file order_def.csv
#
# key_order  - Key name of the order
# order_nr   - Order number for Vu-Handler-Sis

ORDER_OFF;11
ORDER_ON;12
ORDER_RESET_STATION;13
```

Example of Vu-Handler Siveillance Control (AloXerv):

```
key_order;order_nr
# Defines the order numbers for Vu-Handler-AloXerv
# Mapping of the key name of an order to the order number r
eceived from
# the Vu-Handler
# See also the file order_def.csv
#
# key_order - Key name of the order
# order_nr  - Order number for Vu-Handler-
AloXerv or an order name from the file
#             cl/helper/vu/aloxerv/ALXOrders.py.
#
# The mapping of all orders shall be defined here. If the o
rder is not defined
# for AloXerv then the same mapping from vusis/
order_map.csv shall be configured
# for the order.

ORDER_OFF;ALX_ORDER_TBS_OFF
ORDER_ON;ALX_ORDER_TBS_ON
ORDER_RESET_STATION;ALX_ORDER_FIRE_RESET_PANEL
```

7.6.6.1 Available Orders for Connector Vualoxerv

The following orders are available for connector **vualoxerv**. These names should be used instead of the integer value of the order number to ensure a better readability of the configuration.

See also file *ois/cl/helper/vu/aloxerv/ALXOrders.py* where the corresponding order numbers are defined.

- ALX_ORDER_ACCEPT
- ALX_ORDER_ARM_SENSOR
- ALX_ORDER_DELETE
- ALX_ORDER_FIRE_ACOUSTICS_OFF
- ALX_ORDER_FIRE_CONTROLLER_ACTIVATE
- ALX_ORDER_FIRE_EVAC_OFF
- ALX_ORDER_FIRE_EVAC_ON
- ALX_ORDER_FIRE_LOCAL_ALARMING_OFF
- ALX_ORDER_FIRE_LOCAL_ALARMING_ON
- ALX_ORDER_FIRE_RESET_PANEL
- ALX_ORDER_FIRE_START_INVESTIGATION_TIME
- ALX_ORDER_FIRE_STOP_INVESTIGATION_TIME
- ALX_ORDER_FIRE_SWITCH_OFF_ALARMING
- ALX_ORDER_FIRE_VERIFICATION_OFF
- ALX_ORDER_FIRE_VERIFICATION_ON
- ALX_ORDER_INVESTIGATE
- ALX_ORDER_LOCK_PERMANENTLY
- ALX_ORDER_RESPONISBILITY_DISPENSE
- ALX_ORDER_RESPONISBILITY_TAKEOVER
- ALX_ORDER_SECURE
- ALX_ORDER_SET_TO_ALARM
- ALX_ORDER_SET_TO_QUIET
- ALX_ORDER_SHOW_VIDEO
- ALX_ORDER_START_RECORDING
- ALX_ORDER_STOP_RECORDING
- ALX_ORDER_TBS_ACCEPT
- ALX_ORDER_TBS_ERASE_REVISION
- ALX_ORDER_TBS_GROUP_OFF
- ALX_ORDER_TBS_GROUP_ON
- ALX_ORDER_TBS_INTRUSION_ACKNOWLEDGE
- ALX_ORDER_TBS_INTRUSION_ACKNOWLEDGE_PANEL
- ALX_ORDER_TBS_INTRUSION_RESET_INPUT
- ALX_ORDER_TBS_INTRUSION_REVISION_AUTH_OFF
- ALX_ORDER_TBS_INTRUSION_REVISION_AUTH_ON
- ALX_ORDER_TBS_INTRUSION_SET_INPUT
- ALX_ORDER_TBS_INTRUSION_WALKTEST_OFF
- ALX_ORDER_TBS_INTRUSION_WALKTEST_ON
- ALX_ORDER_TBS_INTRUSION_ZONE_ARM
- ALX_ORDER_TBS_INTRUSION_ZONE_ARM_INTERNAL
- ALX_ORDER_TBS_INTRUSION_ZONE_DISARM
- ALX_ORDER_TBS_INTRUSION_ZONE_DISARM_INTERNAL
- ALX_ORDER_TBS_INTRUSION_ZONE_QUERYSTATE
- ALX_ORDER_TBS_INTRUSION_ZONE_REMOVEDISARMINGLOCK
- ALX_ORDER_TBS_INTRUSION_ZONE_SETDISARMINGLOCK
- ALX_ORDER_TBS_OFF
- ALX_ORDER_TBS_ON

- ALX_ORDER_TBS_RESET
- ALX_ORDER_TBS_REVISIONMODE_OFF
- ALX_ORDER_TBS_REVISIONMODE_ON
- ALX_ORDER_TBS_START_ACTUATOR
- ALX_ORDER_TBS_START_DATAPOINT
- ALX_ORDER_TBS_STOP_ACTUATOR
- ALX_ORDER_TBS_STOP_DATAPOINT
- ALX_ORDER_TESTORDER1
- ALX_ORDER_TEST_ORDER
- ALX_ORDER_UNARM_SENSOR
- ALX_ORDER_UNLOCK_PERMANENTLY
- ALX_ORDER_UNLOCK_TEMPORARY

7.6.7 Order Names (text_def_order_names.csv)

This file defines the names of the orders that are visible at the management station and in the traces.

```
key;de_DE
# The file must be encoded in ISO-8859-1 if you use german
umlaute
# This file defines the language dependent names of the ord
ers
# These names will be used in the traces and will be return
ed if a management
# station requests  the list of the orders for a data point
.
#
# key   - Key name of the order
# de_DE - German text

ORDER_OFF;Abschalten
ORDER_ON;Zuschalten
ORDER_RESET_STATION;Rückstellen
```

7.6.8 Event Definition (event_def.csv)

This file defines the reactions to events received from the Cu-Handler. If several BACnet objects are mapped to the same data point, then the event definitions have to be configured separately for each of these neighbors. The definitions for the different neighbors are distinguished by the name of the placeholder.

The value of the **entry** column can be used to optimize the performance if many lines with the same values for **key_event_map** and **placeholder** were defined. This optimization is usually not required.

The last active event that is sent to a Vu-Handler is generally reset automatically before a new active event initiated by the same Cu data point is transmitted.

Only the message codes "set inspection" and "set inspection message" are never reset automatically.

You can see the line number of the reaction to the event found in the DEBUG trace.

(Search for **EventReactionDef** in the trace.)

Column	Description	Examples	Required
key_event_map	Key name of the event definition referenced by *point_type_map.csv*	EVDEF_ZONE	Yes

Column	Description	Examples	Required
placeholder	Name of the placeholder for the Cu neighbor from which the event is received (default = CU)	cu_1 EVENT_REVISION	No
group	Group of the event In general, it is not required to configure the values for **group** and **entry**. **Group** and **entry** should be configured with 100 for the COV notifications only if intrinsic reporting and COV notifications are used. If this column is empty, the definition applies for all values of **group**. The value for **group** is as follows: ● -1 for events from the update ● 100 for COV notifications ● The integer value for the **from_state** property from the event for intrinsic reporting Integer values for the **from_state** property include: ● normal = 0 ● fault = 1 ● offnormal = 2 ● high_limit = 3 ● low_limit = 4 ● life_safety_alarm = 5	100	No
entry	Entry for the event If this column is empty, the definition applies for all values of **entry**. The value for the event from intrinsic reporting is the integer value for the **to-state** property from the event.	100	No
expression	Optional expression to be checked The line is used if the result of the expression is True. The attributes from the Cu-Handler and the attributes defined in the configuration files can be used here. See chapter Attributes from the Cu-Handler COV Notifications. The syntax for the expression is defined by the Python language.	to_event_state=='offnormal' and value=='abnormal'	Yes
transmit	Defines whether the event is transmitted or discarded (optional column) True: Transmit the event (default if the column is empty) False: Discard the event	True False	No
call_function	Defines the application-dependent function that can be called for this event, for example to read further attributes See chapter *Predefined Callback Functions for event_def.csv [↦ 53]* for the predefined callback functions. Implement your own functions in the *__init__.py* file in the workspace. See chapter *Example of the Implementation of an Event Callback Function [↦ 81]*. BACPointTypeExample.callback_example as an example. **NOTICE! You have to use the character & between the function calls to call several callback functions.**	callback_example(value_1=10, value_2=11)	No
<varname>	<varname>, for example T1, defines the name of the attribute. The value is defined by the content of the column.	'ALARM' '%(text_on)s' 'Status=%(text_on)s'	No

Columns for Vu-Handler SIS (Connector Vusis)

Column	Description	Examples	Required
message_code	Message code from **ois.cl.helper.MessageCodes** to be sent to the Vu-Handler SIS with the **ST** attribute	set message reset message	Yes

Columns for Vu-Handler Siveillance Control (Connector Vualoxerv)

You can either use the default mapping of the message code or define the **stateAttributeValue** attributes for **createEventListEntry** for Siveillance Control.

If the *event_def.csv* file is not defined for connector **vualoxerv**, then the *event_def.csv* file for connector **vusis** will be used. The message codes are then mapped to the attributes required for connector **vualoxerv**. See chapter *Example of how to Use a Timer in the Event Callback Function* [➙ 82].

Column	Description	Examples	Required
stateAttributeValue	Defines the **_stateAttributeValue** attribute for Siveillance Control. Only the event states allowed for the **mot** will be used.	ALX_STATE_ALARM_ALARM	Yes
createEventListEntry	Defines the **#createEventListEntry** attribute for Siveillance Control: ● 1 – Create an event list entry in Siveillance Control ● 0 – No event list entry in Siveillance Control	1 0	Yes

Example:

```
key_event_map;placeholder;expression;transmit;message_code;
T1;call_function
# ------ Event Definition for SECTION ------
EVDEF_SECTION;NEIGHBOR_LS;active==True and
value=='alarm';True;set alarm;'%(text_on)s pv=%
(present_value_bi)s';"get_unit_attribute(var_name='present_
value_bi', attr_name='present-value',
placeholder='NEIGHBOR_BI')"
EVDEF_SECTION;NEIGHBOR_LS;active==False;True;reset alarm;'%
(text_off)s'
EVDEF_SECTION;NEIGHBOR_LS;active==True;True;set alarm;'%
(text_on)s'
EVDEF_SECTION;NEIGHBOR_LS;active==False;True;reset alarm;'%
(text_off)s'


EVDEF_SECTION;NEIGHBOR_BI;active==True;True;set sabotage
alarm;'%(text_on)s'
EVDEF_SECTION;NEIGHBOR_BI;active==False;True;reset sabotage
alarm;'%(text_off)s'

# ------ Event Definition for ZONE ------
EVDEF_ZONE;;active==True;True;set alarm;'%(text_on)s'
EVDEF_ZONE;;active==False;True;reset alarm;'%(text_off)s'

# ------ Event Definition for BINARY_INPUT ------
EVDEF_BINARY_INPUT;;active==True;True;set information;'%
(text_on)s'
EVDEF_BINARY_INPUT;;active==False;True;reset information;'%
(text_off)s'

# ------ Event Definition for BINARY_OUTPUT ------
EVDEF_BINARY_OUTPUT;;active==True;True;set information;'%
(text_on)s'
```

```
EVDEF_BINARY_OUTPUT;;active==False;True;reset
information;'%(text_off)s'

# Mutistate value object
EVDEF_MULTI_STATE;;active==False;True;quiet;
EVDEF_MULTI_STATE;;active==True;True;set system fault;%
(state_text)s;"get_state_text()"
```

8 Predefined Callback Functions for event_def.csv

The following callback functions for the event are already defined in the framework CL:

- **get_unit_attribute**
  for reading additional attributes from the Cu-Handler
- **get_state_text**
  maps the appropriate state name to a state number.
- **shell_command**
  executes a shell command and returns the output.

These functions can be used in the **call_function** column in the *event_def.csv* file.

8.1 get_unit_attribute

This function can be used to read additional attributes from the Cu-Handler. The data point to read from must be defined as a neighbor of the OIS data point handling the event.

Parameters of **get_unit_attribute**

Parameter Name	Description
var_name	Name of the variable in **attribute_dict** to be set
attr_name	Name of the BACnet attribute to be read
placeholder	Name of the placeholder for the Cu neighbor from which to read
var_type	Optional parameter Type of the attribute to be read. Default = Integer 's': String 'f': float
delay	Optional parameter Delay before getting the attribute in seconds. The maximum allowed value is 5 seconds. **Important**: The delay blocks the processing of further events of this device.

Example of the Configuration in event_def.csv:

Reading the **present_value** attribute from the Cu data point neighbor 'NEIGHBOR_BI'. The value of the **present_value** attribute will be used for the text T1.

```
key_event_map;placeholder;expression;transmit;message_code;
T1;call_function
EVDEF_SECTION;NEIGHBOR_LS;active==True and
new_state=='alarm';True;set alarm;'%(text_on)s pv=%
(present_value)s';"get_unit_attribute(var_name='present_val
ue', attr_name='present-value',placeholder='NEIGHBOR_BI',
var_type='s')"
```

8.2 shell_command

This function executes shell commands.

It is not required to use the quotation marks " or ' around the **shell_command** itself and the value of the command. This makes it easier to use these quotation marks inside the command itself.

The following shell command definitions are equal:

```
shell_command(command=pwd)
shell_command(command='pwd')
shell_command(command="pwd")
"shell_command(command=pwd)"
"shell_command(command="pwd")"
'shell_command("command=pwd")'
'shell_command("command='pwd'")'
```

Parameters of **shell_command**

Parameter Name	Description

command	Shell command to be executed
	The working directory is the directory of the **plc** unit in the workspace.
	The Python string replacement in the command can be used to access already known variables.
	Examples:
	`command="./etc/ois_clbacnet_example/shell_example.sh '%(value)s' 123"`
	`command="pwd"`
	The path of the shell script is relative to the *__init__.py* file of the CL.
	See chapter *Start File for the CL (__init__.py) [➙ 41]*.
	`command="%(shell_command_dir)s/shell_example.sh '%(value)s' 123"`

var_name_result	Optional name of the variable with the result of the command default='result'

Example of the Configuration in event_def.csv:

Execute the shell script located in the *etc* directory of the **plc** unit.

```
key_event_map;placeholder;expression;transmit;message_code;
T1;call_function
EVDEF_MULTI_STATE;;value==4;True;set system fault;result=%
(result)s; shell_command(command="%(shell_command_dir)s/
shell_example.sh '%(value)s' 123")
```

8.3 get_state_text

This function maps the state to the respective state name. The name of the state will then be available (in the **attribute_dict** dictionary).

For an example of the implementation of the callback function, refer to chapter *Example of the Implementation of an Event Callback Function* [➙ 81].

Parameters of **get_state_text**

Parameter Name	Description
var_name_state	Optional name of the variable in the **attribute_dict** dictionary representing the state. The type of the variable should be an integer default = ' value'
var_name_state_def	Optional name of the variable in the **attribute_dict** dictionary, which should be a dictionary with the mapping of the sates to the state names. This variable can be defined in either of the following files: ● *point_type_map.csv* for example, **state_def** column with the value {*0:'State 0', 1:'State 1', 2:'State 2'}* ● Subsystem configuration file *DPList.xml* for example, **state_def** column with the value {*0:'State 0', 1:'State 1', 2:'State 2'}* default = 'state_def'
var_name_result	Optional name of the variable with the resulting state text default='state_text'

Example of the Configuration in point_type_map.csv:

The **state_def** column defines the mapping of the state value to the state names.

```
key_point_type;key_vu_point_type;key_event_map;classificati
on;T3;T2;state_def
PT_MULTI_STATE_VALUE;VU_PT_SENSOR;EVDEF_MULTI_STATE;SENSOR;
"%(type_name)s*%(point_name)s";"%(description)s";{0:'State
0', 1:'State 1', 2:'State 2'}
```

Example 1 of the Configuration in event_def.csv:

Gets the state name of the state and sets **state_text**, which will be used for the **T1** attribute

```
key_event_map;placeholder;expression;transmit;message_code;
T1;call_function
EVDEF_MULTI_STATE;CU;active==True;True;set system fault;%
(state text)s; "get state text(var name state ='value', var
name state def='state def', var_name_result='state_text')"
```

Example 2 of the Configuration in event_def.csv:

Gets the state name of the state and sets **state_text**, which will be used for the **T1** attribute

Here we use the default values for the parameters.

```
key_event_map;placeholder;expression;transmit;message_code;
T1;call_function
EVDEF_MULTI_STATE;CU;active==True;True;set system fault;%
(state text)s; "get_state_text()"
```

9 String Formatting

The values of the attributes in the configuration files can reference the values of other attributes.

As a reference we use Python string formatting where the attributes for the formatting are given as a dictionary.

The following table shows examples of the attributes and their values and how they are resolved after formatting.

Attribute Name	Attribute Value before Formatting	Attribute Value after Formatting
int_var	12345	12345
float_var	12.34567	12.34567
str_1	'1234567890'	'1234567890'
str_2	'This string is to long'	'This string is to long'
str_3	'Value='	'Value='
int_right_align	'%(int_var)12d'	' 12345'
int_left_align	'%(**int_var**)-12**d**'	'12345 '
float_right_align_3	'%(**float_var**)-12.3**f**'	' 12.346'
float_left_align_3	'%(**float_var**)-12.3**f**'	'12.346 '
str_right_align	'%(**str_1**)12**s**'	' 1234567890'
str_left_align	'%(**str_1**)-12**s**'	'1234567890 '
str_right_align_cut	'%(**str_2**)12.10**s**'	'This strin '
str_left_align_cut	'%(**str_2**)-12.10**s**'	'This strin '
T1	'%(str_3)s%(int_left_align)s persons'	'Value=12345 persons'
hex_left_align	'0x%(int_var)-8X'	'0x3039 '
hex	'0x%(int_var)0.8X'	'0x00003039'

10 Attributes Available in the CL

The following chapters describe the attributes that can be used in the CL, for example in the **expression** column of the *event_def.csv* file.

The values of the attributes defined in the configuration files can use the string formatting described in chapter *String Formatting* [→ 56].

10.1 Attributes from the Cu-Handler COV Notifications

If COV notifications are used for analog objects, then the BACnet property **cov-increment** has to be configured properly in the BACnet object.

Name of the Event Attribute	Type	BACnet Type	Remark
active	int	-	The value of this attribute is always set to 1 (active event).
event_type	str	BACnetEventType	The value is always **cov**.
fault	int	BACnetStatusFlags	**fault** flag from the status flags
in_alarm	int	BACnetStatusFlags	**in_alarm** flag from the status flags
overridden	int	BACnetStatusFlags	**overridden** flag from the status flags
out_of_service	int	BACnetStatusFlags	**out_of_service** flag from the status flags
value	str, float		The type of value depends on the type of object from which the COV notification originates.

10.2 Attributes from the Cu-Handler Event

The attributes described here will be received as a result of confirmed or unconfirmed event notifications (intrinsic-reporting).

10.2.1 Attributes that Are Common for All Types of Events

The following attributes are available for all events from the Cu-Handler.

Name of the Event Attribute	Type	BACnet Type	Remark
active	int	-	The value of this attribute is set to 1 for active events (`to_event_state != 'normal'`) and 0 for inactive events.
event_type	str	BACnetEventType	Type of event For example, `change-of-bitstring` The following values are supported: ● change-of-state ● change-of-value ● floating-limit ● out-of-range ● change-of-life-safety ASN1: UnconfirmedEventNotification-Request.event-type

Name of the Event Attribute	Type	BACnet Type	Remark
to_event_state	str	BACnetEventState	New value of the event state The following values are available depending on the BACnet object type: ● normal (0) ● fault (1) ● offnormal (2) ● high-limit (3) ● low-limit (4) ● life-safety-alarm (5) In the OIS 2.6.1 CL we use, for example, `to_event_state=='life-safety-alarm'` in the **expression** column of the *event_def.csv* file. In the old OIS 2.3 CL we used entry=5. ASN1: UnconfirmedEventNotification-Request.to-event-state
from_event_state	str	BACnetEventState	Old state of the event ASN1: `BACnetNotificationParameters.from-state` Not available for update events where `group==-2` See also **to-event-state** ASN1: UnconfirmedEventNotification-Request.from-state
message_text	str	CharacterStringAsn 1	Optional text of the event ASN1: UnconfirmedEventNotification-Request. message-text

10.2.2 Change-of-State Event

The following attributes are available for events where `event-type==change-of-state`.

Name of the Event Attribute	Type(s)	BACnet Type	Remark
status_flags	str	BACnetStatusFlags	Status flags of the event ASN1: change_of_state.status-flags

value	str,
	int
		BACnetPropertyStates	Change of state (`new-state` attribute from the BACnet event)
			The type depends on the BACnet object,
			for example:
			`BooleanAsn1`
			`BACnetLifeSafetyMode`
			`BACnetLifeSafetyState`
			
			ASN1: change_of_state.new-state

10.2.3 Change-of-Value Changed-Bits Event

The following attributes are available for events where `event-type==change-of-value` and a bit string has changed.

Name of the Event Attribute	Type	BACnet Type	Remark
status_flags	str	BACnetStatusFlags	Status flags of the event ASN1: change_of_value.status-flags
value	str	BitStringAsn1	New value of the bit string (`referenced-bitstring` attribute from the BACnet event) ASN1: change_of_value.new-value.changed-bits

10.2.4 Change-of-Value Changed-Value Event

The following attributes are available for events where `event-type==change-of-value` and a value has changed.

Name of the Event Attribute	Type	BACnet Type	Remark
status_flags	str	BACnetStatusFlags	Status flags of the event ASN1: change_of_value.status-flags
value	float	RealAsn1	New value `new-value` attribute from the BACnet event) ASN1: change_of_value.new-value.changed-value

10.2.5 Floating-Limit Event

The following attributes are available for events where `event-type==floating-limit`.

Name of the Event Attribute	Type	BACnet Type	Remark
status_flags	str	BACnetStatusFlags	Status flags of the event ASN1: floating_limit.status-flags
value	float	RealAsn1	Value exceeding the limit ASN1: floating_limit.reference-value

10.2.6 Out-of-Range Event

The following attributes are available for events where `event-type==out-of-range`.

Name of the Event Attribute	Type	BACnet Type	Remark
status_flags	str	BACnetStatusFlags	Status flags of the event ASN1: out-of-range.status-flags
value	float	RealAsn1	Value exceeding the limit ASN1: out_of_range.exceeding-value

10.2.7 Unsigned-Range Event

The following attributes are available for events where `event-type==out-of-range`.

Name of the Event Attribute	Type	BACnet Type	Remark
status_flags	str	BACnetStatusFlags	Status flags of the event ASN1: unsigned_range.status-flags
value	float	RealAsn1	Value exceeding the limit ASN1: unsigned_range.exceeding-value

10.2.8 Change-of-Life-Safety Event

The following attributes are available for events where `event-type==change-of-life.safety`.

Name of the Event Attribute	Type	BACnet Type	Remark
status_flags	str	BACnetStatusFlags	Status flags of the event ASN1: change_of_life_safety.status-flags
value	str	BACnetLifeSafetyState	New state of the Life-Safety object (from present-value) `new-state` attribute from the BACnet event) ASN1: change_of_life_safety.new-state The following values are available: ● quiet ● pre-alarm ● alarm ● fault ● fault-pre-alarm ● fault-alarm ● not-ready ● active ● tamper ● test-alarm ● test-active ● test-fault ● test-fault-alarm ● holdup ● duress ● tamper-alarm ● abnormal ● emergency-power ● delayed ● blocked ● local-alarm ● general-alarm ● supervisory ● test-supervisory
new_mode	str	BACnetLifeSafetyMode	New mode of the Life-Safety object ASN1: change_of_life_safety.new-mode The following values are available: ● OFF ● ON ● test ● manned ● unmanned ● armed ● disarmed ● prearmed ● slow ● fast ● disconnected ● enabled ● disabled ● automatic-release-disabled ● default

Name of the Event Attribute	Type	BACnet Type	Remark
operation_expected	str	BACnetLifeSafetyOperation	Expected operation of the Life-Safety object ASN1: change_of_life_safety.operation-expected ● none ● silence ● silence-audible ● silence-visual ● reset ● reset-alarm ● reset-fault ● unsilence ● unsilence-audible ● unsilence-visual

10.3 Attributes from the CL

10.3.1 Attributes from the Data Point

Name of the Event Attribute	Type	Remark
point_name	str	The value is from the **ois_point_name** column of the subsystem configuration file. This name is also used as the name of the Vu data point.
type_name	str	Language-dependent text of the point type of the Vu data point as defined in the *text_def_vu_point_types.csv* file

10.3.2 Attributes from the Subsystem Configuration File (DpList.xls)

The attributes are defined in the columns with the following names:

- $cu
- $var
- $description

In general, the following attributes are defined dependent on the data point.

Name of the Event Attribute	Type	Remark
text_on	str	Text for an active event This can be, for example, the property `inactive-text` from a BACnet object of the type `binary-input`. **Example:** `text_on="Alarm"`
text_off	str	Text for an inactive event This can be, for example, the property `active-text` from a BACnet object of the type `binary-input`. **Example:** `text_off="Quiet"`

Name of the Event Attribute	Type	Remark
description	str	Description of the data point. This can be, for example, the property `description` from a BACnet object of any type. **Example:** `"Description SECTION"`

10.3.3 Attributes from the Event Definition (event_def.csv)

Name of the Event Attribute	Type	Remark
T1	str	Text for `T1` attribute for GMA Manager 8.x Text for the `#eventSubtype` attribute for Siveillance Control **Example:** `'%(text_on)s'` This text will be resolved to `"Alarm"` when the `text_on`attribute is defined with `"Alarm"`.

10.3.4 Attributes from the Point Type Mapping (point_type_map.csv)

Name of the Event Attribute	Type	Remark
T2	str	Text for `T2` attribute for GMA Manager 8.x Text for the `#eventCategory` attribute for Siveillance Control **Example:** `"%(message_text)s %(description)s"`
T3	str	Text for `T3` attribute for GMA Manager 8.x Text for the `#eventDetails` attribute for Siveillance Control **Example:** `"%(type_name)s*%(point_name)s"`

11 Tasks for Creating a CL

To create a new BACnet CL, the following tasks need to be performed:

1. Siveillance Control MOT information is available in the **statereport.zip** file. Copy the **statereport.zip** file and unzip the html page in a folder of your choice. In this page you can find the Siveillance Control specific MOTs, orders and events for BACnet.
2. Creation of a new site following the instructions in the chapter Example of Creating a Configuration.
3. Adaptation of the **csv** configuration files as described in the chapter *Adapting the CSV Configuration Files* [➙ 63].
4. Definition of the OIS data points and their neighbors in the *DPList.xls* file.

11.1 Adapting the CSV Configuration Files

In this chapter we configure a new point type for connector **vusis**.

Before starting the configuration, you have to create a model of the required point types for the OIS data points.

Example:

OIS point type for a section

Point type PT_SECTION

Parameter		
Point type	**Parameter**	**Value**
	Key name of the point type `key_point_type`	PT_SECTION
	German text for the point type	Bereich/Abschnitt
	English text for the point type	Area/Section
Vu point type connector vusis	**Parameter**	**value**
	Key name of the point type `key_vu_point_type`	VU_PT_SECTION
	Key name of the event definition used for this point type `key_event_map`	EVDEF_SECTION
	Classification of the data point for GMA Manager 8.x (Example of the values: AREA, SECTION ZONE, SENSOR, OTHER)	SECTION
	mot for Siveillance Control	ALX_MOT_FIRE_ZONE (See definitions in *base.py.ois.cl.helper.vu.aloxerv.ALXObjectTypes.py*)

<table>
<tr><th>Parameter</th><th colspan="2"></th></tr>
<tr><td>Cu neighbors of the OIS data point</td><td colspan="2">Here you have to define the name of the placeholder for all Cu data points to be combined to one OIS data point. The data points used for these placeholders are defined later in the DPList.xls file.</td></tr>
<tr><td></td><td>Name of the Placeholder for the Neighbor</td><td>Description</td></tr>
<tr><td></td><td>NEIGHBOR_LS</td><td>life-safety-point object itself
The following commands are available:
• ORDER_RESET_STATION</td></tr>
<tr><td></td><td>NEIGHBOR_BI</td><td>Binary-input object with the state of sabotage</td></tr>
<tr><td></td><td>NEIGHBOR_BO</td><td>Binary-output object
Only required for commands, but provides no events
The following commands are available:
• ORDER_ON
• ORDER_OFF</td></tr>
</table>

11.1.1 point_type_def.csv

Define the key name of the point type, for example **PT_SECTION**.

```
key_point_type
PT_SECTION
```

11.1.2 point_type_map.csv

Define the mapping of the point type. For example, map **PT_SECTION** to the Vu point type **VU_PT_SECTION** and the event definition **EVDEF_SECTION**.

The Vu point type for **VU_PT_SECTION** must be defined in the *vu_point_type_map.csv* file.

The name of the point type for **VU_PT_SECTION** must be defined in the *text_def_vu_point_types.csv* file.

The reaction to the event **EVDEF_SECTION** is defined by the *event_def.csv* file.

The attributes available for the **T2** and **T3** attributes are described in the chapter *String Formatting* [➙ 56].

Several different OIS point types can be mapped to the same Vu point type when using the same value in the **key_vu_point_type** column.

```
key_point_type;key_vu_point_type;key_event_map;classificati
on;T3;T2
PT SECTION;VU PT SECTION;EVDEF SECTION;SECTION;"%(type
name)s*%(point name)s";"%(message_text)s %(description)s"
```

11.1.3 vu_point_type_map.csv

Define the point type for the Vu-Handler. This is the data point type seen by the consumer (for example, GMA Manager 8.x).

```
key_vu_point_type;point_type
VU_PT_SECTION;6
```

NOTICE	
!	**While integrating BACnet CL with Siveillance Control, the default value of 41 is assigned to the vu data point type VU_PT_SYSTEM. While integrating BACnet CL with GMA Manager, the default value of 65 is assigned to the vu data point type VU_PT_SYSTEM.** Communication errors occur if **VU_PT_SYSTEM** is not equal to the default value set by the system. 1. Avoid using 41 for any other vu point types for example VU_PT_SECTION in Siveillance Control. 2. Avoid using 65 for any other vu point types for example VU_PT_SECTION in GMA Manager.

11.1.4 text_def_vu_point_types.csv

Define the language-dependent name of the point type.

```
key;de_DE;en_GB
VU_PT_SECTION;Sektion;Section
```

11.1.5 event_def.csv

Define the reactions to events from the Cu-Handler.

The expression is the most complicated part of the CL. It is required for the definition of all possible events. An error will occur if no event is found with an expression with the `result==True`. If you do not know if you have defined all possible events, define a reaction to an event with `expression=True` in the last line.

```
key_event_map;placeholder;expression;transmit;message_code;
T1
# ------ Event Definition for SECTION ------
EVDEF_SECTION;NEIGHBOR_LS;active==True;True;set alarm;'%
(text_on)s'
EVDEF_SECTION;NEIGHBOR_LS;active==False;True;reset alarm;'%
(text_off)s'
EVDEF_SECTION;NEIGHBOR_BI;active==True;True;set sabotage
alarm;'%(text_on)s'
EVDEF_SECTION;NEIGHBOR_BI;active==False;True;reset sabotage
alarm;'%(text_off)s'
```

Example of the Expression of a Life-Safety Object:

```
to_event_state=='offnormal' and value in ['supervisory',
'test_supervisory'] and new_mode=='on
```

11.1.6 order_def.csv

Define the orders available for the point type. If several point types use the same orders, use templates to avoid duplicate definitions.

```
key_point_type;key_order;user_group;basic;placeholder;attri
bute;value
Template_LifeSafety;ORDER_RESET_STATION;1;0;NEIGHBOR_LS;LSO
P;'reset'
PT_SECTION;Template_LifeSafety

# Example for 2 orders, each with 2 reactions
PT_SECTION;ORDER_OFF;1;0;NEIGHBOR_BO;present-value;0
```

```
PT_SECTION;ORDER_OFF;;;NEIGHBOR_LS;mode;'off'

PT_SECTION;ORDER_ON;1;0;NEIGHBOR_BO;present-value;1
PT_SECTION;ORDER_ON;1;0;NEIGHBOR_LS;mode;'on'
```

11.1.7 order_map.csv

Define the order numbers for the orders. These are, for example, the order numbers used by GMA Manager 8.x.

```
key_order;order_nr
ORDER_OFF;11
ORDER_ON;12
ORDER_RESET_STATION;13
```

11.1.8 text_def_order_names.csv

Define the language-dependent names of the orders. These are the names of the orders visible at GMA Manager 8.x.

```
key;de_DE;en_GB
ORDER_OFF;Abschalten;Off
ORDER_ON;Zuschalten;On
ORDER_RESET_STATION;Rückstellen;Reset
```

11.2 Defining OIS Data Points

The OIS data points and their Cu neighbors have to be defined in an Excel file. In general, this file has the name *DPList.xls*. *DPList .xls* is created automatically when a gateway is created using a BACnet template as described in the chapter Creating a Gateway. For further information, refer to the chapter *Subsystem Configuration File (DPList.xls)* [➙ 37].

12 Tips for Converting OIS 2.3 CL

This chapter describes how to convert an OIS 2.3 CL into an OIS 2.6.1 CL.

The main tasks are as follows:

- Definition of the point type
- Definition of the mapping of events
- Definition of the orders

12.1 Converting Point Types

12.1.1 Example OIS 2.3 CL

In OIS 2.3, the data point is defined as follows:

```
point_types
{
  sis_point_type 90 type_name "Kommunikation"
  ...
}
```

In OIS 2.6.1, additional steps are required for defining data point types. A Vu point type definition and event definition can, however, be used for several point types if required.

12.1.2 Defining Key Names

The key name **PT_KOMMUNIKATION** is used for this OIS point type, which references its Vu point type by the key **VU_PT_KOMMUNIKATION** and its event definition by the key **EVDEF_KOMMUNIKATION**.

In OIS 2.6.1, this point type needs to be defined and mapped in several files.

12.1.3 point_type_def.csv

Define the key name of the OIS data point.

This key name of the point type is used for the point type name of the OIS data point.

```
key_point_type
PT_KOMMUNIKATION
```

12.1.4 point_type_map.csv

This file defines the connector-dependent mapping for this point type.

Here we have to define the key names used to reference the entries in the mapping files *text_def_vu_point_types.csv* and *vu_pointtype_map.csv*.

The **mot** (managed object type) for the point type needs to be clarified with the Siveillance Control team. It will be visible in the export file with the Vu data points.

```
key_point_type;key_vu_point_type;key_event_map;mot
PT KOMMUNIKATION;VU PT KOMMUNIKATION;EVDEF
KOMMUNIKATION;ALX MOT BMT TRANSMISSION_UNIT_ALARM
```

12.1.5 text_def_vu_point_types.csv

This file defines the language-dependent name of the point type that is visible in GMA Manager 8.x.

```
key;de_DE
VU_PT_KOMMUNIKATION;Kommunikation
```

12.1.6 vu_point_type_map.csv

This file defines the connector-dependent Vu point type that is visible at the management station.

```
key_vu_point_type;point_type
VU_PT_KOMMUNIKATION;90
```

12.2 Converting Events

12.2.1 Differences between OIS 2.3 and OIS 2.6.1

OIS 2.3 Group and Entry:

In the event definition for OIS 2.3, the `group` and `entry` of the event need to be defined. For this purpose, an event reaction needs to be defined for every possible combination of `group` and `entry`.

In OIS 2.3, all possible state transitions between `group` and `entry` had to be defined.

OIS 2.6.1 Group and Entry:

In the event definition for OIS 2.6.1, it is not necessary to define the `group` and `entry` of the event. The `to_event_state` attribute in the **expression** column can be used instead of defining the value for `entry`.

Using the value for `group` is not required because the reaction to the event is independent of the previous state. See also section *Reset the Last Event* below.

OIS 2.3 Reset the Last Event:

The last event has to be reset with the message code `quiet` or the respective reset code.

OIS 2.6.1 Reset the Last Event:

The CL of OIS 2.6.1 automatically sends the reset code for the last event when a new event occurs. If several Cu data points are merged in one OIS data point, then resetting of the event is handled per Cu data point.

12.2.2 Restrictions

For GMA Manager 8.x and Siveillance Control Pro you have to be careful not to use the same message code for the events from different Cu data points that are merged in one OIS data point. These management systems do not support several event entries in the same list for the same data point.

Example:

The BACnet binary input objects `binary-input 1` and `binary-input 2` are merged together in one OIS data point. Here it is not allowed to define, for example, the same message code `set alarm` for both Cu data points.

Define, for example, the message code `set alarm` for `binary-input 1` and `set message` for `binary-input 2`.

12.2.3 Example OIS 2.3 CL

Note:
MSG ("quiet", "Ruhe") could also be used to reset the previous event.

```
#define ES(dpname) cu_point_type "00" point_name dpname att
ribute_name event_state
#define EVSTATE(dpname, evstate) (state == run) ||
 (state == update && ES(dpname) == evstate)
#define EVRULE(dpname, fromstate, tostate) event_number fro
mstate tostate dpname
#define MSG(msg, txt) sis_message msg text1 txt
#define EVENT(dpname, fromstate, tostate, setmsg, settext)
\
  EVRULE(dpname, fromstate, tostate) \
  if (EVSTATE(dpname, tostate)) \
    MSG(setmsg, settext)

point_types
{
 sis_point_type 01 type_name "Einzelmelder"
 {
  events
  {
    // Voralarm und Test-Alarm unterschiedlich abbilden
    EVRULE(@ZUSTAND, BN_ES_NORMAL, BN_ES_OFFNORMAL)  \
    if(EVSTATE(@ZUSTAND, BN_ES_OFFNORMAL)) \
    { \
        if(PV(@ZUSTAND) == 2) \
        { \
            MSG("set pre alarm", "Voralarm") \
        } \
        else \
        { \
            MSG("set inspection message", "Inspektionsalarm
") \
        } \
    }
    EVRULE(@ZUSTAND, BN_ES_OFFNORMAL, BN_ES_NORMAL) \
    if(EVSTATE(@ZUSTAND, BN_ES_NORMAL)) \
    { \
        MSG("reset pre alarm", "Ruhe") \
        MSG("reset inspection message", "Ruhe") \
    }
    EVENT(@ZUSTAND, BN_ES_NORMAL, BN_ES_FAULT, "set message
", "Datenpunktstörung")
    EVENT(@ZUSTAND, BN_ES_OFFNORMAL, BN_ES_FAULT, "set mess
age", "Datenpunktstörung")
    EVRULE(@ZUSTAND, BN_ES_FAULT, BN_ES_OFFNORMAL)  \
    if(EVSTATE(@ZUSTAND, BN_ES_OFFNORMAL)) \
    { \
        MSG("reset message", "Ruhe") \
        if(PV(@ZUSTAND) == 2) \
        { \
            MSG("set pre alarm", "Voralarm") \
        } \
        else \
        { \
            MSG("set inspection message", "Inspektionsalarm
") \
        } \
    }
    EVRULE(@ZUSTAND, BN_ES_FAULT, BN_ES_NORMAL)  \
```

```
    if(EVSTATE(@ZUSTAND, BN_ES_NORMAL)) \
    { \
        MSG("reset message", "Ruhe") \
        MSG("reset inspection message", "Ruhe") \
        MSG("reset pre alarm", "Ruhe") \
    }
  }
 }
}
```

12.2.4 File event_def.csv

In OIS 2.6.1, only the event definition for the new event state needs to be defined.

The previous state is reset automatically. It is also not necessary to distinguish between the states **update** and **run** for the event.

```
key_event_map;placeholder;group;entry;expression;message_co
de;T1
EVDEF_EINZELMELDER;ZUSTAND;;;to_event_state=='offnormal'
and value==2;set pre alarm;"Voralarm"
EVDEF_EINZELMELDER;ZUSTAND;;;to_event_state=='offnormal';se
t inspection message;"Inspektionsalarm"
EVDEF_EINZELMELDER;ZUSTAND;;;to_event_state=='fault';set
message;"Datenpunktstörung"
EVDEF_EINZELMELDER;ZUSTAND;;;to_event_state=='normal';quiet
;"Ruhe"
```

12.3 Example of Converting Orders

Example of OIS 2.3 for an order with several reactions:

```
ois_point_type 06 type_name "Sektion"
{
  orders {
    ois_order 401 "Alarm rücksetzen" user_group 2 { \
    set cu_point_type "00" point_name @CU_ORDER attribute_n
ame present_value = 0 \
    set cu_point_type "00" point_name @CU_ORDER attribute_n
ame present_value = 1 \
    set cu_point_type "00" point_name @CU_ORDER attribute_n
ame present_value = 0 \
  }
}
```

12.3.1 File order_def.csv

The **Reset alarm** order is defined with the key name **ORDER_RESET_ALARM**, and three reactions are defined for it.

```
key_point_type;key_order;user_group;basic;placeholder;attri
bute;value
PT_SECTION;ORDER_RESET_ALARM;2;0;CU_ORDER;present-value;0
PT_SECTION;ORDER_RESET_ALARM;;;CU_ORDER;present-value;1
PT_SECTION;ORDER_RESET_ALARM;;;CU_ORDER;present-value;0
```

12.3.2 File text_def_order_names.csv

Example of an order with the German text for the order name:

```
key;de_DE
ORDER_RESET_ALARM;Alarm rücksetzen
```

12.3.3 File order_map.csv

Example of connector **vusis** for defining the order number:

```
key_order;order_nr
ORDER_RESET_ALARM;401
```

Example of connector **aloxerv** for defining the order number:

```
key_order;order_nr
ORDER_RESET_ALARM;ALX_ORDER_TBS_OFF
```

13 Parameters of the BACnet Cu-Handler

This chapter describes all parameters that are available for the Cu-Handler and their default values.

If a parameter has not been configured, the default value will apply.

13.1 General Parameters

Parameter Name	Default Value	Type	Description	Required
proxyhostname	-		Remote Cu proxy for proxy protocol port forwarding. [{username}[:{password}]@]{hostname}[:{port}]/{localport}[:{remoteport}] Example: kh1wm378@127.0.0.1/10024:20048 Note: This is required if the Cu-Handler is running on a separate computer (remote).	Yes
configfile	etc/import/DPList.xls		Name of the Excel file with data point definitions Note: Name of the subsystem configuration file	No
vu_neighbours	List of all configured Vu-Handlers		List of the unit names of the Vu-Handlers that are connected to the Cu-Handler. Multiple unit names are separated by comma. Example: Vu_1, VU_2 Note: List of the connected Vu-Handlers	No
cl_name	-		Name of the directory with the configuration files for the CL in the *etc* directory of the **plc** unit.	Yes

13.2 Specific Parameters

13.2.1 Vu-Handler Specific Parameters

The following parameters are not available with all Vu-Handlers:

Parameter Name	Default Value	Type	Description	Required
no_reset_inspection	True		**True:** Inspection alarms remain in the **MAINTENANCE / Insp. alarm** list as long as the detector is in inspection mode. At the end of the inspection, the entries in the **MAINTENANCE / Insp. alarm** und **MAINTENANCE / Inspection** lists are deleted automatically. **False:** Only the event that was received last is displayed. Note: Only relevant for GMA Manager 8.x	Yes

13.2.2 Configuring the BACnet Stack

In general, the default values are used for these parameters. These values should only be changed by a BACnet expert.

Parameter Name	Default Value	Type	Description	Required
apdu_timeout	10000	int	Timeout of a BACnet transmission in milliseconds The APDU timeout time in the BACnet device must be set to the same value. The BACnet device will then receive a sign of life twice within the client timeout time. Minimum 2000	No
number_of_apdu_retries	3	int	Number of repetitions when transmitting BACnet telegrams (0 for no repetition)	No

Parameter Name	Default Value	Type	Description	Required

segmentation_supported	0	int	Support of segmented data transmission
			`0 = segmented_both`
			`1 = segmented_transmit`
			`2 = segmented_receive`
			`3 = no_segmentation`
			This parameter depends on the capabilities of the connected BACnet device.
				No

apdu_segment_timeout	5000	int	Timeout of the segmented BACnet data transmission in milliseconds. This value must be smaller than **apdu_timeout**. Minimum 2000	No
max_segments_accepted	128	int	Maximum number of segments in a BACnet transmission. This value must be high enough to ensure that long object lists and event information can be read.	No

13.2.3 Configuring the BACnet Stack Interface

A specific configuration file is provided for the BT BACnet stack. The parameters can however also be configured using the Cu-Handler parameters.

Parameter Name	Default Value	Type	Description	Required

bbs_interface	-	int	BBS interface **<nic>[:<udp>]**
			Example 1:
			The default BACnet UDP port is used.
			`eth0`
			Example 2:
			eth0 interface with UPD port 47809
			`eth0:47809`
				No

bbs_bbmd	-	str	List of BBMD entries
			`"<ip>[:<port>[:<mask>]],..."`
			The **mask** value is generally not used.
			Example:
			`192.168.1.2:47809,192.168.2.3:47809`
				No

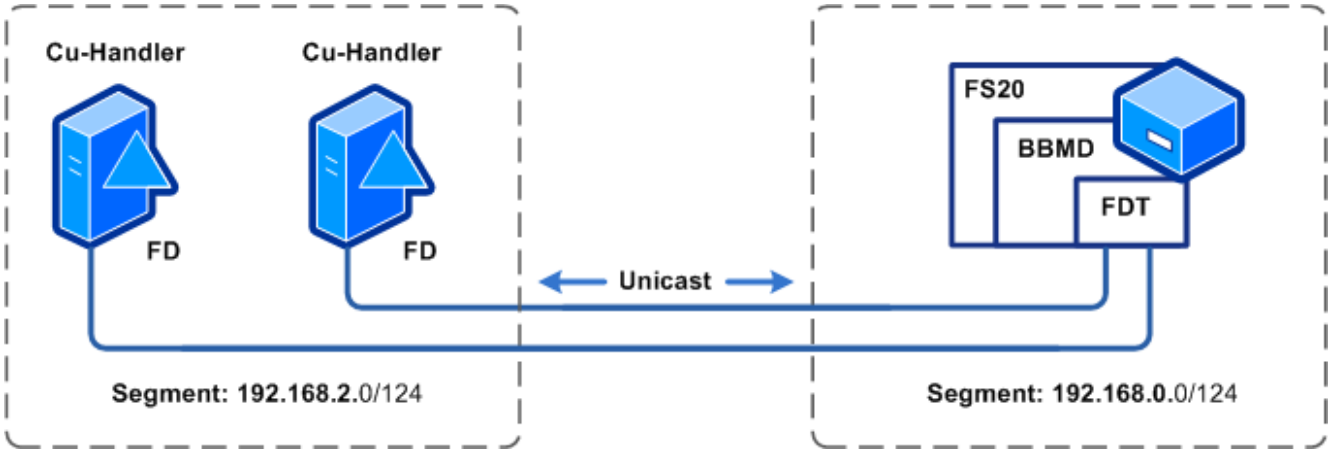
13.2.4 Configuring the Foreign Device

A foreign device is a BACnet device. The IP subnet address of the foreign device differs from the IP subnet addresses of all the devices in the BACnet / IP network the foreign device attempts to communicate with. With the foreign device procedure, remote devices can dynamically link with a BACnet configuration. Specific BACnet services are available for creating an entry in the Foreign Device Table (FDT).

The foreign device functionality is used in the following cases:

- For the redundant configuration of Cu-Handlers that connect to the same BACnet device. The redundant functionality is not possible with a BBMD if the redundant Cu-Handler is located in the same network segment.
- If the Cu-Handler and the BACnet device are in different network segments.
- To avoid broadcast messages between the Cu-Handler and the BACnet device in the network.

Example of a Configuration



In a system with FDT functionality, a BACnet device assumes the functionality of the BBMD. The Cu-Handlers register periodically at this BACnet device as foreign devices.

In case of a BACnet broadcast these UDP packets are sent to the BBMD defined with **bbs_fd_bbmd** using a unicast. The BBMD forwards the UDP packets to the appropriate recipient using a unicast (or broadcast).

Parameter Name	Default Value	Type	Description	Required
bbs_fd_bbmd		str	IP address of the BBMD where this Cu-Handler registers with the lifetime configured using the **bbs_fd_lifetime** parameter The FD functionality is activated by this parameter. The BBMD functionality of this Cu-Handler is then inactive. The **bbs_bbmd** parameter must not be configured in this case.	No
bbs_fd_lifetime	900 seconds = 15 minutes		Lifetime of the registration of a Foreign Device in seconds. Registration at the BACnet device defined by the **bbs_fd_bbmd** parameter takes place periodically.	No

13.2.5 Configuring the BACnet Client (Cu-Handler)

Parameter Name	Default Value	Type	Description	Required
supervision	Calculated minimum value	int	The **supervision** time (in seconds) determines the delay after which a line failure is reported. The value must at least be **(number_of_apdu_retries + 1)* apdu_timeout + 1**. Otherwise a connection error will be detected before all repetitions of a message have been sent by the BACnet stack. If the value set is too small, it will be corrected automatically. Short monitoring times increase the base load in the network. It has to be taken into account that all servers (BACnet devices) are monitored individually. The BACnet server monitoring time is calculated internally ((**supervision**-2)/2 seconds) and cannot be set separately. The BACnet client monitoring time of the BACnet device must be at least as long as the **supervision** time.	Yes
my_device_instance	-	int	Instance number of the BACnet device of the Cu-Handler	Yes
port	1	int	BBS port ID This is the port ID of the BACnet stack.	No

Parameter Name	Default Value	Type	Description	Required
process_identifier	1	int	Process identifier for the following BACnet requests from the Cu-Handler to the BACnet devices: ● BACnetLifeSafetyOperation ● AcknowledgeAlarm_Request ● AddListElement_Request ● RemoveListElement_Request	No
register_notification_class	True	bool	**True**: The Cu-Handler uses the BACnet services **AddListElement / RemoveListElement** to sign on / off from the BACnet devices as the recipient of events. **False**: No registration. Configuration must be made in the BACnet device.	No
subscribe_cov	-	str	Defines the name of the file with the definition of the COV subscriptions (generally *etc/subscribe_cov.csv*)	No
subscribe_cov_cycle		int	Cycle time of the COV subscriptions in seconds	No
support_date_time	True	bool	**True**: The time of the BACnet device can be read from the optional BACnet poperties "local-date" and "local-time". **False:** The BACnet device does not have its own time. Reading of the date and time from the BACnet properties "local-date" and "local-time" is disabled.	

13.2.6 Configuring a BACnet Device

The parameters of a BACnet device are listed in the table below.

Only the **time_server** parameter needs to be set. All other parameters are overwritten with the values that are retrieved from the data channel, provided that these values were configured. The data channel reads these values from the Excel file.

Parameter Name	Default Value	Type	Description
peer_device_instance	-	int	Instance number of the BACnet device The data channel reads this value from the Excel file (subsystem configuration file).
peer_device_address	-	str	IP address of the BACnet device This value is usually not required. It is determined using theBACnet services **Who-Is** and **I-Am**.
time_server	False	bool	True: Activation of time synchronization with the BACnet device This BACnet device must act as a time server for the other BACnet devices. Time synchronization takes place as follows: ● After the BACnet device is connected to the Cu-Handler ● At 02:00 hours ● At 3:00 hours Correct time synchronization is mandatory; otherwise events that occurred prior to the update may be lost to the management system. If the Cu-Handler is running on a separate computer, the time of this computer must be synchronized using the NTP service (Linux service).

14 Appendix

14.1 Tips for Analyzing Traces

The following tables describe the text to be searched for in the traces for the update, orders and events.

14.1.1 Errors

Search for	Description	Debug Level
Traceback	Finds all exceptions. If everything is fine, then no exception should occur.	-
ERROR	Error messages from the CL	ERROR

14.1.2 Update

Search for	Description	Debug Level
UPDATE_BEGIN	Start of the update of a component	INFO
UPDATE_END	End of the update of a component	INFO

14.1.3 Events

Search for	Description	Debug Level
event update_event = PyPlcEventAlt	Update event received from the Cu-Handler	INFO2
event = PyPlcEventAlt	Event received from the Cu-Handler	
: EventReactionDef	Line number of the event definition and its values from *event_def.csv*	DEBUG
Attributes for connector	Attributes after evaluating the event definition and string replacement	DEBUG
send to Vu update	Event and its attributes which were sent to the Vu-Handler during the update	INFO
send to Vu event	Event and its attributes which were sent to the Vu-Handler	INFO
add_event=	Global unique number of the event	DEBUG

14.1.4 Acknowledge Events

Search for	Description	Debug Level
on_acknowledge	Cu-Handler Trace Acknowledge command and its local unique number	INFO
dispatch_acknowledge_notification	Cu-Handler Trace Acknowledge notification received from the BACnet device	DEBUG
Acknowledge Notification	Acknowledge notification and its global and local unique number	DEBUG

14.1.5 Orders

Search for	Description	Debug Level
on_order	Orders received from the Vu-Handler	INFO
set_unit_attribute	Order to the BACnet Cu-Handler	DEBUG

14.1.6 COV Subscriptions

Search for	Description	Debug Level
SubscribeCOV	COV subscriptions	INFO DEBUG
CL on_event (100, 100)	Event from COV notifications	INFO

14.2 Supported Message Codes GMA Manager 8.x

The following table defines the supported message codes of OIS 2.4 and the list name of the GMA Manager 8.x where events with such a message code will be displayed.

Search for	Description
`quiet`	-
`set alarm`	MELDERGRUPPE Alarm
`reset alarm`	MELDERGRUPPE Alarm
`set emergency call`	MELDERGRUPPE Alarm
`reset emergency call`	MELDERGRUPPE Alarm
`set sabotage alarm`	MELDERGRUPPE Alarm
`reset sabotage alarm`	MELDERGRUPPE Alarm
`set intrusion alarm`	MELDERGRUPPE Alarm
`reset intrusion alarm`	MELDERGRUPPE Alarm
`set internal alarm`	MELDERGRUPPE Alarm
`reset internal alarm`	MELDERGRUPPE Alarm
`set silent alarm`	MELDERGRUPPE Alarm
`reset silent alarm`	MELDERGRUPPE Alarm
`silent alarm`	MELDERGRUPPE Alarm
`set technical alarm`	MELDERGRUPPE Alarm
`reset technical alarm`	MELDERGRUPPE Alarm
`set system alarm`	MELDERGRUPPE Alarm
`reset system alarm`	MELDERGRUPPE Alarm
`set loesch alarm`	MELDERGRUPPE Alarm
`reset loesch alarm`	MELDERGRUPPE Alarm
`set pre alarm`	MELDERGRUPPE Voralarm
`reset pre alarm`	MELDERGRUPPE Voralarm
`set investigation`	MELDERGRUPPE Voralarm
`reset investigation`	MELDERGRUPPE Voralarm
`set pre warning`	MELDERGRUPPE Vorwarnung
`reset pre warning`	MELDERGRUPPE Vorwarnung

Search for	Description
set message	MELDERGRUPPE Meldung
reset message	MELDERGRUPPE Meldung
set message 1	MELDERGRUPPE Meldung
reset message 1	MELDERGRUPPE Meldung
set message 2	MELDERGRUPPE Meldung
reset message 2	MELDERGRUPPE Meldung
set message 3	MELDERGRUPPE Meldung
reset message 3	MELDERGRUPPE Meldung
set message 4	MELDERGRUPPE Meldung
reset message 4	MELDERGRUPPE Meldung
set detector fault	MELDERGRUPPE Störung
reset detector fault	MELDERGRUPPE Störung
set detector fault 1	MELDERGRUPPE Störung
reset detector fault 1	MELDERGRUPPE Störung
set detector fault 2	MELDERGRUPPE Störung
reset detector fault 2	MELDERGRUPPE Störung
set detector fault 3	MELDERGRUPPE Störung
reset detector fault 3	MELDERGRUPPE Störung
set detector fault 4	MELDERGRUPPE Störung
reset detector fault 4	MELDERGRUPPE Störung
set detector off	MELDERGRUPPE Abgesch.
reset detector off	MELDERGRUPPE Abgesch.
set detector off Alarm	MELDERGRUPPE Abgesch.
reset detector off Alarm	MELDERGRUPPE Abgesch.
set detector off Fault	MELDERGRUPPE Abgesch.
reset detector off Fault	MELDERGRUPPE Abgesch.
set remote transmission	UE / ALARMIERUNG Auslösung
reset remote transmission	UE / ALARMIERUNG Auslösung
set ue	UE / ALARMIERUNG Auslösung
reset ue	UE / ALARMIERUNG Auslösung
set remote transmission fault	UE / ALARMIERUNG Störung
reset remote transmission fault	UE / ALARMIERUNG Störung
set remote transmission off	UE / ALARMIERUNG Abgesch.
reset remote transmission off	UE / ALARMIERUNG Abgesch.
set external device	UE / ALARMIERUNG Auslösung

Search for	Description
reset external device	UE / ALARMIERUNG Auslösung
set external device fault	UE / ALARMIERUNG Störung
reset external device fault	UE / ALARMIERUNG Störung
set external device off	UE / ALARMIERUNG Abgesch.
reset external device off	UE / ALARMIERUNG Abgesch.
set control	STE / RAL Auslösung
reset control	STE / RAL Auslösung
set control fault	STE / RAL Störung
reset control fault	STE / RAL Störung
set control off	STE / RAL Abgesch.
reset control off	STE / RAL Abgesch.
set evacuation	STE / RAL Auslösung
reset evacuation	STE / RAL Auslösung
set evacuation fault	STE / RAL Störung
reset evacuation fault	STE / RAL Störung
set evacuation off	STE / RAL Abgesch.
reset evacuation off	STE / RAL Abgesch.
set system message	SYSTEM Meldung
reset system message	SYSTEM Meldung
set system fault	SYSTEM Störung
reset system fault	SYSTEM Störung
set system fault 1	SYSTEM Störung
reset system fault 1	SYSTEM Störung
set system fault 2	SYSTEM Störung
reset system fault 2	SYSTEM Störung
set system fault 3	SYSTEM Störung
reset system fault 3	SYSTEM Störung
set system fault 4	SYSTEM Störung
reset system fault 4	SYSTEM Störung
set system off	SYSTEM Abgesch.
reset system off	SYSTEM Abgesch.
set maintenance	SYSTEM Wart.Anfo
reset maintenance	SYSTEM Wart.Anfo
set power fault	SYSTEM Stör.SV
reset power fault	SYSTEM Stör.SV

Search for	Description
set power fault 1	SYSTEM Stör.SV
reset power fault 1	SYSTEM Stör.SV
set power fault 2	SYSTEM Stör.SV
reset power fault 2	SYSTEM Stör.SV
set power fault 3	SYSTEM Stör.SV
reset power fault 3	SYSTEM Stör.SV
set power fault 4	SYSTEM Stör.SV
reset power fault 4	SYSTEM Stör.SV
set change	SYSTEM Proj.Änd.
reset change	SYSTEM Proj.Änd.
set manned	SYSTEM Proj.Änd
reset manned	SYSTEM Proj.Änd
set inspection message	WARTUNG Insp.Meld.
reset inspection message	WARTUNG Insp.Meld.
set inspection	WARTUNG Inspektion
reset inspection	WARTUNG Inspektion
set information	WARTUNG Auskunft
reset information	WARTUNG Auskunft
set external armed	BEREICHE Scharf
set external disarmed	BEREICHE Scharf
set internal armed	BEREICHE Unscharf
set internal disarmed	BEREICHE Unscharf
set revision	BEREICHE Revision
reset revision	BEREICHE Revision
set walktest	BEREICHE Geh-Test
reset walktest	BEREICHE Geh-Test

The following table describes the message codes from OIS 2.3 which are **not** supported by OIS 2.4.2. These message codes are alias definitions for other message codes.

Message Code 2.3 Not supported by OIS 2.4.2	Message Node OIS 2.4.2
set ue fault	set remote transmission fault
reset ue fault	reset remote transmission fault
set ue off	set remote transmission off
reset ue off	set remote transmission off
set ala off	set external device off

Message Code 2.3 Not supported by OIS 2.4.2	Message Node OIS 2.4.2
reset ala off	reset external device off
set ala	set external device
reset ala	reset external device
set ala fault	set horn fault
reset ala fault	reset horn fault
set evac	set evacuation
reset evac	reset evacuation off
set evac off	set evacuation off
reset evac off	reset evacuation off
set evac fault	set evacuation fault
reset evac fault	reset evacuation fault

14.3 Example of the Implementation of an Event Callback Function

The following code shows the implementation of the event callback function **callback_example** in the start file *__init__.py* of the CL.

Implementation of the callback function referenced in *event_def.csv* (see chapter *Event Definition (event_def.csv)* [➙ 49]):

```
class BACPointTypeApplication(BACPointTypeCommon):
...
    def evcb_callback_example(self, on_event_param, attribu
te_dict, value_1, value_2,
                              var_name_result='result'):
        """
        The name of this function shall begin with evcb_  (
event callback)
        This function is called for an event when the colum
n call_function in
        the file event_def.csv references this function.
        The parameters self, on_event_param, attribute_dict
 are required as the
        first 3 parameters and may never be changed. The fo
llowing parameters
        value_1, value_2 are application specific and shall
 be the parameters
        of the callback function callback_example defined i
n the column
        call_function in the file event_def.csv.

        Example of the configuration of column call_functio
n in event_def.csv::

            "callback_example(value_1=(%value)d, value_2=11
)"

        :param on_event_param: Parameter of the event
        :type on_event_param: OnEventParamConnector

        :param attribute_dict: Attributes of the event befo
```

```
re formatting.
                                You can add additional attri
butes, but never
                                change the content of existi
ng attributes
        :type attribute_dict: {<attr_name>: <value>}

        :param value_1: First parameter
        :type value_1: str,int,float

        :param value_2: Second parameter
        :type value_2: str,int,float

        :param var_name_result: Name of the variable with t
he result
        :type var_name_result: str
        """
        self.logger.info(">> callback_example(value_1=%r, v
alue_2=%r)",
                         value_1, value_2)

        # Resolve the placeholders in the parameters
        my_dict = {
            'value_1' : value_1,
            'value_2' : value_2,
        }
        my_dict = on_event_param.format_attributes(my_dict,
                                                   attribut
e_dict,
                                                   True)
        value_1 = float(my_dict['value_1'])
        value_2 = float(my_dict['value_2'])

        result = value_1 * value_2
        attribute_dict[var_name_result] = result
        self.logger.info("<< callback_example result=%r)",
result)
        return attribute_dict
```

14.4 Example of how to Use a Timer in the Event Callback Function

Sometimes it is required to supervise, for example, if an input remains active longer than a given time before initiating an event. In general this will be done using the BACnet Timer object. If this is not possible, you have to implement this functionality with an event callback function.

The following code shows the implementation of the timer in the start file *__init__.py* of the CL in the class **BACPointTypeApplication**. Here we assume that we supervise the state 10 of a BACnet multistate-value-object.

14.4.1 Definition in the event_def.csv File

The following lines in the *event_def.csv* file are required for the event definition of our timer example.

```
key_event_map;group;entry;placeholder;expression;transmit;m
essage_code;T1;call_function
EVDEF_MULTI_STATE;;;;value==10;False;;;amx_disturbed(delay=
```

```
3)
EVDEF_MULTI_STATE;;;;value==10;False;;;amx_disturbed(delay=
3, message_text='STOERUNG-AMX')
```

The event of the multistate-value-object with the state 3 is handled by the second line:

```
EVDEF_MULTI_STATE;;;;value==10;False;;;amx_disturbed(delay=
3)
```

This line calls the callback function **amx_disturbed** if the state value is 10. The callback function is implemented by the function **BACPointTypeApplication.evcb_amx_disturbed**.

The timer callback function **__on_timer_amx_disturbed** initiates an event with `group=101 entry=1` which is handled by the third line:

```
EVDEF_MULTI_STATE;101;1;;True;True;set system fault;%
(message_text)s;
```

This line initiates an event with the message code **set system fault** and uses the **message_text** attribute from the event for the text **T1**.

14.4.2 Initializing the Timer

Here we initialize a timer with the name **amx_disturbed** with the callback function **__on_timer_amx_disturbed**.

```
     def on_start(self):
        super(BACPointTypeApplication, self).on_start()
        self.add_timer("amx_disturbed",
                       BACPointTypeApplication.__on_timer_a
mx_disturbed)
```

14.4.3 Starting the Timer in the Event Callback Function

Here we start the timer with the given delay.

```
    def evcb_amx_disturbed(self, on_event_param, attribute_
dict, delay, message_text):
        self.logger.info("AMX_DISTURBED start the timer")
        if self.timer_amx_disturbed is None:
            timer = self.new_timer("amx_disturbed",
                                   logic_datapoint = self.o
is_datapoint)

        timer.set_attribute(CuAttributes.active, 1) # Must
be an active event
        timer.set_attribute(BACnetAttributes.message_text,
message_text)
        timer.set(duration=delay)
        self.timer_amx_disturbed = timer
        return attribute_dict
```

14.4.4 Handler Function when the Timer Expires

The following function shows an example of the implementation of the timer callback function. In this function we read the **present_value** attribute from the BACnet object. If the value is still 10, an exception is sent to our data point. This event is handled as if it had been sent directly from the Cu-Handler.

```
    @defer.oisInlineCallbacks()
    def __on_timer_amx_disturbed(self, timer):
        self.logger.info(">> AMX_DISTURBED timer expired")
        # Shall be a name of a cu_placeholder of the data p
oint from the
        # DpList file.
        placeholder = 'CU'
        attribute = BACnetAttributes.present_value
        present_value = yield self.get_unit_attribute(place
holder, attribute)
        self.logger.info("AMX_DISTURBED present-
value=%r", present_value)

        if present_value == 10:
            self.logger.info("AMX_DISTURBED send fault")
            datapoint = self.this_datapoint
            group_entry = (101, 1) # See also event_def.csv
            attribute_dict = timer.get_attributes()
            event = self.new_event(datapoint, group_entry,
placeholder)

            for attribute, value in attribute_dict.iteritem
s():
                event.set_attribute(attribute, value,
                                    attribute_type=Attribut
e.TypeMapAttribute)

            event.send()
        self.timer_amx_disturbed = None
        self.logger.info("<< AMX_DISTURBED timer expired")
```

15 Excel Tips

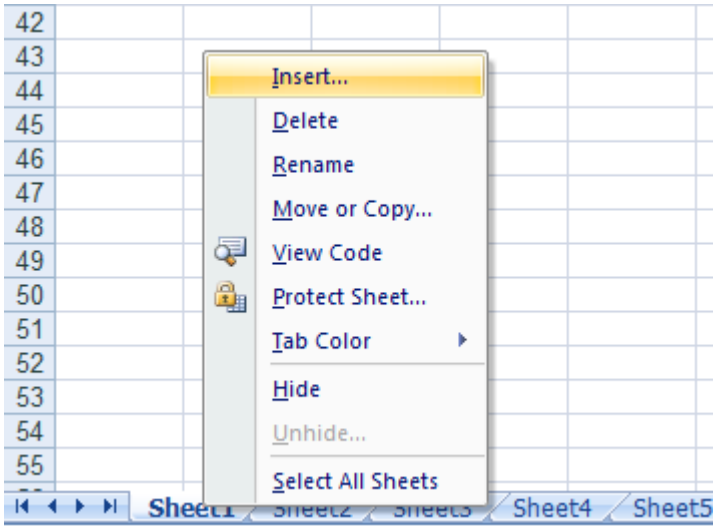
15.1 Named Reference

The Named Reference defines the values available in the **drop-down cell**.

15.1.1 Creating a Table for the Reference

To create a table for the reference, the following steps are required:

1. Click the icon  .
   – or
2. Right-click **Table** and select **Insert** in the context menu.



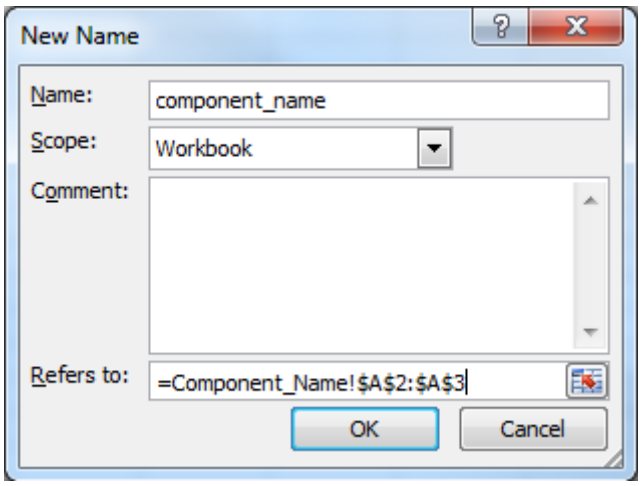
3. Save the newly created table sheet, for example with the name *Componet_Name.xls*.
   ⇨ A new table sheet has been created and saved.
4. Define the header in the first row, for example **component_name**.
5. In the rows that follow, define the values available for the reference.

Tip: To freeze a header row, select **View > Freeze Panes > Freeze Top Row** in the menu bar.

⇨ The table for the reference has been created.

15.1.2 Creating a Named Reference

1. Select a column, for example **component_name**.
2. Select **Formulas > Name Manager** in the menu bar.
3. Click **New**.
   ⇨ The following dialog box opens:



4. Click the icon .
5. Select the rows to be used.
6. Click the icon .
   ⇨ The values are applied.

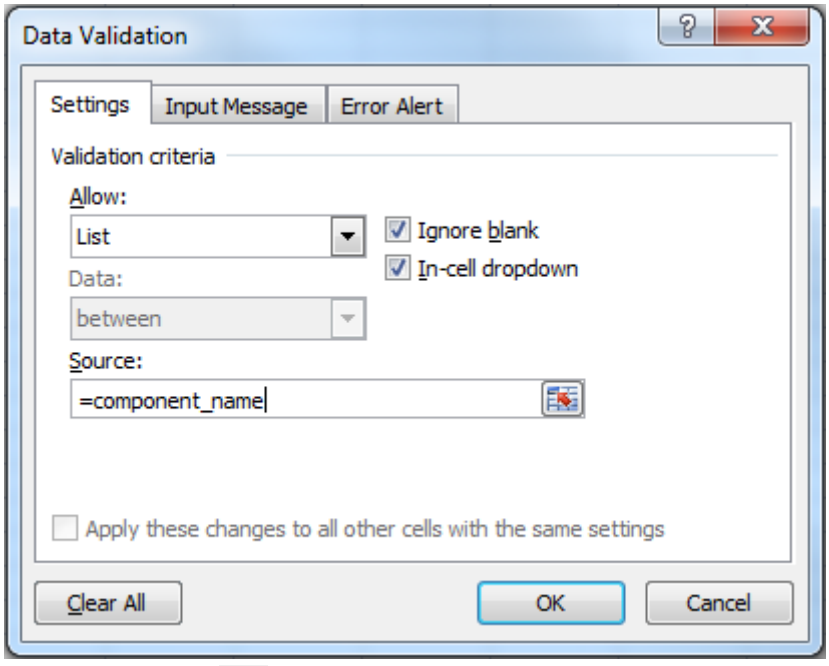
NOTICE	
!	Make sure that the new value overwrites the old value and that it is not just added.

7. Click **OK**.

15.1.3 Creating a Drop-down List

The list that was created with the named reference is used here.

1. Select **Data > Data Validation > Data Validation** in the menu bar.
2. Select **List** in the **Allow** drop-down box.



3. Click the icon .

4. Select the column whose contents should be included in the drop-down list.
5. Click **OK**.

⇨ The drop-down list has been created.

15.2 Freezing Titles

◈ Select **View > Freeze Panes > Freeze Top Row** in the menu bar.

15.3 Aligning Text

The text in a cell is automatically aligned to the bottom of the cell if other columns are multiple-line columns. To align the text at the top of a cell, proceed as follows:

1. Select the column and click the right mouse button.
2. Select **Format Cells** in the context menu.
3. Select the **Orientation** tab.
4. Select **Top** in the **Vertical** drop-down list.

15.4 Excel Shortcuts

Description	Shortcut
Copy the selected cells to the clipboard	**<Strg>** + **<Einfg>**
Paste, overwriting existing cells	**<SHIFT>** + **<Einfg>**
Paste new rows with content from the clipboard Important: The clipboard is then empty.	**<Strg>** + **<+>**
Delete selected rows	**<Strg>** + **<->**
Line break in cell	**<ALT>** + **<Enter>**
Select row	**<SHIFT>** + **<Leerzeichen>**
Select column	**<Strg>** + **<Leerzeichen>**
Convert table	Right mouse button **Table →Convert to range**

16 Glossary

PLC
PLC is the Programmable Logic Controller.

Issued by
Siemens Switzerland Ltd
Smart Infrastructure
International Headquarters
Theilerstrasse 1a
CH-6300 Zug, Switzerland

www.siemens.com/buildingtechnologies