IEC61850 Server

straton User Guide – Rev. 18

sales@straton-plc.com

CP_R31_006_H

straton

Content

1. Setup

1.1. Specific to Windows

Before using the IEC61850 Server and GOOSE, additional components need to be installed if the application is aimed to run on a Windows Runtime.

The Ethernet Multiple Network Protocol Driver is available for 32 bits and 64 bits operating systems. It should be installed automatically with the Editor and/or the Runtime.

You can check it in your Windows Control Panel > Install/Uninstall a program, under the name "COPA-DATA Multiple Network Protocol Driver". If not installed, please contact your local support.

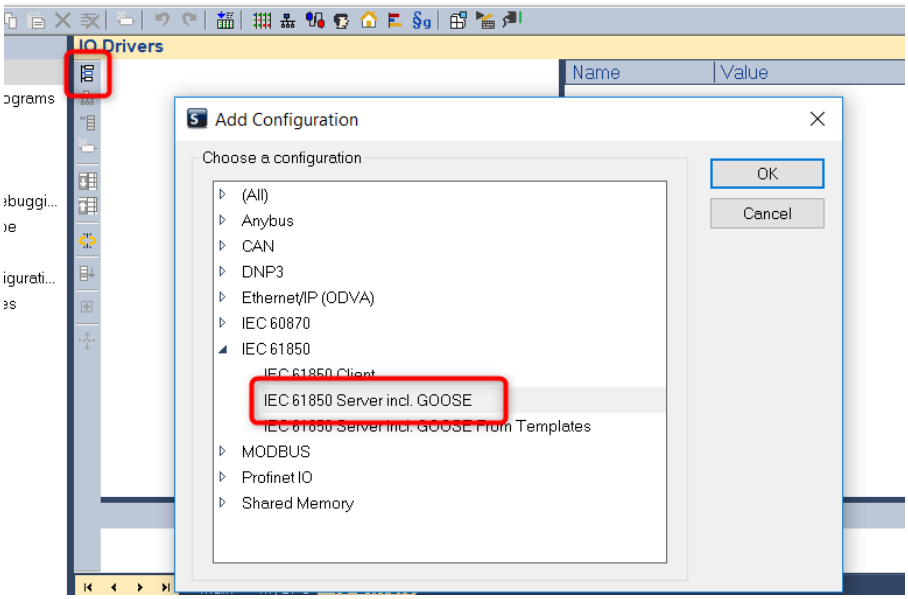
1.2. Install straton IDE and Runtime

Download and install from https://straton-plc.com/telechargements/

2. Create the IEC 61850 Server application

2.1. New straton project

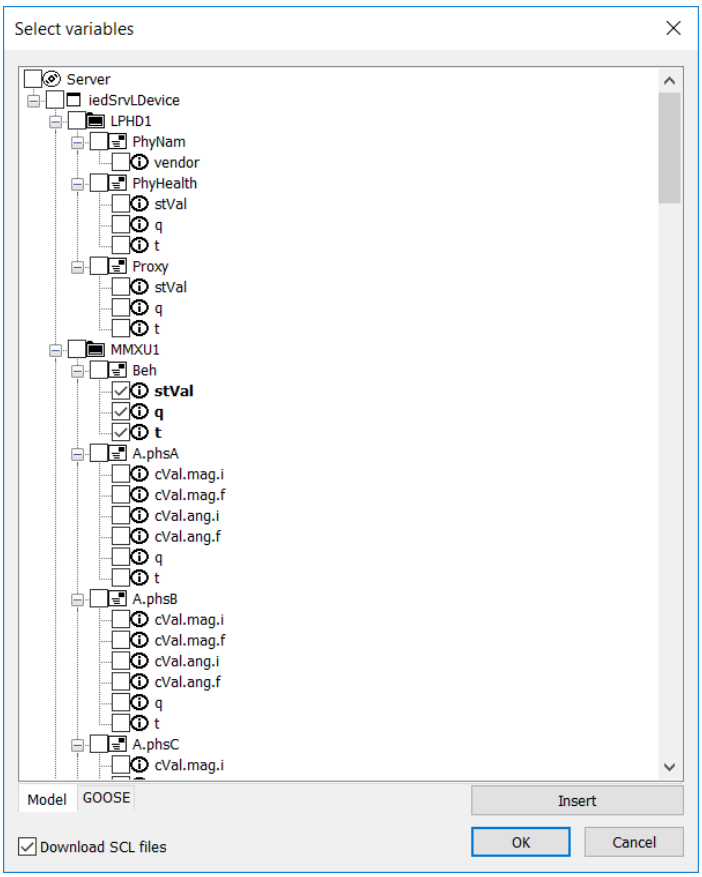
Open the IO Drivers window ( ) and insert the IEC 61850 Server driver ( )



2.2. Import variables in straton

Right-click on the Server configuration and select "*Read variables from SCL file*".

Use the *Insert* button to load an SCL file.



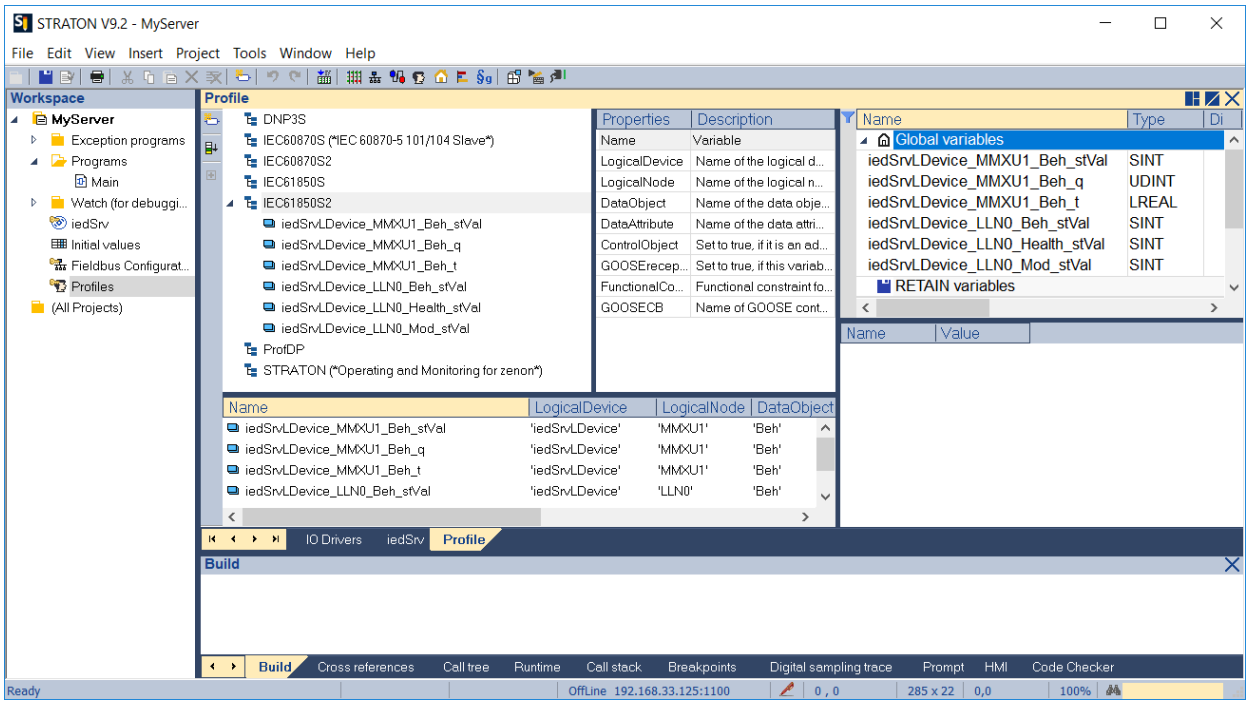
2.3. Select the variables to declare in straton

Simply check the variables you want to declare in the application. Then go to the Profiles view ( ) to see the IEC 61850 variables. These variables are declared as Global Variables in the dictionary and can be accessed directly from anywhere in the straton application (Fieldbus, program, graphics).

IMPORTANT:

If the IED/SCL file is modified, think to **re-insert** the latest updated SCL file clicking on "Insert", to overwrite the old one.

Please, see the Annex A of this document if the latest version of the SCL files must always be automatically downloaded on the Runtime at the same time as the application. But be careful, doing so means that if a small change is done in the SCL (eg. for testing), this new file will automatically be downloaded.



IMPORTANT:

Since straton 9.2 (zenon 8.00), IEC61850S2 Profiles must be used if the Runtime version is higher or equal to 9.2.0 (driver version higher or equal to 8.0.0.43522). Data with the bad Profile will be ignored by the driver.

HINT

To export/import variables from a 61850 Profile to another, right-click on a Profile and "Export Children". Save the *.CSV file. In the straton Profiles, remove all profiled variables (in the Profiles, not in the dictionary). Then, right-click on the desired Profile and "Import Children".

2.4. t-attribute

The *.t Data Attribute contains the date and timestamp of the IEC61850 variable in Unix format.

Updating the ***.t** DA of a Data Object is automatically managed by the stack. For example, if *.stVal changes, then *.t is updated by the stack with the system clock. The precision depends on the OEM implementation.

Be careful: if the *.t DA of a DO is declared in the dictionary (as a variable), the driver does not update the t-value anymore. Since then, it must be fully managed by the application, assigning the value to the *.t variable manually (LREAL in UNIX format).

2.5. q-attribute

Updating the ***.q** Data Attribute of a Data Object must be done in the application. The stack never updates the *.q attribute automatically.

It matches to the Table 2 of the iec61850-7-3_ed2.0 standard.

Eg. STRATONLDevice1_XCBR1_Pos_q = 16#0020 = Good + overflow

Eg. STRATONLDevice1_XCBR1_Pos_q = 16#0041 = Invalid + old data

Value in straton	Meaning
16#0000	**Good**
16#0040	**Invalid**
16#0080	**Reserved**
16#00C0	**Questionable**
16#0020	Overflow
16#0010	Out of Range
16#0008	Bad Reference
16#0004	Oscillatory
16#0002	Failure
16#0001	Old Data
16#8000	Inconsistent
16#4000	Inaccurate
16#2000	Substituted
16#1000	Test
16#0800	Blocked by Operator

2.6. TimeQuality

The Time Quality as defined in the part 6.1.2.9.3.3 of the iec61850-7-2_ed2.0 is managed by the mapping of an additional variable. It is updated for the whole Logical Device and automatically embedded in all *.t DA; meaning that if it is updated in the application, then the *.t must also be updated (eg. on data change) so the time quality can be sent over MMS (to a Client) and/or GOOSE (to another Server).

Add a SINT variable with an IEC61850 Profile and the following attributes:

- Logical Device = Name of your logical device
- Logical Node = *Empty*
- Data object = *Empty*
- Attribute = "TimeQuality"
- Control = Checked
- FC = *Empty*
- Goose = *Empty*

Then the default quality (**`2#11111000`**) can be changed forcing bit(s) of this variable:

Bit	Value	IEC61850-8-1	Description
0		Leap Second Known	Switching second
1		Clock Failure	Clock Failure
2		Clock not synchronized	Clock not synchronized
3-7		Time accuracy of fractions of second	Time accuracy of fractions of second
	00000	0 bit of accuracy	0 bit of accuracy
	00001	1 bit of accuracy	1 bit of accuracy
	00010	2 bits of accuracy	2 bits of accuracy
	00011	3 bits of accuracy	3 bits of accuracy
	00100-11000	Integer value of number of bits of accuracy	Amount of Bits of accuracy, e.g. 01010b means 10 Bits accuracy, equals 1 to 999 (1111100111b) milliseconds behind the coma
	11000-11110	Invalid	Invalid
	11111	unspecified	unspecified

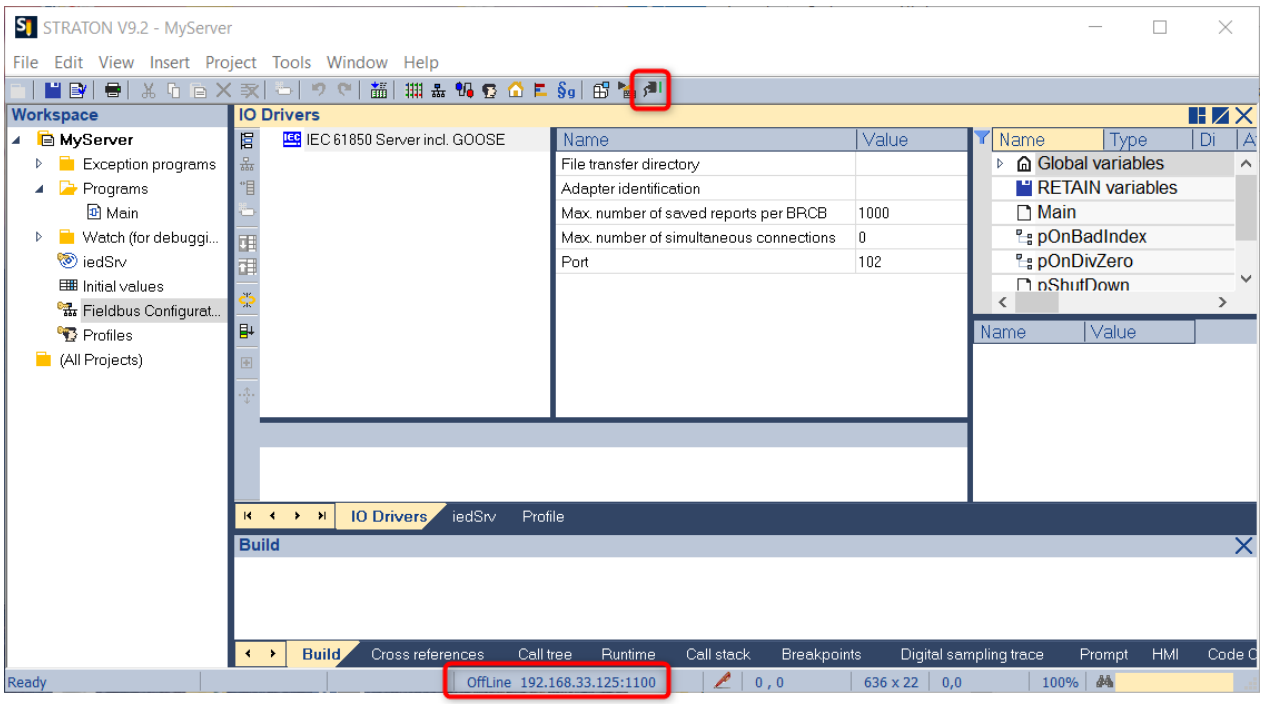
2.7. Download the SCL file

The IEC 61850 Server needs the configuration file of the IED, this file needs to be downloaded to the straton Runtime. Think to check the "Download SCL file" checkbox in the "Select variable" window (See "Import variables in straton").

2.8. Download the straton application

Select the communication parameters in menu Tools > Communication Parameters

Establish the connection through menu Project > Online (or use the Online button)



IMPORTANT:

On the target, SCL files are downloaded in the "Custom" folder (same directory as the runtime).

The default port for communication between the Editor and the Runtime is 502 (Windows) or 1100 (other OS like Linux)

On Linux, think to create the Custom folder "`sudo mkdir Custom`" and start the runtime with root privileges with:

```
# sudo ./t5linux /path850=Custom/
```

Result is:

The download is successful and application starts correctly.

The straton runtime is not started or communication parameters are wrong.

The application is not yet downloaded or an error occurs during startup. More details can be found in the Output view.

3. Create GOOSE communication

3.1. Goose configuration in the SCL

GOOSE Communication is **exclusively** between two Servers: a Publisher and a Subscriber.

To create a GOOSE communication you need to first configure the IEC 61850 Server with the appropriate GOOSE settings in the SCL file. Typically, GOOSE Report Control Blocks (GRCB) linked to Data Sets (DS).

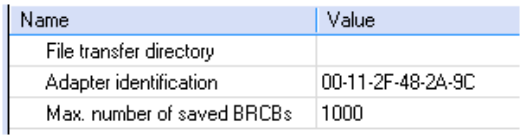
The GOOSE RCB contains the multicast address. It is standard and **must** have the following structure:

- The first three octets are assigned by IEEE with 01-0C-CD.
- The fourth octet shall be 01 for GOOSE, 02 for GSSE, and 04 for multicast sampled values.
- The last two octets shall be used as individual addresses assigned by the range defined in the following table.

Service	Starting Address	Ending Address
GOOSE	01-0C-CD-01-00-00	01-0C-CD-01-01-FF
GSSE	01-0C-CD-02-00-00	01-0C-CD-02-01-FF
Multicast sampled values	01-0C-CD-04-00-00	01-0C-CD-04-01-FF

The configuration of the Server is the same as described previously.

Additionally in the "IEC61850 Server incl. GOOSE" configuration you must fill the MAC address of the Ethernet controller to use for the GOOSE communication:



With **Linux**, the adapter identification must be the name of the interface (for example: eth0).

WARNING

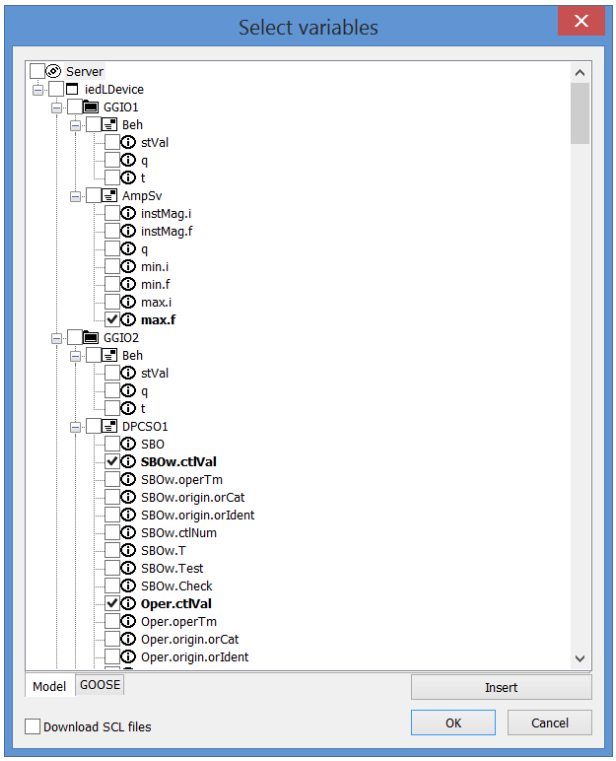
The "MAC Address" present in the GOOSE Report Control block, in the SCL Editor, corresponds to the multicast address (see the table above)

The "Adapter identification" in the Fieldbus configuration is the MAC address (or interface name on other OS than Windows) of the Ethernet card publishing or receiving the GOOSE messages; both are different.

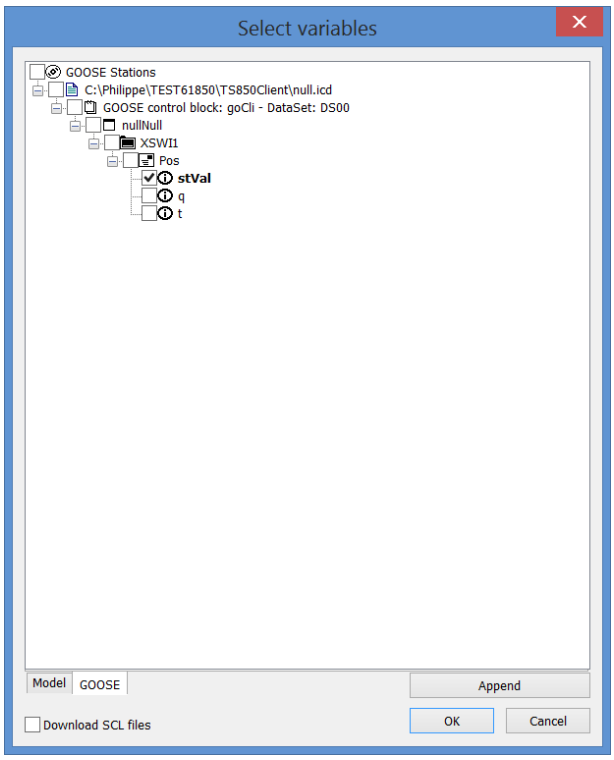
3.2. GOOSE Publisher and Subscriber

The Server publishing GOOSE messages is configured with an SCL file as a classical Server. This means that the SCL for the GOOSE **Publisher** must be inserted the "**Model**" tab after choosing "Read variables from SCL file".

For GOOSE **Subscription**, the SCL file of the **distant** Server must be inserted in the "**GOOSE**" tab.



Server configuration (GOOSE Publisher)



GOOSE Subscription configuration

After inserting the SCL of the Server in the "**Model**" tab, the driver will automatically recognize if GOOSE is configured. If this is the case the Server will automatically act as a GOOSE **Publisher**.

After inserting the SCL of the **distant** Server in the "**GOOSE**" tab, to subscribe to GOOSE messages, the Server will recognize it has to act as a GOOSE **Subscriber** too. If no variable appears, it means that the SCL file does not contain any GOOSE configuration.

If the Server must subscribe to several GOOSE messages coming from several GOOSE publishers, then click on "Append" again to insert as many SCL files you need in the "GOOSE" tab.

Note: With some distribution of Linux, GOOSE may encounter difficulties to emit packets directly if no connection request has been made first. To solve this issue, type in a console, with root rights, the following command line:

```
# ifconfig eth0 promisc
```

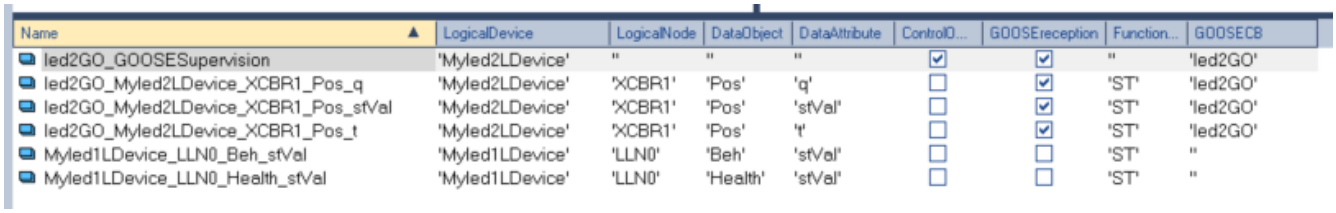
(Replace eth0 by your ethernet interface's name.)

3.3. GOOSE supervision

If two Servers communicate via GOOSE messages, it is possible (since version 9.2/7.20) to supervise the GOOSE communication via a specific variable. It must be a DINT and declared with an IEC61850S2 Profile.

It is linked to a GOOSE Control Block of the Logical Device; thus, its Profile properties must be:

- LogicalDevice = same as the GOOSE Reception variables
- ControlObject = checked
- GOOSEReception = checked
- GOOSECB = same as the GOOSE Reception variables
- The rest can be left empty



Name	LogicalDevice	LogicalNode	DataObject	DataAttribute	ControlO...	GOOSEreception	Function...	GOOSECB
led2GO_GOOSESupervision	'Myled2LDevice'	''	''	''	☑	☑	''	'led2GO'
led2GO_Myled2LDevice_XCBR1_Pos_q	'Myled2LDevice'	'XCBR1'	'Pos'	'q'	☐	☑	'ST'	'led2GO'
led2GO_Myled2LDevice_XCBR1_Pos_stVal	'Myled2LDevice'	'XCBR1'	'Pos'	'stVal'	☐	☑	'ST'	'led2GO'
led2GO_Myled2LDevice_XCBR1_Pos_t	'Myled2LDevice'	'XCBR1'	'Pos'	't'	☐	☑	'ST'	'led2GO'
Myled1LDevice_LLN0_Beh_stVal	'Myled1LDevice'	'LLN0'	'Beh'	'stVal'	☐	☐	'ST'	''
Myled1LDevice_LLN0_Health_stVal	'Myled1LDevice'	'LLN0'	'Health'	'stVal'	☐	☐	'ST'	''

For the moment only the three less significant bits are used.

Bit #	Description
0	Set to TRUE in case of connection loss. The connection loss is detected if the "TimeAllowedToLive" contained in the GOOSE message is elapsed
1	Set to TRUE if the GOOSE message has a wrong sequence number
2	Set to TRUE if a GOOSE Publisher server is or was detected

Example: Server_1 is a GOOSE subscriber, Server_2 is a GOOSE publisher:

Server_1 has never received GOOSE messages from Server_2 --> Variable = 0

Server_1 receives GOOSE messages from Server_2 --> Variable = 4

Server_1 does not receive GOOSE messages anymore --> Variable = 5

Note that only if the GOOSE Supervision variable is declared, the following message appears if a GOOSE message is received after its time allowed to live:

> *TimeAllowedToLive for GOOSE control block IEDPUBLDevice/LLN0.GCB1 expired. Last report received 4001 ms ago, TimeAllowedToLive = 4000*

4. Configuring the keepalive of the operating system

The 61850 protocol relies on the TCP/IP keep-alive packets to check if the connection is still alive and detect when a peer is disconnected.

It is used for example to detect if a RCB subscription must be disabled or not in case of connection loss since too long. On some Linux platforms this delay is 2h by default.

With Linux, configure these parameters typing in a console, with administrator rights:

```
echo 5 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 5 > /proc/sys/net/ipv4/tcp_keepalive_probes
echo 1 > /proc/sys/net/ipv4/tcp_keepalive_intvl
echo 3 > /proc/sys/net/ipv4/tcp_retries2
```

This should reduce the time the system takes to close a connection after a hard disconnection (typically the RJ45 socket unplugged) to about 20 seconds; because the disconnection is detected after 5s, then 5 probes are sent every 1s, three times (5s + 5*1s*3 = 20s)

With VxWorks, the configuration depends on the VxWorks version.

See http://www.windriver.com/

With Windows, see these parameters:

http://technet.microsoft.com/en-us/library/cc957549.aspx

http://technet.microsoft.com/en-us/library/cc957548.aspx

http://technet.microsoft.com/en-us/library/cc938210.aspx

http://technet.microsoft.com/en-us/library/cc938209.aspx

With WinCE 5.x or 6.x machine the 'keep-alive' can be modified in the *Registry Editor*:

- HKEY_LOCAL_MACHINE\Comm\Tcpip\Parms

If the *KeepAliveTime* variable is not yet created, go to Value -> Add DWORD and set, for example, 5000 in decimal for 5s.

5. MMS error codes

When the Server is in *RUN* mode, if the Client sends bad requests the following error codes can occur:

Error code	Description
0	Object - Invalidated
1	Hardware fault
2	Temporarily unavailable
3	Object - Access denied
4	Object - Undefined
5	Invalid address
6	Type - Unsupported
7	Type - Inconsistent
8	Object attribute - Inconsistent
9	Object access - Unsupported
10	Object - Non-existent

6. GOOSE simulation mode

6.1. Prerequisites

Since version 11.0, the GOOSE simulation mode is available; both on the Publisher and Adapter side.

Refer to the part 7.8.2 of the iec61850-7-1 Ed2.0 standard.

To activate it, edit the LPHD LN in the SCL Editor and activate the "Sim" DA.

When reading variables from the SCL file, check the LPHD1_Sim_stVal DA so it is declared in the project's dictionary.

6.2. GOOSE Publisher

As a GOOSE Publisher – once the application is in RUN mode – simply set the *xxx_LPHD1_Sim_stVal_ST* value to TRUE. Since then, GOOSE messages will be published with the simulation bit set.

6.3. GOOSE Subscriber

As a GOOSE Subscriber, once the application is in RUN mode, if *xxx_LPHD1_Sim_stVal_ST*

- is FALSE, then only GOOSE messages with the simulation bit **not** set will be taken in account by the Server. Others will be discarded, and the Runtime will output, if relevant:

"Ignoring received simulation GOOSE PDU, APPID=0x0000, GoCBRef="IEDPUBLDevice/LLN0GOGCB1", DataSet="IEDPUBLDevice/LLN0$DSGOPUB1"

- is TRUE, then if two GOOSE messages are incoming with the **same** GOOSE Control Block reference, only the one with the simulation bit set will be taken in account by the Server. Others will be discarded, and the Runtime will output, if relevant:

"Ignoring received GOOSE PDU, Device is in simulation mode, APPID=0x0000, GoCBRef="IEDPUBLDevice/LLN0GOGCB1", DataSet="IEDPUBLDevice/LLN0$DSGOPUB1"

Additionally, if the LGOS LN is declared on the GOOSE Subscriber side, and if LGOSx.GoCBRef.setSrcCB is equal to the GOCB reference of received messages, then the following DAs are set automatically:

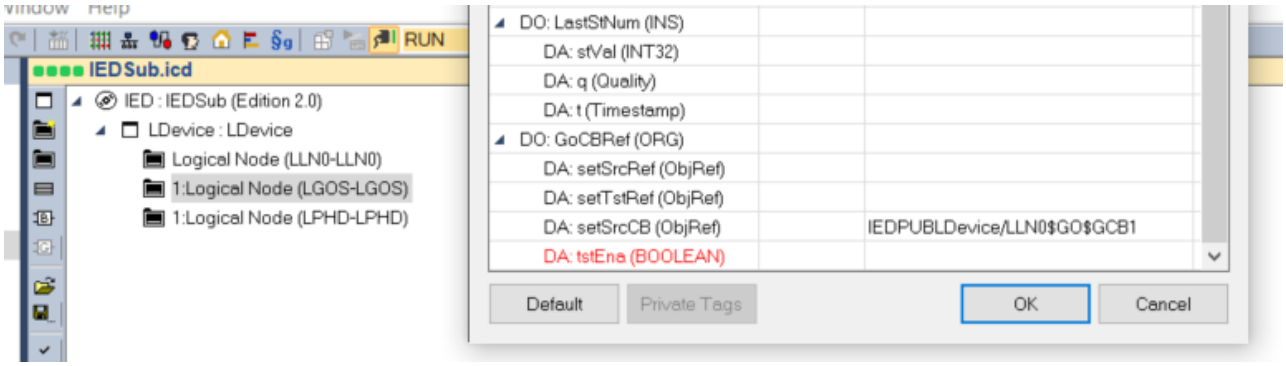
- LGOSx.St
- LGOSx.SimSt
- LGOSx.NdsCom
- LGOSx.LastStNum

(Think to Edit the LPHD type in the SCL editor and add the previous DAs + BDAs)

For example, if in the GOOSE messages contain: **gocbRef: IEDPUBLDevice/LLN0GOGCB1**

Then, set IEDPUBLDevice/LLN0GOGCB1 as an initial value of LGOS.GoCBRef.setSrcCB

In the straton SCL Editor, right-click on the LGOS LN > Edit DOI/DAI



7. Frequently Asked Questions

How to create an SCL file?

The SCL file can be created using an SCL editor according to the capabilities required in the device. straton includes an SCL Editor. Right-click in the project tree > Insert New Item > IEC 61850 – SCL Editor.

The Runtime outputs "Error: Could not parse SCL file"

Either the file is not existing on the target, so the Runtime cannot read it, either it has not been downloaded by the Editor. In both cases, ensure that the "Custom" folder exists on the target and that the Runtime is launched with the option "/path850=" indicating where this folder is.

Typically:

```
# sudo ./t5runtime /path850=Custom/
```

ATTENTION, on Linux this is case sensitive.

The Editor indicates: "Runtime system cannot open file for write"

See the question above. The Custom folder probably does not exist on the Runtime.

The Editor indicates: "Runtime system cannot open local file for read"

It means that the SCL file(s) is/are not existing in the local project. Think to insert it using "Read variables from SCL file". After the insertion, these must be present in the project's Custom folder (on your PC).

Runtime indicates: "Error: Data attribute ..... does not exist in IEC 61850 object model"

Please check in the variable Profile list; it means that some variable(s) is/are declared but not present in the SCL file. It can happened for example if it has not been deleted from a previous import, or if the Profile properties have been altered or not filled correctly (eg. GOOSE Supervision variable with an invalid profile).

The IEC61850-based file transfer fails

The IEC 61850 "File transfer directory" property needs to be filled in the fieldbus configuration. If not, the file transfer is not activated.

Connections are not closing correctly when the Client disconnects

Refer to the chapter about keepalives of this document.

I encounter difficulties to emit GOOSE packets with Linux

Type in a console, with administrator rights, the following command line:

```
# ifconfig eth0 promisc
```

Replace **eth0** by your interface's name. This will enable your network interface in promiscuous mode and remove filtering of packets.

I cannot re-use an RCB I already enabled, and then disconnect

This may happen if the first connection has not been closed correctly. Refer to the chapter about keepalives of this document.

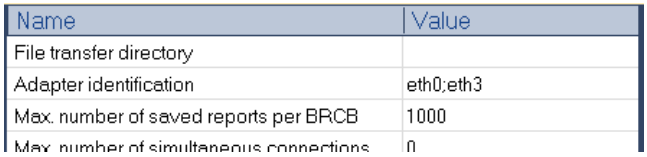
*I can create an array in the SCL file, but then the project does not start and displays the error message "Data attribute *** in IEC 61850 object model has wrong type", even if the type is correct*

You need to change the project settings (in menu Project > Settings) and set the options "Complex variables in a separate segment" to YES

Runtime indicates: "Could not enable GOOSE control block ..." or "Could not start GOOSE..."

Think to set a correct "Adapter Identification" in the Server configuration. On Windows, use the MAC address of the Ethernet card. On Linux, use its name (eg. eth0).

If several adapters must be used, use the ";" as separator. Eg. **eth0;eth3**



Name	Value
File transfer directory	
Adapter identification	eth0;eth3
Max. number of saved reports per BRCB	1000
Max. number of simultaneous connections	0

Runtime indicates: "Error opening Ethernet socket on adapter xxx"

Think to start the Runtime with admisistrator rights. If this is already done, be sure that the adapter exists and that the "Adapter identification" has been filled correctly (see question above).

Annex A – How to download the latest updated SCL files automatically?

After modifying an SCL file, the user needs to right-click on the configuration > Read variables from SCL file and re-select the latest updated SCL file. This way, it is possible to modify the SCL file independently of the application.

To override this security, the user can manually edit the download procedure and choose which files will be downloaded at the same time as the application. To do so, go to

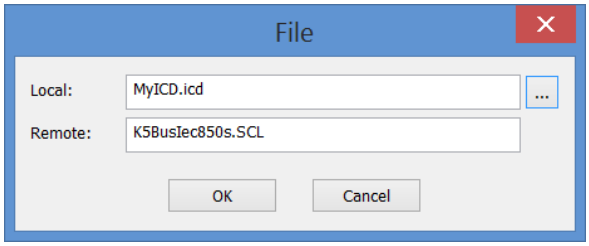
Project > Settings > Debugging > Download procedure > Edit

"SCL: IEC850 Server+GOOSE > \Custom" is automatically created when checking "Download SCL file" option (see "Import variables in straton").

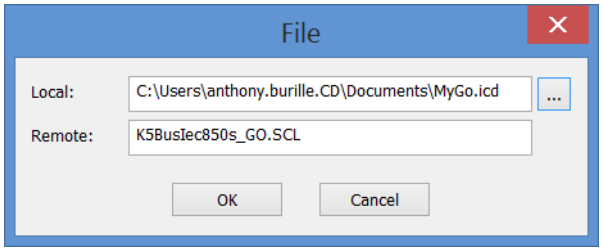
Remove this line.

Add the Server SCL file to be downloaded by clicking on "Add".

Choose the SCL file and choose the name it must have on the runtime side: K5Buslec850s.SCL



Do the same thing for the GOOSE SCL file. On the Runtime side the name must be K5Buslec850s_GO.SCL

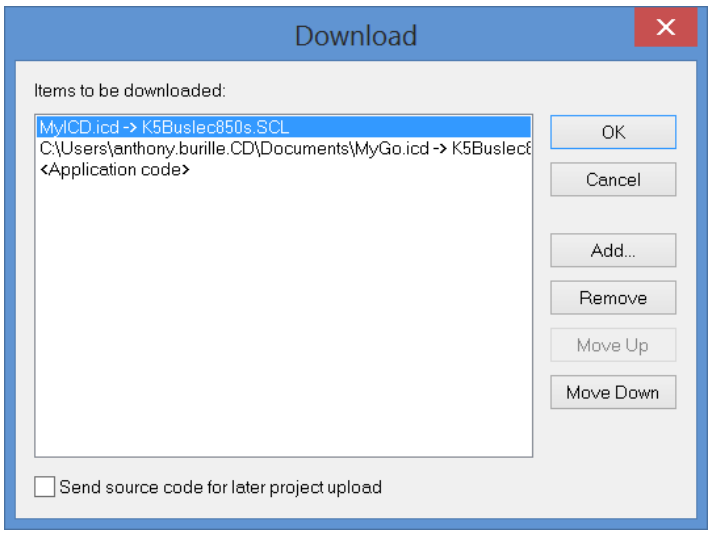


Be careful, the names are case sensitive!

IEC850 model file is "`K5BusIec850s.SCL`" on the target.

IEC850 GOOSE file is "`K5BusIec850s_GO.SCL`" on the target.

These two items must be downloaded before the application (<Application code>!)



If the popup window "Cannot open local file for read" appears it means that one of the documents in the download procedure does not exist.