BACSL

Contents

BACSL Driver

BACSL Driver for Serial Communication with Slave Devices Using the BACnet/IP or BACnet MS/TP Protocol (version 3.3, last revised 31 May 2015).

The BACSL driver enables communication between the Studio system and remote devices using the BACnet protocol, according to the specifications discussed in this document.

This document assumes that you have read the "Development Environment" section in the main Studio documentation.

This document also assumes that you are familiar with the Microsoft Windows XP/Vista/7 environment. If you are not familiar with Windows, then we suggest using the **Help and Support** feature (available from the Windows **Start** menu) as you work through this document.

Driver specifications

This section identifies all of the software and hardware components required to implement communication between the BACSL driver in Studio and remote devices using the BACnet protocol.

Driver files

The BACSL driver package comprises the following files, which are automatically installed in the `Drv` folder of the Studio application directory:

- `BACSL.DLL`: Compiled driver.
- `BACSL.INI`: Internal driver file. *You must not modify this file.*
- `BACSL.MSG`: Internal driver file defining error messages for the possible error codes. (These error codes are described in detail in the Troubleshooting section.) *You must not modify this file.*
- `BACSL.PDF`: This document, which provides complete information about using the driver.

> **Note:** You must use a compatible PDF reader to view the `BACSL.PDF` file. You can install Acrobat Reader from the Studio installation CD, or you can download it from Adobe's website.

You can use the BACSL driver on the following operating systems:

- Windows XP/Vista/7/8
- Windows Server 2003/2008
- Windows Embedded Compact 5.x/6.x

Device specifications

To establish communication, your target device must meet the following specifications:

- Manufacturer:
- Compatible Equipment: Any device that is compatible with the BACnet protocol
- Programmer Software: It depends on the device

The BACSL driver supports the following device registers:

Supported registers

Property	Type	AI	AO	AV	BI	BO	BV	DEV
object-name	Character String	X	X	X	X	X	X	X
object-type	BACnet Object Type	X	X	X	X	X	X	X
present-value	REAL	X	X	X	X	X	X	-
description	Character String	X	X	X	X	X	X	X
device-type	Character String	X	X	-	X	X	-	-
status-flags	BACnet Status Flags	X	X	X	X	X	X	-
event-state	BACnet Event State	X	X	X	X	X	X	-
reliability	BACnet Reliability	X	X	X	X	X	X	-
units	BACnet Engineering Units	X	X	X	-	-	-	-
min-pres-value	REAL	X	X	-	-	-	-	-
max-pres-value	REAL	X	X	-	-	-	-	-
resolution	REAL	X	X	-	-	-	-	-
cov-increment	REAL	X	X	X	-	-	-	-
priority-array	BACnet Priority Array	-	X	X	-	X	X	-
time-delay	Unsigned	X	X	X	X	X	X	-
notification-class	Unsigned	X	X	X	X	X	X	-
high-limit	REAL	X	X	X	-	-	-	-
low-limit	REAL	X	X	X	-	-	-	-
deadband	REAL	X	X	X	-	-	-	-
limit-enable	BACnet Limit Enable	X	X	X	-	-	-	-
event-enable	BACnet Event Transition Bits	X	X	X	X	X	X	-
acked-transitions	BACnet Event Transition Bits	X	X	X	X	X	X	-
notify-type	BACnet Notify Type	X	X	X	X	X	X	-
profile-name	Character String	X	X	X	X	X	X	-
polarity	BACnet Polarity	-	-	-	X	X	-	-
inactive-text	Character String	-	-	-	X	X	X	-
active-text	Character String	-	-	-	X	X	X	-
change-of-state-time	BACnet Date Time	-	-	-	X	X	X	-
change-of-state-count	Unsigned	-	-	-	X	X	X	-

Time-of-state-countreset	BACnet Date Time	-	-	-	X	X	X	-
elapsed-active-time	Unsigned32	-	-	-	X	X	X	-
Time-of-active-timereset	BACnet Date Time	-	-	-	X	X	X	-
alarm-value	BACnet Binary PV	-	-	-	X	-	X	-
minimum-off-time	Unsigned32	-	-	-	-	X	X	-
minimum-on-time	Unsigned32	-	-	-	-	X	X	-
feedback-value	BACnet Binary PV	-	-	-	-	X	-	-
number-of-states	Unsigned	-	-	-	-	-	-	-
event-time-stamps	Character String	X	X	X	X	X	X	-
System-status	Enumerated	-	-	-	-	-	-	X
Vendor-Name	Character String	-	-	-	-	-	-	X
Vendor-Identifier	Unsigned	-	-	-	-	-	-	X
Model-Name	Character String	-	-	-	-	-	-	X
Firmware-Revision	Character String	-	-	-	-	-	-	X
Application-SoftwareVersion	Character String	-	-	-	-	-	-	X
Location	Character String	-	-	-	-	-	-	X
Protocol Version	Unsigned	-	-	-	-	-	-	X
Protocol Revision	Unsigned	-	-	-	-	-	-	X
Protocol-ServicesSupported	Binary String	-	-	-	-	-	-	X
Protocol-Object-Types-Supported	Binary String	-	-	-	-	-	-	X
Database-Revision	Unsigned	-	-	-	-	-	-	X

Network specifications

To establish communication, your device network must meet the following specifications:

- Device Communication Port: 47808 for BACnet/IP
- Physical Protocol: (MS/TP) RS232/RS845 and BACnet/IP
- Logic Protocol: BACnet
- Device Runtime Software: None
- Specific PC Board: None
- Cable Wiring Scheme: It depends on the device

Conformance testing

The following hardware/software was used for conformance testing:

- Driver Version: 3.3
- Studio Version: 7.1+SP3
- Operating System (development): Windows 7/8
- Operating System (target): Windows XP/7/8, Windows Embedded Compact 6.0
- Equipment: BACnet Slave Devices
- Communication Settings:
- Port: 1
- Baud Rate: 9600
- Protocol: RTU
- Data Bits: 8
- Stop Bits: 1
- Parity: Odd
- COM Port: COM1
- Cable: Use specifications described in the "Network Specifications" section above.

Configuring the device's communication settings

This section explains how to configure the communication settings for the remote device.

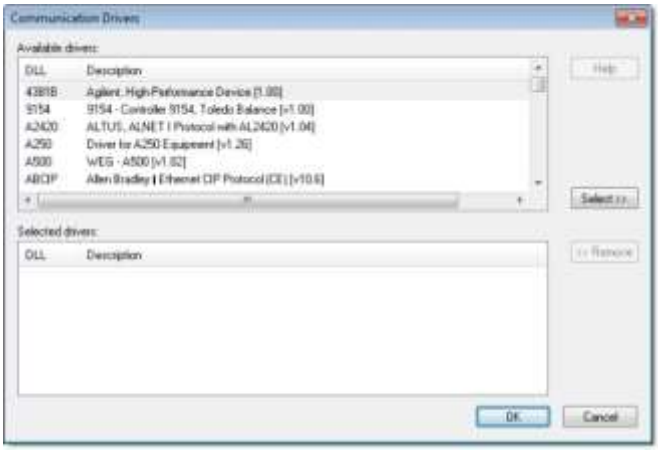
Because there are several brands of equipment that use the BACnet protocol, it is impossible to define a standard device configuration. Therefore, we suggest using the following default configuration:

- **Protocol**: BACnet/IP, BACnet MS/TP
- **Baud Rate**: Depends on the device
- **Data Bits**: Depends on the device
- **Stop Bits**: Depends on the device
- **Parity**: Depends on the device

Adding a communication driver to your project

This section explains how to add a communication driver to your project.

1. On the **Insert** tab of the ribbon, in the **Communication** group, click **Add/Remove Driver**.
   The *Communication Drivers* dialog is displayed.



Communication Drivers dialog

2. In the *Available drivers* list, click the communication driver that you want to add.
3. Click **Select**.
   The driver is added to the *Selected drivers* list.
4. Click **OK**.
   The *Communication Drivers* dialog is closed and the selected driver is inserted in the **Drivers** folder in the Project Explorer.

Configuring the driver's communication settings

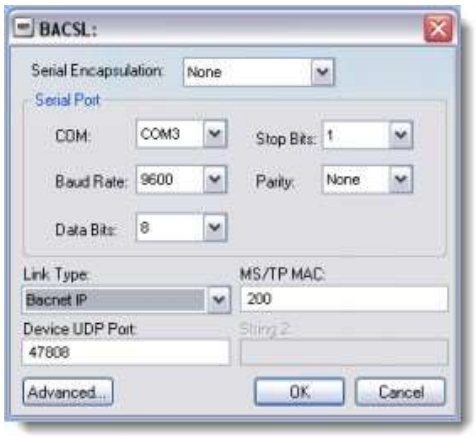
This section explains how to configure the communication settings for the driver.

You must add the communication driver to your project before you can configure its settings. For more information, see Adding a communication driver to your project on page 8.

The general procedure for configuring a driver's communication settings is the same for all drivers. However, the specific settings are different for each driver, depending on the options and protocols used by the target device.

To configure the communication settings:

1. In the **Comm** tab of the Project Explorer, expand the **Drivers** folder.
   The folder contains the drivers that are currently enabled. If you do not see the driver that you want to configure, then you need to add it.
2. Right-click the driver that you want to configure, and then click **Settings** on the shortcut menu.
   The *Communication Settings* dialog is displayed.



Communication Settings: BACSL dialog

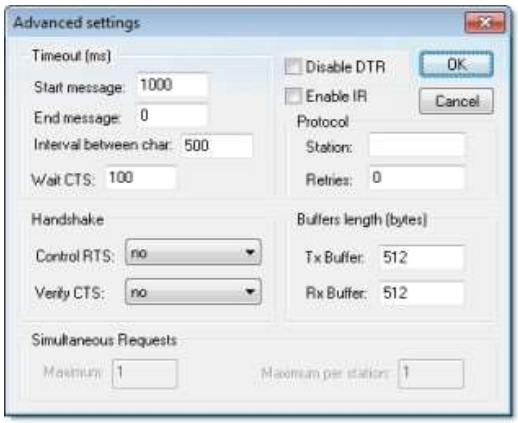
3. Configure the remaining, driver-specific settings as needed.

Driver-specific communication settings

Setting	Default Value	Valid Values	Description
Link Type	`Bacnet IP`	`Bacnet IP` or `MSTP`	Select the type of communication link to use: BACnet IP (UDP) or MS/TP interface.
MS/TP MAC	`0`	`0` to `254`	MAC address of the MS/TP network.
Device UDP Port	`47808`	`1` to `65535`	Specify the UDP port. If there is nothing configured, then the driver will use the default value of 47808(BAC0h). It will be used only when **Link Type** is set to `Bacnet IP`.

4. Click **Advanced**.

   The *Advanced Settings* dialog is displayed.



Advanced Settings dialog

5. Configure the advanced communication settings as needed. The following advanced settings are required:

Advanced communication settings

Setting	Default Value	Valid Values	Description
Station	0	*<Network>:<Device ID>*	You must specify a network number a device ID for the slave device. For example, `1234:12`. This field cannot be left blank.

6. In particular, if you are using a Data Communication Equipment (DCE) converter (e.g., 232/485) between your project and the target device, then you need to adjust the **Control RTS** setting to account for the converter.

Option	Description
no	Do not set the RTS handshake signal.
yes	Set the RTS handshake signal before communication.
Yes + echo	Set the RTS handshake signal before communication, and then echo the signal received from the target device.
Always on	Set the RTS handshake signal and keep it on.

> **Note:** If you incorrectly configure the **Control RTS** setting, then runtime communication will fail and the driver will generate a -15 error.

7. Click **OK**.

   The settings are saved and the *Advanced Settings* dialog is closed.

8. Click **OK**.
   The settings are saved and the *Communication Settings* dialog is closed.

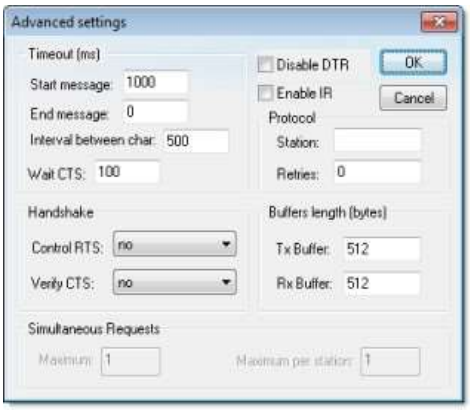
Advanced Settings dialog

The *Advanced Settings* dialog is used to configure advanced communication settings for a specific communication driver.

Accessing the dialog

To access the *Advanced Settings* dialog for a specific driver:

1. In the **Comm** tab of the Project Explorer, expand the **Drivers** folder. The folder contains the drivers that are currently added to the project.
2. Right-click the specific driver, and then click **Settings** on the shortcut menu. The *Communication Settings* dialog is displayed.
3. Click **Advanced**. The *Advanced Settings* dialog is displayed. **The dialog in detail**



Advanced Settings dialog

- *Timeout (ms)* area:
- **Start message** field: Specify the timeout for the message start.
- **End message** field: Specify the timeout for the message end.
- **Interval between char** field: Specify the timeout between each character.
- **Wait CTS** field: Specify the timeout for the Clear To Send wait.
- *Handshake* area:
- **Control RTS** drop-down list: Specify whether to use the Request To Send control.
- **Verify CTS** drop-down list: Specify whether to use the Clear To Send type of verification.
- **Disable DTR** checkbox: Select to disable the DTR function (the driver will not set the DTR signal before starting the communication).
- **Enable IR** checkbox (*only available on Windows Embedded target systems*): Select to enable the serial driver to use an Infrared interface (COM2 port) instead of a standard serial port to communicate with the device (such as the PLC, I/O, hand-held computers, and so forth).
- *Protocol* area:
- **Station** field: Some slave drivers such as the Modbus Slave (MODSL) require a slave network address. Use this field to specify the slave address.
- **Retries** field: Type a numeric value to specify how many times the driver will attempt to execute the same communication command before considering a communication error for this command.
- *Buffers length (bytes)* area:

- **Tx Buffer** field: Specify the transmission buffer length (in bytes).
- **Rx Buffer** field: Specify the reception buffer length (in bytes).
- *Simultaneous Requests* area (available only on selected drivers):
- **Maximum** field: Specify the maximum number of requests that may be sent simultaneously to all connected devices.
- **Maximum per station** field: Specify the maximum number of requests that may be sent simultaneously to a single device.

> **Note:** The maximum number of simultaneous requests depends on the device and protocol specifications. Please consult the device manufacturer's documentation.

About driver worksheets

Like the other parts of your project, communication with remote devices is controlled by worksheets. This section explains how to add worksheets to your project and then configure them to associate project tags with device registers.

Each selected driver includes one or more Standard Driver Sheets (SDS). Standard Driver Sheets can be inserted to define tag/register associations that are triggered by specific project behaviors.

The configuration of these worksheets is described in detail in the "Communication" chapter of the *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

For the purposes of this document, only BACSL driver-specific parameters and procedures are discussed here.

Adding and configuring a Standard Driver Sheet

By default, a communication driver does not include any Standard Driver Sheets. This section explains how to add a Standard Driver Sheet to your project and then configure it.

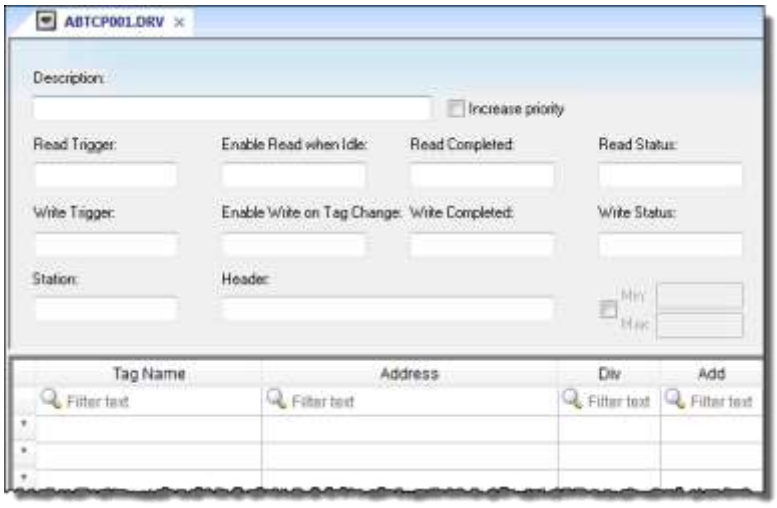
The BACSL driver must be added to the project before you can configure any of its worksheets. For more information, see Adding a communication driver to your project on page 8.

Standard Driver Sheets can be inserted to define additional tag/register associations that are triggered by specific project behaviors.

> **Note:** Most of the settings on this worksheet are standard for all drivers; for more information about configuring these settings, see the “Communication” chapter of the *Technical Reference Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the BACSL driver.

1. Do one of the following.
   - On the **Insert** tab of the ribbon, in the **Communication** group, click **Driver Sheet** and then select **BACSL** from the list.
   - In the **Comm** tab of the Project Explorer, right-click the **BACSL** folder and click **Insert** on the shortcut menu.

   A new BACSL driver worksheet is inserted into the **BACSL** folder, and then it is automatically opened for configuring.



Standard Driver Sheet

> **Note:** Worksheets are numbered in order of creation, so the first worksheet is `BACSL001.drv`.

2. Configure the Station and Header fields as described below.

Station

The **Station** field is not used on this slave driver.

Header

Specify the address of the first register of a block of registers on the target device. The addresses declared in the body of the worksheet are simply offsets of this **Header** address. When Read and Write actions are executed for the entire worksheet (using **Read Trigger** and **Write Trigger**, respectively), it scans the entire block of registers from the first address to the last. The **Header** field uses the following syntax:

<Type>:<Instance Reference>

Where:

<Type>

Register type. Valid values are:

- `AI` - Analog Input
- `AO` - Analog Output
- `AV` - Analog Value
- `BI` - Binary Input
- `BO` - Binary Output
- `BV` - Binary Value
- `DEV` - Device

DEV is the Device Information. For this **DEV** type Header the *Instance Reference* parameter will define the DEVICE ID in the BACnet Network

After you edit the **Header** field, the development application checks the syntax to determine if it is valid. If the syntax is invalid, then the development application automatically inserts a default value of `AI:0`.

You can also specify an indirect tag (e.g. `{MyHeader}`), but the tag that is referenced must follow the same syntax and contain a valid value.

3. For each tag/register association that you want to create, insert a row in the worksheet body and then configure the row's fields as described below.

Tag Name

Type the name of the project tag.

Address

Specify the address of the associated device register.

For all register types other than `ST/STS` (String), use the following syntax:

<Instance Offset>:<Property>

Where:

<Instance Offset>

The value that is added to the *<Instance Reference>* parameter (configured in the **Header** field above) to produce the complete instance number.

<Property>

The specific property of the instance. For a list of valid properties, see the table of supported registers in "Driver specifications".

> **Note:**
> For each object with the same instance it must be on the same worksheet, it cannot be split into different worksheets.
>
> **Example:**
>
> Same objects with same instances but with different properties
>
> AI:0:PRESENT-VALUE
>
> AI:0:DESCRIPTION
>
> (THIS MUST BE ON THE SAME DRIVER SHEET)

> **Note:** Each Standard Driver Sheet can have up to 4096 rows. However, the **Read Trigger**, **Enable Read When Idle**, and **Write Trigger** commands attempt to communicate the entire block of addresses that is configured in the sheet, so if the block of addresses is larger than the maximum block size that is supported by the driver protocol, then you will receive a communication error (e.g., "invalid block size") during run time. Therefore, the maximum block size imposes a practical limit on the number of rows in the sheet.

For examples of how device registers are specified using **Header** and **Address**, see the following table.

Examples of Header and Address fields in Standard Driver Sheet

Address on the Device	Header	Address
AI:0:OBJECT-NAME	`AI:0`	`0:OBJECT-NAME` or `0:77`
AI:2:PRESENT-VALUE	`AI:1`	`1:PRESENT-VALUE` or `1:85`
AI:5:LOW-LIMIT	`AI:3`	`2:LOW-LIMIT` or `2:59`

For more information about the device registers and addressing, please consult the manufacturer's documentation.

4. Save and close the worksheet.

Additional notes

Additional notes about the BACSL driver.

Data types

The BACnet protocol uses several data types to transport values from devices. The primitive data types, such as integers, strings and floating-point numbers are easily used and understood. However, some properties employ enumerations, dates and times to represent its data.

The enumerations are associated with textual values on the properties that use this data type. The BACSL driver returns the numeric value of these enumerations on the tags used to read them. The following enumerations are currently used:

Enumeration	Values
BACnetEventState	normal (0), fault (1), offnormal (2), high-limit (3), low-limit (4), life-safety-alarm (5)
BACnetNotifyType	alarm (0), event (1), ack-notification (2)
BACnetPolarity	normal (0), reverse (1)
BACnetBinaryPV	inactive (0), active (1)
Tag Number	NULL (0), Boolean (1), Unsigned Integer (2), Integer (3), Real (4), String (7), Enumerated (9)

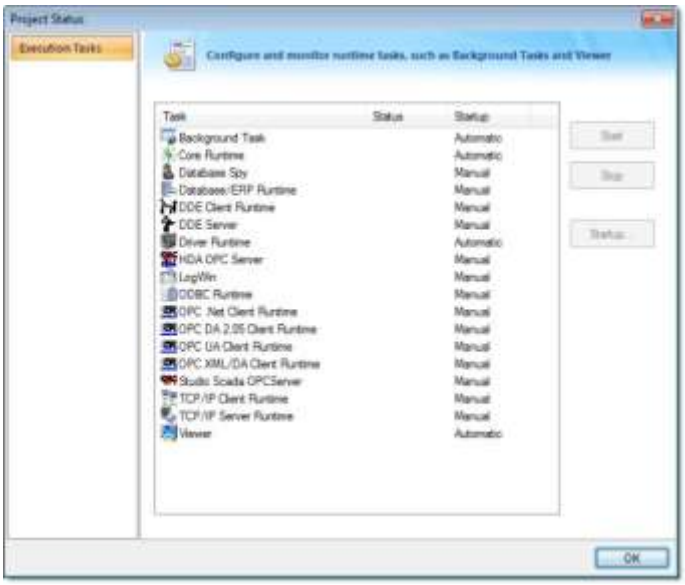
The **BACnetEngineeringUnits** enumeration is also used, on the **UNITS** property. However, due to space limitations, its possible values are not shown here.

Checking the Driver Runtime task

This section describes how to check the status of the Driver Runtime task in the list of execution tasks.

The Driver Runtime task handles communication with remote devices and the processing of the driver worksheets. By default, the task is configured to start up automatically when the project is run, but you can check it for yourself.

1. On the **Home** tab of the ribbon, in either the **Local Management** or the **Remote Management** group (depending on where you project server will be running), click **Tasks**.
   The *Project Status* dialog is displayed.



Project Status dialog

2. Verify that the **Driver Runtime** task is set to **Automatic**.

- If the setting is correct, then proceed to the next step.
- If the **Driver Runtime** task is set to **Manual**, select the task and then click **Startup** to change the task to **Automatic**.

3. Click **OK** to close the *Project Status* dialog.

Troubleshooting

This section lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.

Checking status codes

If the BACSL driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** and **Write Status** fields of the driver sheets will receive a status code. Use this status code and the following tables to identify what kind of failure occurred and how it might be resolved. **Status codes for the driver**

Error	Description	Possible Causes	Procedure To Solve
0	OK	No problem detected	None
1	Invalid operation	Trying to read or write	There is no read or write operation supported on this driver; this driver is slave. For more information, see "About driver worksheets".
3	Malformed packet	An invalid packet with a wrong or corrupted format has been received.	None.
5	Could not create receiver sink	Internal Error	Restart the driver.
7	Unsupported BVLL function	The received function is not supported	None.
9	Segmentation is not supported	The received message is segmented	None.
11	Object not found	The Master requested an Object from the slave that does not exist.	None.
12	Invalid station on driver settings	Invalid information typed	See "Configuring the driver's communication settings".
13	Invalid MAC on driver settings	Invalid information typed	See "Configuring the driver's communication settings".
14	Invalid UDP Port on driver settings	Invalid information typed	See "Configuring the driver's communication settings".

Common status codes

Status Code	Description	Possible Causes	Procedure To Solve
0	OK	Communicating without error.	None required.
-15	Timeout waiting for message to start	• Disconnected cables. • PLC is turned off, in stop mode, or in error mode. • Wrong station number. • Wrong parity (for serial communication). • Wrong RTS/CTS configuration (for serial communication).	• Check cable wiring. • Check the PLC mode — it must be RUN. • Check the station number. • Increase the timeout in the driver's advanced settings. • Check the RTS/CTS configuration (for serial communication).

Monitoring device communications

You can monitor communication status by establishing an event log in Studio's *Output* window (LogWin module). To establish a log for Field Read Commands, Field Write Commands and Serial Communication, right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the LogWin module to establish an event log on a remote unit that runs Windows Embedded. The log is saved on the unit in the `celog.txt` file, which can be downloaded later.

If you are unable to establish communication between Studio and the target device, then try instead to establish communication using the device's own programming software. Quite often, communication is interrupted by a hardware or cable problem or by a device configuration error. If you can successfully communicate using the programming software, then recheck the driver's communication settings in Studio.

Contacting Technical Support

If you must contact Technical Support, please have the following information ready:

- **Operating System** and **Project Information**: To find this information, click **Support** in the **Help** tab of the ribbon.
- **Driver Version** and **Communication Log**: Displays in the *Output* window (LogWin module) when the driver is enabled and the project is running is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Revision history

This section provides a log of all changes made to the driver.

Revision history

Driver Version	Revision Date	Description of Changes	Author
3.0	26 May 2011	First version released	Paulo Balbino
3.1	12 April 2013	• Added support to new Objects and Properties • Added auto response of default values of the Device Object	Paulo Balbino
3.2	20 March 2015	• Added Event-Time-Stamps property support for the relative headers.	Vijay Kankanala
3.3	31 May 2015	• Solved the problem of repetitive object properties which are shown at the BACnet Lookout software.	Paulo Balbino