Connector Implementation Reference

Build custom data connectors.

Reference Code Extensions API Connector Implementation Reference

---

Connector Implementation (Reference): The software platform supports various protocols, such as Modbus, OPC UA, and AB Rockwell ControlLogix, to ensure compatibility with a wide range of industrial devices. In addition to the larger set of protocols provided, it is possible also to create new Connector Drivers with the Driver Toolkit.

When creating new drivers, or doing advanced diagnostics in existing ones, it is important to have a deep understanding on how a communication protocol for industrial devices works.

Key Concepts and Terms
Commonly Used Protocols
- Modbus
- OPC UA (Open Platform Communications Unified Architecture)
- AB Rockwell ControlLogix

Mapping Protocol Addresses
- Read Groups and Write Groups
- Communication Optimization
- Read On Display Access Type

Communication Protocol Execution
- Communication Messages
- Protocol Example: Modbus
- Modbus TCP/IP Message Structure
- Protocol Example: STX and ETX
- Defining Communication Blocks

Key Concepts and Terms

Protocols define the rules and message formats that govern device communication. Each external device follows specific communication rules established by its manufacturer or protocol creator. These rules dictate the structure of TX messages (sent from the platform to the device) and RX messages (sent from the device to the platform). Users select the appropriate protocol for the device they want to connect to and configure the necessary communication drivers to enable communication.

- **Communication protocol**: A set of rules that define how external devices communicate with the software platform.
- **Communication driver**: Software designed for data transmission between the software platform and an external device following the device's communication protocol.
- **TX message**: A message sent from the software platform to the external device.
- **RX message**: A message sent from the external device to the software platform.
- **Byte**: The basic unit of information used in TX and RX messages.
- **External device**: Any device or software used with the software platform, such as PLCs, sensors, and actuators.

Commonly Used Protocols

The platform supports multiple communication protocols, each with specific characteristics and benefits suited to different applications. Here are descriptions of some commonly used protocols:

Modbus

A serial protocol that uses a master-slave architecture to communicate with external devices. It operates on a request-response basis, where the master device sends a request to the slave device and waits for a response before sending the next request. Modbus supports various data types, including bits, bytes, integers, and floats. See Modbus Master Connector and Modbus Slave Connector for examples on the configuration and addresses for that protocol.

OPC UA (Open Platform Communications Unified Architecture)

It is a platform-independent protocol based on web services. It uses a client-server model to communicate with external devices and supports advanced security and encryption to protect transmitted information. See OPC UA Client Connector on how to configure the network addresses and Points addresses to that protocols.

AB Rockwell ControlLogix

This protocol is used to communicate with devices in the ControlLogix family from Rockwell Automation. It uses a master-slave architecture and supports various data types, including bits, bytes, integers, and floats. The ControlLogix protocol is widely used in the industry and is relatively easy to implement. See AB Rockwell ControlLogix and CompactLogix devices on how to configure the network addresses and Points addresses to that protocols.

> There are several other communication protocols available to connect to external devices using the software. Each protocol has its own specific characteristics and benefits that may be more suitable for different applications, or reflect standards used by hardware vendors. For the updated list of available protocols, go to: Communication Drivers.

Mapping Protocol Addresses

The following organization is created when mapping address to the Tags in the Application.

Address Syntax

Different protocols will have distinct ways to identify their internal elements, but essentially in all protocols, there is an address that will be mapped (for reading or writing) to a variable in the solution.

Read Groups and Write Groups

Read Groups and Write Groups are crucial for managing communication between the platform and external devices. They contain information about tags and communication operands, organizing and optimizing the data exchange process. A Read Group reads data from devices, while a Write Group writes data to devices. These groups are dynamically created based on the tags' configuration, such as Access Type, Polling Rate, and addresses, and are managed by the communication driver.

Communication Optimization

Efficient communication is vital for the performance of any industrial automation system. Several strategies can optimize communication between the platform and external devices, including:

- Grouping tags with similar polling rates and access types.
- Combining sequential addresses in a single request message to reduce the number of messages.
- Adjusting the MaximumBlockSize parameter according to the device's capabilities.
- Using multiple communication channels to prioritize specific requests or handle different devices concurrently.

Read On Display Access Type

The Read On Display Access Type setting enables communication only when data is displayed on the screen. This can improve performance for data that doesn't require constant updates or historical storage. Read Groups and Write Groups are created during the Devices Module startup without considering the Access Type. When a screen opens, the platform checks if any points have the Read On Display setting and are linked to the tags of the open screen. Points with Read On Display Access Type that are not displayed on the screen are disabled, and the associated group is only turned off if all its points are disabled.

Understanding how the Read On Display Access Type works is essential for efficient communication, as it helps reduce unnecessary data exchange. To learn more, see Access Types.

Communication Protocol Execution

Communication Messages

Communication messages are the foundation of data exchange between the platform and external devices. They transmit information from one entity to another, following a specific communication protocol. There are two types of communication messages:

- **TX messages**: Sent from the software to the device.
- **RX messages**: Sent from the device to the software.

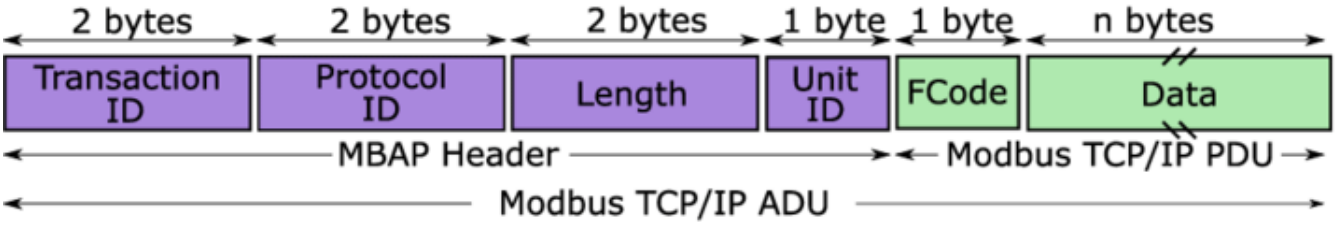
Both types contain a series of bytes, each with its specific meaning, following the rules established by the chosen communication protocol. Message types can include:

- Commands
- Response
- Data
- Status

Each device manufacturer and communication protocol creator defines communication message rules. These messages are called TX and RX. TX messages go from the software platform to the device, and RX messages go from the device to the software platform. Each entity in this “conversation” needs to know the communication protocol to understand what the TX and RX messages mean. Each byte within the message has a specific meaning.

Protocol Example: Modbus

To better understand the communication process, let's analyze Modbus TCP/IP, a widely used protocol in industrial automation that allows data exchange between various devices through a TCP/IP network. This protocol uses a client-server model, where the client (the platform) sends requests to the server (the device) to read or write data.



Modbus TCP/IP Message Structure

- **Transaction ID (2 bytes)**: This unique number helps match requests and responses.
- **Protocol ID (2 bytes)**: This is a constant value (0x0000) for Modbus.
- **Length Field (2 bytes)**: Specifies the length of the remaining message.
- **Unit ID (1 byte)**: The address of the remote device.
- **Function Code (1 byte)**: Indicates the requested action, such as read or write.
- **Data (variable)**: Contains the data to be read or written or error information.

Understanding the structure of a specific protocol is important for advanced scenarios where performance optimization or network-level diagnostics are necessary.

Protocol Example: STX and ETX

Consider the following fictitious communication protocol definition:

- **[STX]**: Start Message (02 hexa).
- **[CMD]**: Command (04 hexa for a Read Action, 05 hexa for a Write Action, or 06 hexa for Ack Answer).
- **[OPR]**: Operand (31 hexa for an Integer or 32 hexa for a Float).
- **[STA]**: Start Address - Beginning address to read or write (two bytes).
- **[CNT]**: Counter of the quantity of bytes in the data packet to read or write (one byte).
- **<DATA>**: Data bytes related to the content read or written (each operand can have 2 or 4 bytes).
- **[ETX]**: End Message (0D hexa).

Based on this communication protocol definition, the following TX and RX lines are valid.

- TX: 02 05 31 00 00 04 0D (Send to Device)
- RX: 02 06 31 00 00 04 00 00 00 00 0D (Device answer)

That means the TX is composed by:

02[STX]

05[CMD RD]

31 [Integer]

00 00 [Start Address : 0]

04[Quantity: 4 bytes]

0D [ETX]

The RX was composed by:

02 [STX]

06 [ACK ANSWER]

31 [Integer]

00 00 [Start Address : 0]

04 [Quantity: 4 bytes (two operands integer)]

00 00 [Operand 1 with data value 0]

00 00 [Operand 2 with data value 0]

0D [ETX]

Defining Communication Blocks

- If the CNT is a single byte indicating the number of bytes and the FF hex means 255, a communication block can have a maximum of 255 bytes. Thus, each TX/RX can transmit a maximum of 127 integers or 64 floats, which are the operand limits per communication block. The CNT, or 255 in this case, is the MaximumBlockSize, defined as the maximum size of the communication block.
- Since a device's available operands can be much larger than the MaximumBlockSize, multiple TX/RX exchanges may be required to read or write all the data.
- In a TX/RX, a read dataset is called a ReadGroup. Each ReadGroup contains information regarding FS Tags and communication operands. Consequently, the number of operands must be less than the maximum block size.
- ReadGroups and WriteGroups are created during the Devices Module startup. The Points settings are sorted, and the Groups are sequentially created by considering whether two different points can be in the same communication TX/RX. The information considered includes Node, Address (Operand, Address, Name), MaximumBlockSize, Read/Write Command, and Polling Time.

In this section...
