AVEVA™ Communication Drivers Pack – Allen Bradley - ABCIP Driver

User Guide

Publication date: Tuesday, May 9, 2023

Publication ID: 868875

Contact Information

AVEVA Group Limited
High Cross
Madingley Road
Cambridge
CB3 0HB. UK

https://sw.aveva.com/

For information on how to contact sales and customer training, see https://sw.aveva.com/contact.

For information on how to contact technical support, see https://sw.aveva.com/support.

To access the AVEVA Knowledge and Support center, visit https://softwaresupport.aveva.com.

Contents

Chapter 1

Introduction to the ABCIP Communication Driver

About the ABCIP Communication Driver

The ABCIP Communication Driver is a Microsoft Windows application that allows client applications direct and indirect access to Allen-Bradley families of ControlLogix, GuardLogix, FlexLogix, CompactLogix, SoftLogix 5800, MicroLogix, PLC-5, and SLC500 controllers.

The Communication Driver does not require the Rockwell Software RSLinx™ package.

Supported Client Protocols

Client applications connect to the ABCIP Communication Driver using:

- OPC
- SuiteLink
- DDE/FastDDE
- PCS

For more information refer to the "Support Client Protocols" section of the Communication Drivers Pack Help.

Supported Device Protocols

The ABCIP Communication Driver to communicate with the following supported Allen-Bradley controllers:

- PLC-5 controllers
- SLC500 controllers
- MicroLogix controllersCommunication Driver

Supported Device Networks

The ABCIP Communication Driver communicates with supported devices either directly or indirectly across the following device networks:

- ControlNet
- Data Highway 485 (DH485)
- Data Highway Plus (DH+)
- DeviceNet
- Ethernet

Supported Devices

The ABCIP Communication Driver provides direct and indirect connectivity to the following Allen-Bradley controllers:

- ControlLogix Controllers
- GuardLogix Controllers
- SoftLogix 5800 Controllers
- CompactLogix Controllers
- FlexLogix Controllers
- MicroLogix Controllers
- PLC-5 Controllers
- SLC500 Controllers

Note: The Optimize for Startup mode is not supported on Logix Controllers using firmware version 21 or above. For more information, see Logix5000 Optimization Mode.

ControlLogix Controllers

The 1756-EWEB enhanced Web-server module provides both CIP communications and Internet browser web-services. ABCIP Communication Driver supports ONLY CIP communications. The ABCIP Communication Driver is capable of accessing multiple ControlLogix processors in a single chassis.

- All ControlLogix processors (1756-series processors) directly accessible from the Ethernet using the ControlLogix Ethernet or EtherNet/IP Bridge module (1756-ENET, 1756-ENBT, 1756-EN2T, or 1756-EWEB) through the backplane.
- All ControlLogix processors (1756-series processors) that are Accessible through the ControlLogix Gateway from the ControlNet network by means of the ControlLogix ControlNet Bridge module (1756-CNB/CNBR or 1756-CN2/CN2R).

- All ControlLogix 1756-RM and 1757-SRM processors directly accessible from the Ethernet using the ControlLogix Ethernet or EtherNet/IP Bridge module (1756-EN2T for the 1756-RM or 1756-ENBT for the 1757-SRM) or accessible through the ControlLogix Gateway from the ControlNet Bridge module (1756-CN2R).

GuardLogix Controllers

- All GuardLogix Integrated Safety processors (1756-LSP & 1756-L6xS) directly accessible from the Ethernet using the ControlLogix Ethernet or EtherNet/IP Bridge module (1756-ENBT or 1756-EWEB) through the backplane.
- All GuardLogix Integrated Safety processors (1756-LSP & 1756-L6xS) accessible through the ControlLogix Gateway from the ControlNet network by means of the ControlLogix ControlNet Bridge module (1756-CNB/CNBR or 1756-CN2).

SoftLogix 5800 Controllers

- All SoftLogix 5800 controllers (1789-series) directly accessible from the Ethernet on an industrial or desktop PC.

CompactLogix Controllers

- All CompactLogix processors (5069/1769/1768-series) directly accessible from the Ethernet using the integrated EtherNet/IP port.
- All CompactLogix processors (1769/1768-series) accessible from the Ethernet via the EtherNet/IP interface module for CompactLogix/MicroLogix (1761-NET-ENI).

FlexLogix Controllers

- All FlexLogix processors (1794-series) accessible from the Ethernet using the EtherNet/IP communications daughter-card (1788-ENBT).
- All FlexLogix processors (1794-series) accessible through the ControlLogix Gateway from the ControlNet network by means of the ControlNet daughter-card (1788-CNC).

MicroLogix Controllers

- All MicroLogix 1100 processors (1763-series) and 1400 processors (1766-series) directly accessible from the Ethernet using the integrated EtherNet/IP port.
- All MicroLogix 1000/1200/1500 processors accessible from the Ethernet via the Ethernet/IP interface module for CompactLogix/MicroLogix (1761-NET-ENI) series B or higher.
- All MicroLogix 1000/1200/1500 processors accessible from the DH485 network using the RS-232C-to-DH485 Advanced Interface Converter module (1761-NET-AIC) to connect to the Data Highway Plus network through a DH+-to-DH485 Bridge module (1785-KA5) and routed through the ControlLogix Gateway by means of the ControlLogix DH+/RIO Bridge module (1756-DHRIO) to Ethernet.

PLC-5 Controllers

- All PLC-5 processors (1785-series) accessible through the ControlLogix Gateway from the Data Highway Plus network by means of the ControlLogix DH+/RIO Bridge module (1756-DHRIO)
- All ControlNet-capable PLC-5 processors (1785-series) accessible through the ControlLogix Gateway from the ControlNet network by means of the ControlLogix ControlNet Bridge module (1756-¬CNB/CNBR).

SLC500 Controllers

- All SLC 5/05 processors accessible from the Ethernet using the built-in EtherNet/IP interface.
- All SLC 5/03, /04 processors (1747-series) accessible from the Ethernet using the EtherNet/IP interface module (1761-NET-ENI).
- All SLC 5/04 processors (1747-series) accessible through the ControlLogix Gateway from the Data Highway Plus network by means of the ControlLogix DH+/RIO Bridge module (1756-DHRIO).
- All SLC 5/03, /04, /05 processors (1747-series) linked to the SLC500 ControlNet RS-232 interface module (1747-KFC15) accessible through the ControlLogix Gateway from the ControlNet network by means of the ControlLogix ControlNet Bridge module (1756-CNB).
- All SLC 5/03, /04, /05 processors (1747-series) accessible from the DH485 network using the RS-232C-to-DH485 Advanced Interface Converter module (1761-NET-AIC) to connect to the Data Highway Plus network through a DH+-to-DH485 Bridge module (1785-KA5) and routed through the ControlLogix Gateway by means of the ControlLogix DH+/RIO Bridge module (1756-DHRIO) to the Ethernet.

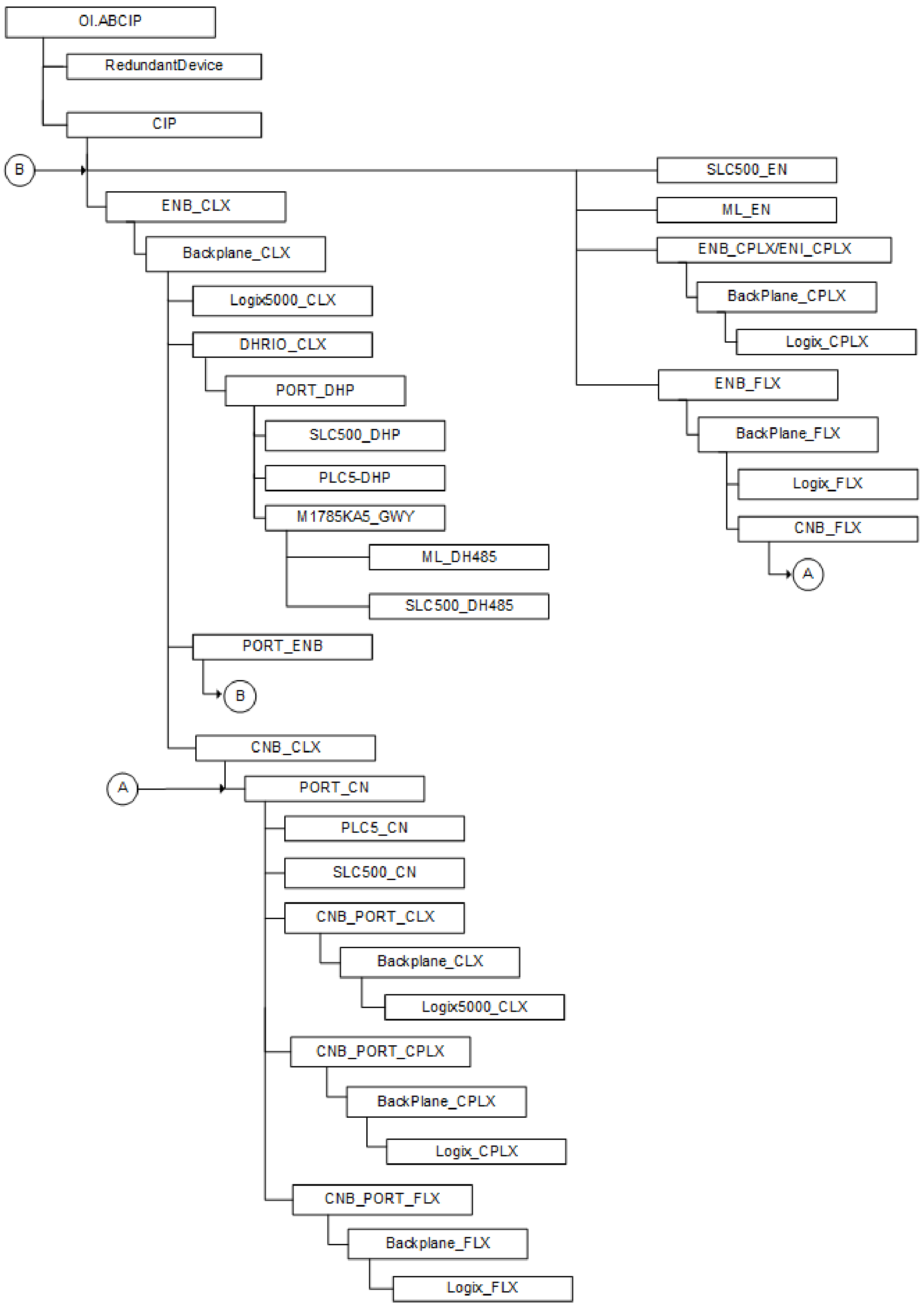
While primarily intended for use with InTouch®, the Communication Driver may be used by any Microsoft Windows program capable of acting as a DDE, FastDDE, SuiteLink™, or OPC client.

Supported Topologies

This ABCIP Communication Driver communicates with supported Allen-Bradley families of CompactLogix, ControlLogix, FlexLogix, GuardLogix, MicroLogix, PLC-5, SLC500, and SoftLogix 5800 controllers across:

- ControlNet
- Data Highway 485 (DH485)
- Data Highway Plus (DH+)
- DeviceNet
- Ethernet

Before attempting to configure your Communication Driver, you should determine the hierarchical structure of your network/controller environment.



Dual ENB Routing Topology

The ABCIP Communication Driver can connect to a ControlLogix rack via an ENB module on the same subnet, and route from the backplane to a second ENB module on a different subnet.

The ControlLogix rack becomes a router between two subnets. The ABCIP Communication Driver can connect to the following controllers and devices on a second Ethernet subnet:

- ControlLogix
- CompactLogix
- FlexLogix
- MicroLogix
- SLC500

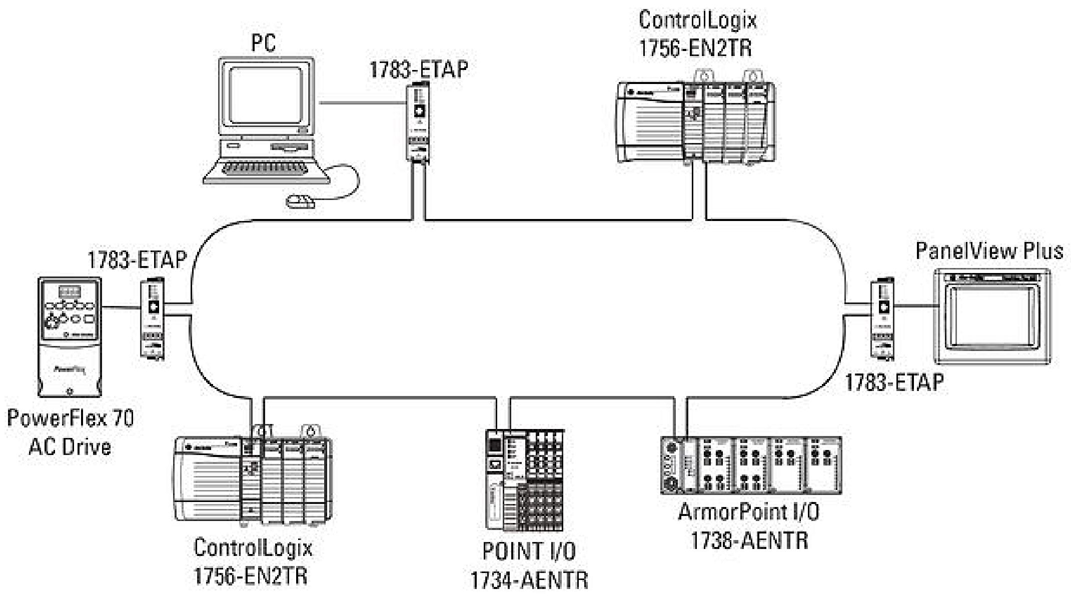
Note: Only one Ethernet subnet hop is supported.

Device-Level Ring (DLR) Topology

DLR is network technology provided by Rockwell Automation to enable Ethernet ring network topologies at the device level. The DLR protocol enables Ethernet devices to connect directly to neighboring nodes through dual network ports to form a ring topology.

When a DLR detects a break in the ring, it provides alternate routing of the data to help recover the network.

The ABCIP Communication Driver provides data connectivity to other supported controllers connected on the same DLR network. The computer where the ABCIP Communication Driver is installed must be connected to the DLR network via the 1783-ETAP device. Following is a sample DLR topology:



Windows Firewall Considerations

If the Communication Driver runs on a computer with a firewall enabled, a list of application names or port numbers must be put in the firewall exception list so the Communication Driver can function correctly. The Communication Driver installation program makes the required entries in the firewall exception list for you.

The following applications are added to the firewall exception list on the computer where the Communication Driver run-time application is installed:

- ABCIP.exe
- aaLogger.exe
- DASAgent.exe
- dllhost.exe
- mmc.exe
- OPCEnum.exe
- Slssvc.exe

The following port numbers are added to the firewall exception list on the computer where the Communication Driver run-time application is installed:

- 5413 - TCP port for slssvc.exe
- 445 - TCP port for file and printer sharing
- 135 - TCP port for DCOM

The following applications are added to the firewall exception list on the computer where the OI Server Manager (configuration part) is installed:

- aaLogger.exe
- dllhost.exe
- mmc.exe

The following port numbers are added to the firewall exception list on the computer where the OI Server Manager (configuration part) is installed:

- 445 - TCP port for file and printer sharing
- 135 - TCP port for DCOM

Un-installing the Communication Driver does not remove the firewall exception list entries. You must delete the firewall exception list entries manually. For more information on how to do this, see your firewall or Windows security documentation.

Chapter 2

Configuring the ABCIP Communication Driver

Configuring Port Objects for the ABCIP Communication Driver

Network Communication Bridge/Interface Modules are the communication links between the ABCIP Communication Driver and its supported Allen-Bradley controllers. You must create these links within the OI Server Manager hierarchy to bridge/route control and information data between different networks to target controllers.

This is accomplished by creating Port Objects. These Port Objects simulate the physical hardware layout and must be built to establish communications with each of the controllers. Once you have built the ABCIP hierarchy, you can configure the respective devices for communications. Finally, you can create the desired Device Groups for each controller.

Before you add these Ports in the OCMC, you need to identify your hardware topology to the devices being connected.

Once you have established this hierarchy you will then add, rename, or delete Port objects to accurately represent how your network is organized.

Adding, Renaming, Deleting Port Objects

Use the procedures described in this section to add, rename, or delete port objects.

Adding a Port

The first step in specifying the network between the Communication Driver and a device is to add Port objects. After you add the necessary Ports depicting your network, you will then be able to add and communicate with your devices.

To add a port

1. Open the OI Server Manager in the OCMC.
2. Locate and expand the target Communication Driver group hierarchy you wish to add ports to.

3. Right-click the default **Configuration** node and select the applicable **Add Port Object**. The console tree will now show the new port with its default port name selected.
4. Edit the name as needed and press Enter.

Renaming a Port

After you create ports in the OI Server Manager, it may be necessary to rename them to work with your client applications.

To change an existing port object name

1. In the OI Server Manager, expand the Communication Driver hierarchy tree to display the target port object node you wish to rename.
2. Select and right-click the port object's name (or press the <F2> key). Click **Rename.**
3. Type the new name and press **Enter**.

Note: Changing the port name prevents clients from registering data using the old name. Data for existing queries is set to bad quality. It is recommended not to make changes to parameters like the Port name after you develop a large client application.

Deleting a Port

If your hardware network topology is changed you may need to delete a port object.

When you delete a port, all nodes below the port in its hierarchy (child nodes) are also deleted. If a client application requests new data from a deleted port or a node on a deleted port, the request is rejected. Data for existing queries is set to bad quality.

To delete a port

1. In the OI Server Manager, expand the Communication Driver hierarchy tree to display the target port object node you wish to delete.
2. Right-click the port object node to be deleted and click **Delete**.
3. Read the warning and then click **Yes**. The port object and all nodes (devices) below it in the hierarchy are deleted.

Configuring ABCIP Communication Driver Ports

The ABCIP Communication Driver hierarchy in the OI Server Manager starts with the PORT_CIP Object, followed by the supported communication-interface/gateway modules that allow the Communication Driver to access the supported networks and devices. The logical endpoint for each branch of the ABCIP hierarchy tree is always a Processor Type node which represents the controller device.

The following sections detail the steps necessary to configure your Communication Driver Port Objects according to your network type.

Note: Before attempting to configure your Communication Driver, you should determine the hierarchical structure of your device/network environment.

CIP Port Connection Set-up

The Communication Driver hierarchy tree under the OI Server Manager starts at the PORT_CIP port object. It is a logical representation of the Ethernet port for CIP communications on a computer.

Note: Only one PORT_CIP connection is allowed per ABCIP Communication Driver.

To create PORT_CIP objects from the Configuration branch

1. Select and right-click on **Configuration**.
2. Select **Add PORT_CIP Connection** from the shortcut menu. A New_PORT_CIP_000 object is created.
3. Rename the newly created object as appropriate. The **Port_CIP_000 Parameters** configuration view is displayed in the Configuration branch of the hierarchy.

This configuration view has two parameters, one of which is configurable:

- **Port Type**: The information is provided automatically by the OI Server Manager (CIP).
- **Maximum Queued Msgs**: The default number of unconnected messages that the Communication Driver can send to a device before a reply is received. When this number is reached, the Communication Driver queues messages until a reply is received from the device.
  - The valid range is 1 - 40.
  - The default value is 4.

  Note:

  - Applies to MicroLogix or SLC500 Connections, only if not using CIP Connections.

  - Simultaneous Message Configuration for Logix Controllers is set in Device Configuration.

The Ethernet Network

Through the PORT_CIP object, the ABCIP Communication Driver accesses data from the ControlLogix, CompactLogix, FlexLogix, GuardLogix, MicroLogix, SLC500, and SoftLogix 5800 controllers on the Ethernet network that uses the EtherNet/IP protocol.

ENB_CLX Object

The ENB_CLX object represents the physical Allen-Bradley EtherNet/IP Communications module within a ControlLogix chassis.

- 1756-ENET
- 1756-ENBT
- 1756-EN2T
- 1756-EWEB

The ENB_CLX object is hosted by CIP.

Note: A maximum of 65535 ENB_CLX objects can be created for the Communication Driver.

To add ENB_CLX objects to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CIP_000** object.
2. Select **Add ENB_CLX Connection** from the shortcut menu. A New_ENB_CLX_000 object is created.

3. Rename the newly created object as appropriate. The **ENB_CLX Parameters** configuration view is displayed.

This configuration view has three parameters, two of which are configurable:

- **Module Type:** Information provided automatically by the OI Server Manager (Ethernet Comm).
- **Host Name:** Host Name or IP Address of the destination 1756-ENET/ENBT/EN2T/EWEB module.
  - The Host Name is defined in the system Host file, usually found in: \Windows\system32\drivers\etc\hosts folder.
  - Type in the network address where the PLC is located (for example, "10.11.12.13") or type in a host name if one is defined in the LocalHost list. The number of characters cannot be more than 255. The field cannot be blank.

  Note: The Host Name defaults to the LocalHost. If the LocalHost is selected and deleted, resulting in a blank Host Name box, and you apply the changes, this will result in an error message.

  Important: If setting up a SoftLogix or GuardLogix device, the host or IP address of the corresponding SoftLogix or physical GuardLogix device must be entered in the Ethernet/IP Bridge Module (ENB) node within the ABCIP Communication Driver hierarchy to establish communications with the device. For more information, see SoftLogix 5800 Controllers or GuardLogix Controllers.
- **Connection Timeout:** Time (in milliseconds) allowed for establishing a socket connection to a target device.
  - The valid range is 10 - 10000 milliseconds.
  - The default value is 2000.
  - The connection timeout is used if the object is underneath PORT_CIP.

BACKPLANE_CLX Object

The BackPlane_CLX object represents the physical backplane of an Allen-Bradley ControlLogix controller chassis.

This object is hosted by the ENB_CLX and CNB_Port_CLX objects.

Note: Only one instance of the BACKPLANE_CLX object can be created per ENB_CLX and CNB_Port_CLX branch. The Communication Driver is capable of operating with multiple ControlLogix processors in a single backplane.

To add the BACKPLANE_CLX object to your ABCIP hierarchy

1. Select and right-click on the **New_ENB_CLX_000** object.
2. Select **Add BACKPLANE_CLX Connection** from the shortcut menu. The New_BACKPLANE_CLX_000 object is created.
3. Rename the newly created object as appropriate. The **BACKPLANE_CLX Parameters** configuration view is displayed.

This configuration view has one element:

- **Device Type:** The information is provided automatically by the OI Server Manager (BackPlane).

PORT_ENB Object

The Port_ENB object represents the physical Ethernet port for the Allen-Bradley Ethernet Network bridge module.

This object is hosted by the BACKPLANE_CLX object.

To add the PORT_ENB object to your ABCIP hierarchy

1. Select and right-click on the New_BACKPLANE_CLX_000 object.
2. Select **Add PORT_ENB Connection** from the shortcut menu. The New_PORT_ENB_000 object is created.
3. Rename the newly created object as appropriate. The **PORT_ENB Parameters** configuration view is displayed.

This configuration view has the following elements:

- **Module Type**: Information provided automatically by the OI Server Manager (Ethernet Communication).
- **Slot Number**: A sequential number beginning with 0 (zero) assigned to each slot in a ControlLogix chassis.
  - The slot number indicates where the module resides in the parent backplane.
  - The valid range is 0 - 16.
  - The default value is 0 (zero).

LOGIX5000_CLX Object

The Logix5000_CLX object is a logical representation of the Allen-Bradley ControlLogix processor modules within a ControlLogix chassis.

- 1756-L1
- 1756-L55
- 1756-L6x
- 1756-L7x

The Logix5000_CLX object is also a logical representation of the following Allen-Bradley processor modules:

- SoftLogix 5800
- GuardLogix 1756-L6xS

This object is hosted by BackPlane_CLX.

To add the LOGIX5000_CLX object to your ABCIP hierarchy

1. Select and right-click on the **New_BACKPLANE_CLX_000** object.
2. Select **Add LOGIX5000_CLX Connection** from the shortcut menu. The New_LOGIX5000_CLX_000 object is created.
3. Rename the newly created object as appropriate. The **Logix5000_CLX Parameters** configuration view is displayed.

This configuration view has nine parameters, eight of which are configurable:

- **Processor Type**: Information provided automatically by the OI Server Manager (ControlLogix /GuardLogix / SoftLogix).
- **Slot Number**: A sequential number beginning with 0 (zero) assigned to each slot in a ControlLogix chassis.
  - The slot number indicates where the module resides in the parent backplane.
  - The valid range is 0 - 16.
  - The default value is 0 (zero).

- **Reply Timeout**: Time (in seconds) the Communication Driver will wait for the acknowledgement after it sends out a message. The message will be resent when time-out occurs.
  - The valid range is 1 - 300 seconds.
  - The default value is 15.
- **Max CIP Connections**: The maximum number of CIP connections that can be originated from the Communication Driver to this device.
  - The valid range is 1 - 31.
  - The default value is 4 (four).

  Note: For each CIP connection established, the PLC allocates certain resources to handle the connection. If too many CIP connections are established by the Communication Driver, the PLC processor allocates more resources for the CIP, and other operations will suffer. For example, with 31 connections, the data updates become extremely sluggish since the PLC ran out of resources to handle the runtime logic and updating the control items.
- **Optimization Mode** (For detailed information, see Logix5000 Optimization Mode):
  - **No optimization**: The server uses the most basic communication method available by using the tag name for each communication with the controller. The tag database will be uploaded from the processor to validate the tag names.
  - **Optimize for read** (Default): All tags are accessed by predefining messages in the controller, thus optimizing blocks of information from the controller. Initialization of this mode requires that these message blocks are built when connecting to the controller, therefore startup time will require more time. This mode is most effective with a large number of tags on continuous scan.
  - **Optimize for startup time**: This option provides the best overall performance. All tags are accessed from the Logix processor using the device's memory location table. If this option is checked, the 'Auto Synchronize Tag' option is checked automatically and cannot be unchecked.

  Note: "Optimize for Startup time" is not supported on Logix Controllers using firmware version 21 or above. For more information, see Logix5000 Optimization Mode.
- **Optimize User Defined Data Types**: The optimization for reading structures is enabled when selected (Default is unchecked). For more detailed information, see UDT Optimization.

  If selected, the server will retrieve the whole structure in one packet provided the size of the structure is 488 bytes or less.
- **Tag Database Options**: Three options are selectable to implement manual or automated updates of the Logix processor's tag database. For more information, see Logix5000 Online Tag Management
  - Auto Load Tags on Startup (Default)
  - Auto Synchronize Tags
  - Use Persisted Tags (Default)

  Note: If "Optimize for startup time" is selected, the "Auto Synchronize Tags" option will be automatically selected and will not be changeable (option will be dimmed). The Communication Driver will need to synchronize physical address tags from the device.

Important: Support for secured Logix5000 controllers will affect the way the 'Auto Synchronize Tags' and 'Persisted Tags' behave. For detailed information, see Accessing Secured Logix5000-series Controllers.

ENB_FLX Object

The ENB_FLX object represents the physical Allen-Bradley FlexLogix Ethernet Communication Daughter Card.

- 1788-ENBT

This object is hosted by the CIP Network Object.

Note: A maximum of 65535 ENB_FLX objects can be created for the Communication Driver.

To add ENB_FLX objects to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CIP_000** object.
2. Select **Add ENB_FLX Connection** from the shortcut menu. A New_ENB_FLX_000 object is created.
3. Rename the newly created object as appropriate. The **ENB_FLX Parameters** configuration view is displayed.

This configuration view has three parameters, two of which are configurable:

- **Module Type**: Information provided automatically by the OI Server Manager (Ethernet Communication).
- **Host Name**: The Host Name or IP Address of the destination 1788-ENBT module.
  - The Host Name is defined in the system Host file, usually found in: \Windows\system32\drivers\etc\hosts folder.
  - The number of characters cannot be more than 255. The field cannot be blank.

  Note: The Host Name defaults to the LocalHost. If the LocalHost is selected and deleted, resulting in a blank Host Name box, and you apply the changes, this will result in an error message.
- **Connection Timeout**: Time (in milliseconds) allowed for establishing a socket connection to a target device.
  - The valid range is 10 - 10000 milliseconds.
  - The default value is 2000.
  - The connection timeout is used if the object is underneath PORT_CIP.

BACKPLANE_FLX Object

The BackPlane_FLX object represents the physical backplane of an Allen-Bradley FlexLogix controller assembly.

This object is hosted by ENB_FLX and CNB_Port_FLX.

Note: The Communication Driver is capable of operating with multiple FlexLogix processors in a single backplane. Only one instance of the BACKPLANE_FLX object can be created per ENB_FLX branch.

To add the BACKPLANE_FLX object to your ABCIP hierarchy

1. Select and right-click on the New_ENB_FLX_000 object.
2. Select **Add BACKPLANE_FLX Connection** from the shortcut menu. A New_BACKPLANE_FLX_000 object is created.
3. Rename the newly created object as appropriate. The **BACKPLANE_FLX Parameters** configuration view is displayed.

This configuration view has one element:

Device Type: The information is provided automatically by the OI Server Manager (Backplane).

LOGIX_FLX Object

The Logix_FLX object represents the physical Allen-Bradley FlexLogix processor module.

- 1794-Lxx

This object is hosted by BACKPLANE_FLX.

To add the LOGIX_FLX object to your ABCIP hierarchy

1. Select and right-click on the **NEW**_BACKPLANE_FLX_000 object.
2. Select **Add LOGIX_FLX Connection** from the shortcut menu. The New_LOGIX_FLX_000 object is created.
3. Rename the newly created object as appropriate. The **LOGIX_FLX Parameters** configuration view is displayed.

This configuration view has nine parameters, eight of which are configurable:

Processor Type: (not configurable) Information provided automatically by the OI Server Manager (FlexLogix).

Slot Number: A sequential number beginning with 0 (zero) assigned to each slot in a FlexLogix chassis.

- The slot number indicates where the module resides.
- The valid range is 0 - 16.
- The default value is 0.

Reply Timeout: Time (in seconds) the Communication Driver will wait for the acknowledgement after it sends out a message. The message will be resent when time-out occurs.

- The valid range is 1 - 300.
- The default value is 15.

Max CIP Connections: The maximum number of CIP connections that can be originated from the Communication Driver to this device.

- The valid range is 1 - 31.
- The default value is 4 (four).

Note: For each CIP connection established, the PLC allocates certain resources to handle the connection. If too many CIP connections are established by the Communication Driver, the PLC processor allocates more resources for the CIP, and other operations will suffer. For example, with 31 connections, the data updates become extremely sluggish since the PLC ran out of resources to handle the runtime logic and updating the control items.

Optimization (For detailed information, see Logix5000 Optimization Mode):

- No optimization: The server uses the most basic communication method available by using the tag name for each communication with the controller. The tag database will be uploaded from the processor to validate the tag names.
- Optimize for read (Default): All tags are accessed by predefining messages in the controller, thus optimizing blocks of information from the controller. Initialization of this mode requires that these

message blocks are built when connecting to the controller, therefore startup time will require more time. This mode is most effective with large number of tags on continuous scan.

- Optimize for startup time: This option provides the best overall performance. All tags are accessed from the Logix processor using the device's memory location table. If this option is checked, the 'Auto Synchronize Tag' option is checked automatically and cannot be unchecked.

Note: "Optimize for Startup time" is not supported on Logix Controllers using firmware version 21 or above. For more information, see Logix5000 Optimization Mode.

Optimize User Defined Data Types: The optimization for reading structures is enabled when selected (Default is unchecked). For more detailed information, see UDT Optimization.

- If selected, the server will retrieve the whole structure in one packet provided the size of the structure is 488 bytes or less.

Tag Database Options: Three options are selectable to implement manual or automated updates of the Logix processor's tag database. For more information, see Logix5000 Online Tag Management.

- Auto Load Tags on Startup (Default)
- Auto Synchronize Tags
- Use Persisted Tags (Default)

Note: If the Optimization setting is selected for "Optimize for startup time", the "Auto Synchronize Tags" option is automatically selected and unchangeable (dimmed). The Communication Driver will need to synchronize physical address tags from the device.

Important: Support for secured Logix5000 controllers will effect the way the 'Auto Synchronize Tags' and 'Persisted Tags' behave. For detailed information, see Accessing Secured Logix5000-series Controllers.

ML_EN Object

The ML_EN object represents the physical Allen-Bradley MicroLogix processor with the built-in EtherNet/IP port or coupled with the Ethernet Interface module for MicroLogix and CompactLogix (1761-NET-ENI).

- 1763-L16xxx
- 1761-L10xxx, 1761-L16xxx, 1761-L20xxx, 1761-L32xxx
- 1762-L24xxx, 1762-L40xxx
- 1764-LSP, 1764-LRP

This object is hosted by CIP Network Object

To add ML_EN objects to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CIP_000** object.
2. Select **Add ML_EN Connection** from the shortcut menu. A New_ML_EN_000 object is created.
3. Rename the newly created object as appropriate. The **ML_EN Parameters** configuration view is displayed.

This configuration view has six parameters, five of which are configurable:

- **Processor Type:** (not configurable) Information provided automatically by the OI Server Manager (MicroLogix).

- **Host Name:** The Host Name or IP Address of the destination MicroLogix processor or 1761-NET-ENI module connected to a MicroLogix processor.
    - The Host Name is defined in the system Host file, usually found in: \Windows\system32\drivers\etc\hosts folder.
    - The number of characters cannot be more than 255. The field cannot be blank.
- **Reply Timeout:** Time (in seconds) the Communication Driver will wait for the acknowledgement after it sends out a message. The message will be resent when time-out occurs.
    - The valid range is 1 - 300 seconds.
    - The default value is 15.
- **Connection Timeout**: Time (in milliseconds) allowed for establishing a socket connection to a target device.
    - The valid range is 10 - 10000 milliseconds.
    - The default value is 2000.
    - The connection timeout is used if the ML_EN object is beneath PORT_CIP.
- **Use CIP Connection:** This option specifies if the CIP connection should be used to communicate with the MicroLogix controller. It must be selected to support MicroLogix model 1100/1400-series controllers with direct CIP connection. It is optional for all other MicroLogix models.
    - The default value is True.

    Note: The number of CIP connections in the controllers are limited (See Max CIP Connections below).
- **Maximum CIP Connections:** The maximum number of CIP connections which can be originated from the Communication Driver to this device.
    - The valid range is 1 - 31.
    - The default value is 1 (one).

    Note: Max CIP Connections setting available only if the CIP connection is selected.

The logical endpoint for each branch of the ABCIP hierarchy tree is always a Processor Type node, which represents the controller device.

ENB_CPLX Object

The ENB_CPLX object represents the physical integrated EtherNet/IP port on the Allen-Bradley CompactLogix Ethernet processor.

This object is hosted by CIP Network Object

Note: A maximum of 65536 ENB_CPLX objects can be created for the Communication Driver.

To add ENB_CPLX objects to your ABCIP hierarchy

1. Select and right-click on the New_PORT_CIP_000 object.
2. Select **Add ENB_CPLX Connection** from the shortcut menu. A New_ENB_CPLX_000 object is created.
3. Rename the newly created object as appropriate. The **ENB_CPLX Parameters** configuration view appears.

This configuration view has three parameters, two of which are configurable:

- **Module Type:** (not configurable) Information provided automatically by the OI Server Manager (Ethernet Communication).
- **Host Name:** The Host Name or IP Address of the destination Ethernet-capable CompactLogix processor.
  - The Host Name is defined in the system Host file, usually found in: \Windows\system32\drivers\etc\hosts folder.
  - The number of characters cannot be more than 255. The field cannot be blank.

  Note: The Host Name defaults to the LocalHost. If the LocalHost is selected and deleted, resulting in a blank Host Name box, and you apply the changes, this will result in an error message.
- **Connection Timeout:** Time (in milliseconds) allowed for establishing a socket connection to a target device.
  - The valid range is 10 - 10000 milliseconds.
  - The default value is 2000.
  - The connection timeout is used if the ENB_CPLX object is beneath PORT_CIP.

ENI_CPLX Object

The ENI_CPLX object represents the physical Allen-Bradley Ethernet Interface module for MicroLogix and CompactLogix (1761-NET-ENI).

- 1761-NET-ENI Module

This object is hosted by CIP Network Object

Note: A maximum of 65535 ENI_CPLX objects can be created for the Communication Driver.

To add ENI_CPLX objects to your ABCIP hierarchy

1. Select and right-click on the New_PORT_CIP_000 object.
2. Select A**dd ENI_CPLX Configuration** from the shortcut menu. A New_ENI_CPLX_000 object is created.
3. Rename the newly created object as appropriate. The **ENI_CPLX Parameters** configuration view is displayed.

This configuration view has three parameters, two of which are configurable:

- **Module Type:** (not configurable) Information provided automatically by the OI Server Manager (Ethernet Interface).
- **Host Name:** The Host Name or IP Address of the destination 1761-NET-ENI module.
  - The Host Name is defined in the system Host file, usually found in: \Windows\system32\drivers\etc\hosts folder.

  Note: The Host Name defaults to the LocalHost. If the LocalHost is selected and deleted, resulting in a blank Host Name box, and you apply the changes, this will result in an error message.
- **Connection Timeout:** Time (in milliseconds) allowed for establishing a socket connection to a target device.
  - The valid range is 10 - 10000 milliseconds.
  - The default value is 2000.
  - The connection timeout is used if the ENI_CPLX object is beneath PORT_CIP.

BACKPLANE_CPLX Object

The BACKPLANE_CPLX object represents the physical backplane of a CompactLogix controller assembly.

This object is hosted by ENB_CPLX and ENI_CPLX.

Note: The Communication Driver is capable of operating with multiple CompactLogix processors in a single backplane. Only one instance of the BACKPLANE_CPLX object can be created per ENB_CPLX or ENI_CPLX branch.

To add the BACKPLANE_CPLX object to your ABCIP hierarchy

1. Select and right-click on the New_ENB_CPLX_000 or New_ENI_CPLX_000 object.
2. Select **Add BACKPLANE_CPLX Connection** from the shortcut menu. The New_BACKPLANE_CPLX_000 object is created.
3. Rename the newly created object as appropriate. The **BACKPLANE_CPLX Parameters** configuration view appears.

This configuration view has one element:

Device Type: The information is provided automatically by the OI Server Manager (Backplane).

LOGIX_CPLX Object

The LOGIX_CPLX object represents the physical CompactLogix processor module.

- 1768-Lxx
- 1769-Lxx

This object is hosted by BACKPLANE_CPLX.

To add the LOGIX_CPLX object to your ABCIP hierarchy

1. Select and right-click on the **NEW_BACKPLANE_CPLX_000** object.
2. Select **Add LOGIX_CPLX Connection** from the shortcut menu. The New_LOGIX_CPLX_000 object is created.
3. Rename the newly created object as appropriate. The **LOGIX_CPLX Parameters** configuration view is displayed.

This configuration view has nine parameters, eight of which are configurable:

- **Processor Type:** (not configurable) Information provided automatically by the OI Server Manager (LOGIX5000).
- **Slot Number:** A sequential number beginning with 0 (zero) assigned to each slot in a Logix5000 chassis.
  - The slot number indicates where the module resides.
  - The valid range is 0 - 16.
  - The default value is 0 (zero).
- **Reply Timeout:** Time (in seconds) the Communication Driver will wait for the acknowledgement after it sends out a message. The message will be resent when time-out occurs.
  - The valid range is 1 - 300.
  - The default value is 15.

- **Max CIP Connections:** The maximum number of CIP connections which can be originated from the Communication Driver to this device.
  - The valid range is 1 - 31
  - The default value is 4 (four)

  Note: For each CIP connection established, the PLC allocates certain resources to handle the connection. If too many CIP connections are established by the Communication Driver, the PLC processor allocates more resources for the CIP, and other operations will suffer. For example, with 31 connections, the data updates become extremely sluggish since the PLC ran out of resources to handle the runtime logic and updating the control items.
- **Optimization** (For detailed information, see Logix5000 Optimization Mode):
  - No optimization: The server uses the most basic communication method available by using the tag name for each communication with the controller. The tag database will be uploaded from the processor to validate the tag names.
  - Optimize for read (Default): All tags are accessed by predefining messages in the controller, thus optimizing blocks of information from the controller. Initialization of this mode requires that these message blocks are built when connecting to the controller, therefore startup time will require more time. This mode is most effective with large number of tags on continuous scan.
  - Optimize for startup time: This option provides the best overall performance. All tags are accessed from the Logix processor using the device’s memory location table. If this option is checked, the ‘Auto Synchronize Tag’ option is checked automatically and cannot be unchecked.

  Note: "Optimize for Startup time" is not supported on Logix Controllers using firmware version 21 or above. For more information, see Logix5000 Optimization Mode.
- **Optimize User Defined Data Types:** The optimization for reading structures is enabled when selected (Default is unchecked). For more detailed information, see UDT Optimization.
  - If selected, the server will retrieve the whole structure in one packet provided the size of the structure is 488 bytes or less.
- **Tag Database Options:** Three options are selectable to implement manual or automated updates of the Logix processor’s tag database. For more information, see Logix5000 Online Tag Management.
  - Auto Load Tags on Startup (Default)
  - Auto Synchronize Tags
  - Use Persisted Tags (Default)

  Note: If the Optimization setting is selected for "Optimize for startup time", the "Auto Synchronize Tags" option is automatically selected and unchangeable (dimmed). The Communication Driver will need to synchronize physical address tags from the device.

  Important: Support for secured Logix5000 controllers will affect the way the ’Auto Synchronize Tags’ and ’Persisted Tags’ behave. For detailed information, see Accessing Secured Logix5000-series Controllers.

SLC500_EN Object

The SLC500_EN object represents the physical Allen-Bradley SLC500 and SLC505 processors connected to an Allen-Bradley Ethernet Interface for MicroLogix and CompactLogix (1761-NET-ENI).

- 1747-L55x
- 1747-L5xx with 1761-NET-ENI

This object is hosted by CIP Network Object

To add SLC500_EN objects to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CIP_000** object.
2. Select **Add SLC500_EN Connection** from the shortcut menu. A New_SLC500_EN_000 object is created.
3. Rename the newly created object as appropriate. The **SLC500_EN Parameters** configuration view is displayed.

This configuration view has six parameters, five of which are configurable:

- **Processor Type:** (not configurable) Information provided automatically by the OI Server Manager (SLC500).
- **Host Name**: The Host Name or IP Address of the destination 1761-NET-ENI Module.
  - The Host Name is defined in the system Host file, usually found in: \Windows\system32\drivers\etc\hosts folder.
  - The number of characters cannot be more than 255. The field cannot be blank.
- **Reply Timeout:** Time (in seconds) the Communication Driver will wait for the acknowledgement after it sends out a message. The message will be resent when time-out occurs.
  - The valid range is 1 - 300 seconds.
  - The default value is 15.
- **Connection Timeout:** Time (in milliseconds) allowed for establishing a socket connection to a target device.
  - The valid range is 10 - 10000 milliseconds.
  - The default value is 2000.
  - The connection timeout is used if the SLC500_EN object is beneath PORT_CIP.
- **Use CIP Connection:** Must be selected to support SLC 5/05-series controllers with direct CIP connection.
  - The default value is True.
- **Maximum CIP Connections:** The maximum number of CIP connections which can be originated from the Communication Driver to this device.
  - The valid range is 1 - 31.
  - The default value is 4 (four).

  Note: Max CIP Connections setting available only if the direct CIP connection is selected. For each CIP connection established, the PLC allocates certain resources to handle the connection. If too many CIP connections are established by the Communication Driver, the PLC processor allocates more resources for the CIP, and other operations will suffer. For example, with 31 connections, the data updates become extremely sluggish since the PLC ran out of resources to handle the runtime logic and updating the control items.

The ControlNet Network

Routing through the CNB_CLX or the CNB_FLX object from Ethernet network, the ABCIP Communication Driver accesses data from ControlLogix, GuardLogix, CompactLogix, FlexLogix, PLC-5, and SLC500 processors over the ControlNet network.

CNB_CLX Object

The CNB_CLX object represents the physical Allen-Bradley ControlLogix ControlNet Bridge module.

- 1756-CNB(R)
- 1756-CN2(R)

This object is hosted by BACKPLANE_CLX.

To add the CNB_CLX object to your ABCIP hierarchy

1. Select and right-click on the **New_BACKPLANE_CLX_000** object.
2. Select **Add CNB_CLX Connection** from the shortcut menu. The **New_CNB_CLX_000** object is created.
3. Rename the newly created object as appropriate. The **CNB_CLX Parameters** configuration view is displayed.

This configuration view has two parameters, one of which is configurable:

- **ModuleType:** (not configurable) Information provided automatically by the OI Server Manager (ControlNet Communication)
- **Slot Number:** A sequential number beginning with 0 (zero) assigned to each slot in a ControlNet communications interface module.
    - The slot number indicates where the sub-module resides.
    - The valid range is 0 - 16.
    - The default value is 0 (zero).

CNB_FLX Object

The CNB_FLX object represents the physical Allen-Bradley FlexLogix ControlNet Communication Daughter Card.

- 1788-CNC(R)
- 1788-CNF(R)

This object is hosted by BACKPLANE_FLX.

To add the CNB_FLX object to your ABCIP hierarchy

1. Select and right-click on the **NEW_BACKPLANE_FLX_000** object.
2. Select **Add CNB_FLX Connection** from the shortcut menu. The New_CNB_FLX_000 object is created.
3. Rename the newly created object as appropriate. The **CNB_FLX Parameters** configuration view is displayed.

This configuration view has two parameters, one of which is configurable:

- **ModuleType:** (not configurable) The information is provided automatically by the OI Server Manager (ControlNet Comm.)
- **Slot Number:** A sequential number beginning with 0 (zero) assigned to each slot in a FlexLogix chassis.
    - The slot number indicates where the module resides.

- The valid range is 0 - 16.
- The default value is 0.

Note: ABCIP Communication Driver supports single hops from one ControlNet link to another for accessing data in the target ControlLogix or FlexLogix processor. That is, an additional level of Logix_CLX or Logix_FLX object can be populated under the respective BACKPLANE_CLX_000 or BACKPLANE_FLX_000 object along the CNB_CLX or CNB_FLX hierarchy branch.

PORT_CN Object

The Port_CN object represents the physical ControlNet port for the Allen-Bradley ControlNet Bridge module.

This object is hosted by CNB_CLX and CNB_FLX.

To add the PORT_CN object to your ABCIP hierarchy

1. Select and right-click on the **New_CNB_CLX_000** object.
2. Select **Add PORT_CN Connection** from the shortcut menu. The **New_PORT_CN_000** object is created.
3. Rename the newly created object as appropriate. The **PORT_CN Parameters** configuration view is displayed.

This configuration view has three parameters, two of which are configurable:

- **Port Type:** (not configurable) Information provided automatically by the OI Server Manager (ControlNet).
- **Channel Number:** The number of physical channels/ports used on the ControlNet network.
  - Select Channel A or Channel B.
- **ControlNet Address:** The node address on the ControlNet network.
  - The valid range is 1 - 99 decimal.
  - The default value is 1 (one).

PLC5_CN Object

The PLC5_CN object represents the physical Allen Bradley ControlNet-capable PLC-5 processor.

- 1785-LxxC

This object is hosted by PORT_CN.

To add the PLC5_CN object to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CN_000** object.
2. Select **Add PLC5_CN Connection** from the shortcut menu. The **New_PLC5_CN_000** object is created.
3. Rename the newly created object as appropriate. The **PLC5_CN Parameters** configuration view is displayed.

The configuration view contains five parameters, four of which are configurable:

- **Processor Type:** (not configurable) The information is automatically provided (PLC-5).
- **ControlNet Address:** The node address on the Control Net network (Octal).
  - The valid range is 1 - 99 decimal.
  - The default value is 1 (one).
- **Source Link ID:** The source link ID of the module. This link ID has to match what has been defined in the ControlNet Routing table for the ControlNet network.

    - The valid range is 1 - 199.
    - The default value is 1 (one).
- **Reply Timeout:** Enter the maximum amount of time (in seconds) that the Communication Driver will wait for a response from the controller.
    - The valid range is 1 - 300 seconds.
    - The default value is 15 seconds.
- **Max CIP Connections:** The maximum number of CIP connections which can be originated from the Communication Driver to this device.
    - The valid range is 1 - 31.
    - The default value is 4 (four).

    Note: For each CIP connection established, the PLC allocates certain resources to handle the connection. If too many CIP connections are established by the Communication Driver, the PLC processor allocates more resources for the CIP, and other operations will suffer. For example, with 31 connections, the data updates become extremely sluggish since the PLC ran out of resources to handle the runtime logic and updating the control items.

SLC500_CN Object

The SLC500_CN object represents the physical Allen Bradley SLC500 processor coupled with the Allen-Bradley SLC500 ControlNet RS-232 Interface module (1747-KFC15).

- 1747-L5xx with 1747-KFC15

This object is hosted by PORT_CN.

To add the SLC500_CN object to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CN_000** object.
2. Select **Add SLC500_CN Connection** from the shortcut menu. The **New_SLC500_CN_000** object is created.
3. Rename the newly created object as appropriate. The **SLC500_CN Parameters** configuration view is displayed.

The configuration view contains four parameters, three of which are configurable:

- **Processor Type:** (not configurable) The information is automatically provided (SLC500).
- **ControlNet Address:** The node address on the ControlNet Network (Octal).
    - The valid range is 1 - 99 decimal.
    - The default value is 1 (one).
- **Reply Timeout:** Enter the maximum amount of time (in seconds) that the Communication Driver will wait for a response from the controller.
    - The valid range is 1 - 300 seconds.
    - The default value is 15 seconds.
- **Max CIP Connections:** The maximum number of CIP connections which can be originated from the Communication Driver to this device.
    - The valid range is 1 - 31.

- The default value is 4 (four).

Note: For each CIP connection established, the PLC allocates certain resources to handle the connection. If too many CIP connections are established by the Communication Driver, the PLC processor allocates more resources for the CIP, and other operations will suffer. For example, with 31 connections, the data updates become extremely sluggish since the PLC ran out of resources to handle the runtime logic and updating the control items.

CNB_PORT_CLX Object

The CNB_Port_CLX object is a logical representation of the ControlNet port for the Allen-Bradley ControlNet bridge module.

This object is hosted by PORT_CN.

To add the CNB_PORT_CLX object to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CN_000 object**.
2. Select **Add CNB_PORT_CLX Connection** from the shortcut menu. The **New_CNB_PORT_CLX_000** object is created.
3. Rename the newly created object as appropriate. The **CNB_PORT_CLX Parameters** configuration view is displayed.

This configuration view has four parameters, three of which are configurable:

- **ModuleType:** (not configurable) The information is provided automatically by the OI Server Manager (ControlNet).
- **Slot Number:** A sequential number beginning with 0 (zero) assigned to each slot in the ControlLogix ControlNet Bridge module.
  - The slot number indicates where the sub-module resides.
  - The valid range is 0 - 16.
  - The default value is 0 (zero).
- **Channel Number:** The number of physical channels/ports used on the ControlLogix ControlNet interface module.
  - Select Channel A or Channel B.
- **ControlNet Address:** The node address on the ControlNet network.
  - The valid range is 1 - 99 decimal.
  - The default value is 1 (one).

CNB_PORT_FLX Object

The CNB_Port_FLX object represents the physical ControlNet port for the Allen-Bradley FlexLogix ControlNet Communication Daughter Card.

This object is hosted by PORT_CN.

To add the CNB_PORT_FLX object to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CN_000** object.

2. Select **Add CNB_PORT_FLX Connection** from the shortcut menu. The **New_CNB_PORT_FLX_000** object is created.
3. Rename the newly created object as appropriate. The **CNB_PORT_FLX Parameters** configuration view is displayed.

This configuration view has four parameters, three of which are configurable:

- **Module Type:** (not configurable) Information provided automatically by the OI Server Manager (ControlNet).
- **Slot Number:** A sequential number beginning with 0 (zero) assigned to each slot in the ControlLogix ControlNet Bridge module.
  - The slot number indicates where the sub-module resides.
  - The valid range is 0 - 16.
  - The default value is 0.
- **Channel Number:** The number of physical channels/ports used on the ControlLogix ControlNet interface module.
  - Select Channel A or Channel B.
- **ControlNet Address:** The node address on the ControlNet network.
  - The valid range is 1 - 99 decimal.
  - The default value is 1 (one).

CNB_PORT_CPLX Object

The CNB_Port_CPLX object represents the physical ControlNet port for the Allen-Bradley CompactLogix ControlNet processor module.

This object is hosted by PORT_CN.

To add the CNB_PORT_CPLX object to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_CN_000** object.
2. Select **Add CNB_PORT_CPLX Connection** from the shortcut menu. The **New_CNB_PORT_CPLX_000** object is created.
3. Rename the newly created object as appropriate. The **CNB_PORT_CPLX Parameters** configuration view is displayed.

This configuration view has four parameters, three of which are configurable:

- **Module Type:** (not configurable) The information is provided automatically by the OI Server Manager (ControlNet Communication).
- **Slot Number:** A sequential number beginning with 0 (zero) assigned to each slot in the ControlLogix ControlNet Bridge module.
  - The slot number indicates where the sub-module resides.
  - The valid range is 0 - 16.
  - The default value is 0 (zero).
- **Channel Number:** The number of physical channels/ports used on the ControlLogix ControlNet interface module.

  - Select Channel A or Channel B.
- **ControlNet Address:** The node address on the ControlNet network.
  - The valid range is 1 - 99 decimal.
  - The default value is 1 (one).

The DeviceNet Network

The following Allen-Bradley controllers can be configured to communicate with I/O data from the DeviceNet devices for the Communication Driver to access using the following methods:

- ControlLogix controller by means of its DeviceNet Bridge module.
- FlexLogix controller by means of its DeviceNet daughter-card.
- CompactLogix controller by means of its DeviceNet scanner.
- PLC-5 controller by means of its DeviceNet scanner.
- SLC500 controller by means of its DeviceNet scanner.
- MicroLogix controller by means of its DeviceNet scanner.

Important: The DeviceNet connectivity is achieved with the DeviceNet scanner attached to the corresponding controller. The ABCIP Communication Driver does not internally implement the DeviceNet protocol.

The Data Highway Plus Network

Routing through the DHRIO_CLX object, the ABCIP Communication Driver accesses data from the PLC-5 and SLC500 processors on the Data Highway Plus network, as well as the MicroLogix processors on the DH485 network via the DH+/DH485 Bridge module (1785-KA5).

DHRIO_CLX Object

The DHRIO_CLX object represents the physical Allen-Bradley ControlLogix DH+/RIO Communication Interface module.

  - 1756-DHRIO

This object is hosted by BACKPLANE_CLX.

To add the DHRIO_CLX object to your ABCIP hierarchy

1. Select and right-click on the **New_BACKPLANE_CLX_000** object.
2. Select **Add DHRIO_CLX Connection** from the shortcut menu. The **New_DHRIO_CLX_000** object is created.
3. Rename the newly created object as appropriate. The **DHRIO_CLX Parameters** configuration view is displayed.

This configuration view has three parameters, two of which are configurable:

- **Module Type:** (not configurable) Information provided automatically by the OI Server Manager (DH+/RIO Communication).
- **Slot Number:** A sequential number beginning with 0 (zero) assigned to each slot in a ControlLogix DH+/RIO Bridge module.
  - The slot number indicates where the sub-module resides.

  - The valid range is 0 - 16.
  - The default value is 0 (zero).
- **Max CIP Connections per Channel:** The maximum number of CIP connections allowed per channel.
  - The valid range is 1- 31.
  - The default value is 4 (four).

  Note: For each CIP connection established, the PLC allocates certain resources to handle the connection. If too many CIP connections are established by the Communication Driver, the PLC processor allocates more resources for the CIP, and other operations will suffer. For example, with 31 connections, the data updates become extremely sluggish since the PLC ran out of resources to handle the runtime logic and updating the control items.

PORT_DHP Object

The PORT_DHP object represents the physical DH+ port for the Allen-Bradley DH+/RIO Communication Interface e module.

This object is hosted by DHRIO_CLX.

To add the PORT_DHP object to your ABCIP hierarchy

1. Select and right-click on the **New_DHRIO_CLX_000** object.
2. Select **Add PORT_DHP Connection** from the shortcut menu. The **New_PORT_DHP_000** object is created.
3. Rename the newly created object as appropriate. The **PORT_DHP Parameters** configuration view is displayed.
4. This configuration view has four parameters, three of which are configurable:

- **Port Type:** (not configurable) Information provided automatically by the OI Server Manager (DH Plus).
- **Channel Number:** The number of physical channels/ports used on the ControlLogix DH+/RIO Bridge module. Select Channel A or Channel B.
- **DH Plus Node Address:** The node address on the DH+ network (Octal).
  - The valid range is 0 - 77 octal.
  - The default value is 1 (one) octal.
- **DH Plus Link ID:** The DH+ link ID of the channel.
  - The link ID is defined in the DHRIO routing table for the channel.
  - The valid range is 1 - 199.
  - The default value is 1 (one).

SLC500_DHP Object

The SLC500_DHP object represents the physical Allen-Bradley SLC500 processor on the Data Highway Plus network.

  - 1747-L54x

This object is hosted by PORT_DHP.

To add the SLC500_DHP object to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_DHP_000** object.
2. Select **Add SLC500_DHP Object** from the shortcut menu. The **New_SLC500_DHP_000** object is created.
3. Rename the newly created object as appropriate. The **SLC500_DHP Parameters** configuration view is displayed.

The configuration view contains three parameters, two of which are configurable:

- **Processor Type:** (not configurable) The information is automatically provided (SLC500).
- **DH Plus Node Address:** The node address on the DH+ network (Octal). Select the DH+ node number from the drop-down box.
  - The valid range is 0 - 77 octal.
  - The default value is 0 (zero) octal.
- **Reply Timeout:** Enter the maximum amount of time (in seconds) that the Communication Driver will wait for a response from the controller.
  - The valid range is 1 - 300 seconds.
  - The default value is 15 seconds.

M1785KA5_GWY Object

The M1785KA5_GWY object represents the physical Allen-Bradley DH+/DH485 Gateway (1785-KA5) Communication module.

- 1785-KA5

This object is hosted by PORT_DHP.

To add the M1785KA5_GWY object to your ABCIP hierarchy

1. Select and right-click on the **New_PORT_DHP_000** object.
2. Select **Add M1785KA5_GWY Connection** from the shortcut menu. The N**ew_M1785KA5_GWY_000** object is created.
3. Rename the newly created object as appropriate. The **M1785KA5_GWY Parameters** configuration view is displayed.

There are three parameters in this configuration view, two of which are configurable:

- **Module Type:** (not configurable) The information is automatically provided (1785-KA5).
- **DH-485 Node Address:** The node address on the DH-485 network.
  - Select the DH-485 node number from the drop-down box.
  - The valid range is 1 - 31.
  - The default value is 1 (one).
- **DH-485 Link ID:** The DH-485 link ID of the module.
  - This link ID has to match what has been defined in the DHRIO Routing table for the DH+ Bridge.
  - The valid range is 1 - 199.

- The default value is 1 (one).

ML_DH485 Object

The ML_DH485 object represents the physical Allen-Bradley MicroLogix processor coupled with the Allen-Bradley Advanced Interface Converter for DH485 (1761-NET-AIC).

- 176x-Lxxx with 1761-NET-AIC

This object is hosted by M1785KA5_GWY.

To add the ML_DH485 object to your ABCIP hierarchy

1. Select and right-click on the **New_M1785KA5_GWY_000** branch.
2. Select **Add ML_DH485 Connection** from the shortcut menu. The **New_ML_DH485_000** object is created.
3. Rename the newly created object as appropriate. The **ML_DH485 Parameters** configuration view is displayed.

The configuration view contains three parameters, two of which are configurable:

- **Processor Type:** (not configurable) The information is automatically provided (MicroLogix).
- **DH485 Node Address:** The node address on the DH485 network.
  - The valid range is 0 - 31.
  - The default value is 1 (one).
- **Reply Timeout:** The maximum amount of time (in seconds) that the Communication Driver will wait for a response from the controller.
  - The valid range is 1 - 300 seconds.
  - The default value is 15 seconds.

SLC500_DH485 Object

The SLC500_DH485 object represents the physical Allen-Bradley SLC500 processor coupled with the Allen-Bradley Advanced Interface Converter for DH485 (1761-NET-AIC).

- 1747-L5xx

This object is hosted by M1785KA5_GWY.

To add the SLC500_DH485 object to your ABCIP hierarchy

1. Select and right-click on the **New_M1785KA5_GWY_000** branch.
2. Select **Add SLC500_DH485 Connection** from the shortcut menu. The **New_SLC500_DH485_000** object is created.
3. Rename the newly created object as appropriate. The **SLC500_DH485 Parameters** configuration view is displayed.

The configuration view contains three parameters, two of which are configurable:

- **Processor Type:** (not configurable) The information is automatically provided (SLC500).
- **DH485 Node Address:** The node address on the DH485 network.
  - Valid range is 0 - 31.

    - The default value is 1 (one).

- **Reply Timeout:** The maximum amount of time (in seconds) that the Communication Driver will wait for a response from the controller.
    - The valid range is 1 - 300 seconds.
    - The default value is 15 seconds.

Device Groups and Device Items

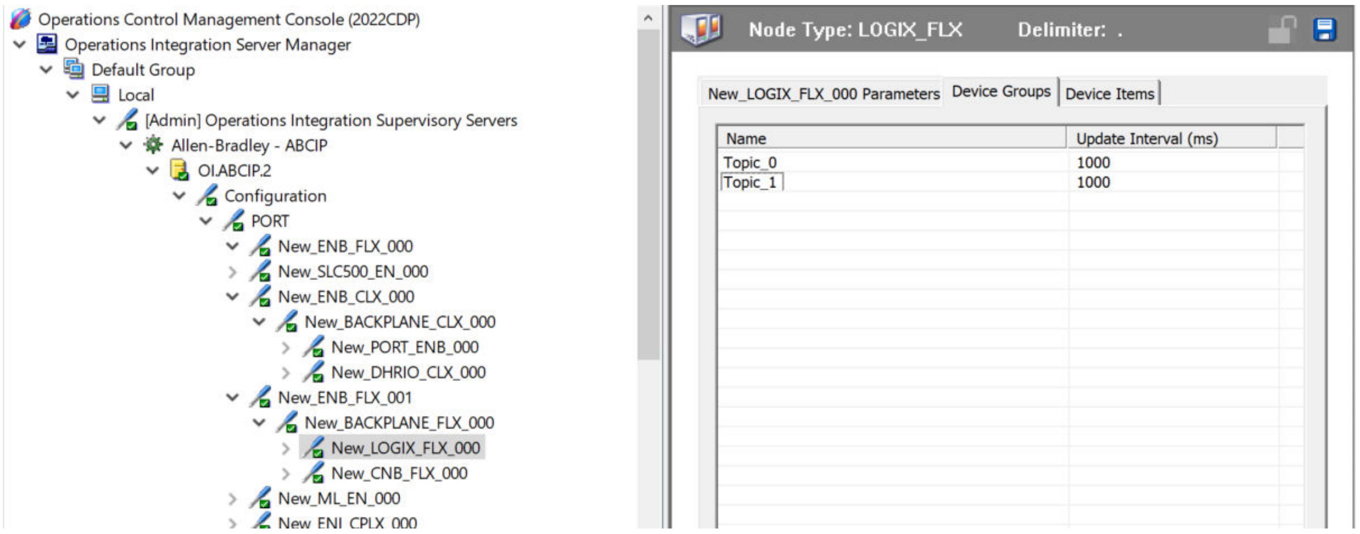
The Device Group and Device Item tabs in the OI Server Manager user interface are used to create new, modify, or delete device group and item definitions for an object.

For DDE/SuiteLink communications, one or more device group definitions must exist for each controller that the Communication Driver will communicate with. Each device group (topic) definition should contain a unique name for the controller associated with it.

Device Group Definitions

The Device Groups dialog box is displayed by clicking the **Device Groups** tab in the CIP, LOGIX5000_CLX, LOGIX_FLX, ML_EN, LOGIX_CPLX, SLC500_EN, PLC5_CN, SLC500_CN, PLC5_DHP, SLC500_DHP, ML_DH485, SLC500_DH485 node configuration view. The **Device Groups** dialog box allows you to add, define, and delete device groups, in addition to configuring default update intervals and editing update intervals for the objects.

Note: When you add a new device group, enter a unique name. When you select another part of the Communication Driver tree hierarchy, you are prompted to save the modifications to the configuration set.



Name	Update Interval (ms)
Topic_0	1000
Topic_1	1000

To create or add device groups

1. Right-click in the Device Groups box.
2. Select the **Add** command from the shortcut menu.
    - When adding a new device group, enter a unique name (up to 32 characters long).
3. Click the **Save** icon (the floppy disk icon in the upper right corner).

To make edits on device groups

1. In the **Name** column, double-click on the device group's name to be edited and make the change.
2. In the **Update Interval** column, double-click on the device group's value to be edited and make the change.
3. To enable unsolicited messages, right-click on the device group name, and select **Edit** to display Device Group Parameters. Select **Support Unsolicited Messages** and click **OK**.
4. Click the **Save** icon (the floppy disk icon in the upper right corner).

To delete device groups

1. Right-click on the device group to be deleted.
2. Select the **Delete** command from the shortcut menu.
   - The OI Server Manager confirmation box is displayed.
3. Click **Yes** to proceed with the deletion.
4. Click the **Save** icon (the floppy disk icon in the upper right corner).

To configure default update intervals

1. Right-click in the Device Groups box.
2. Select **Config Default Update Interval** from the shortcut menu.
3. Click the **Save** icon (the floppy disk icon in the upper right corner).

To edit update intervals

- Double-click its value in the Update Interval column and make the edit.
  - Update Interval is the frequency (in milliseconds) that the Communication Driver acquires data from the topics associated with that device group.
  - Different topics can be polled at different rates in a controller by defining multiple device group names for the same controller and setting a different Update Interval for each device group.

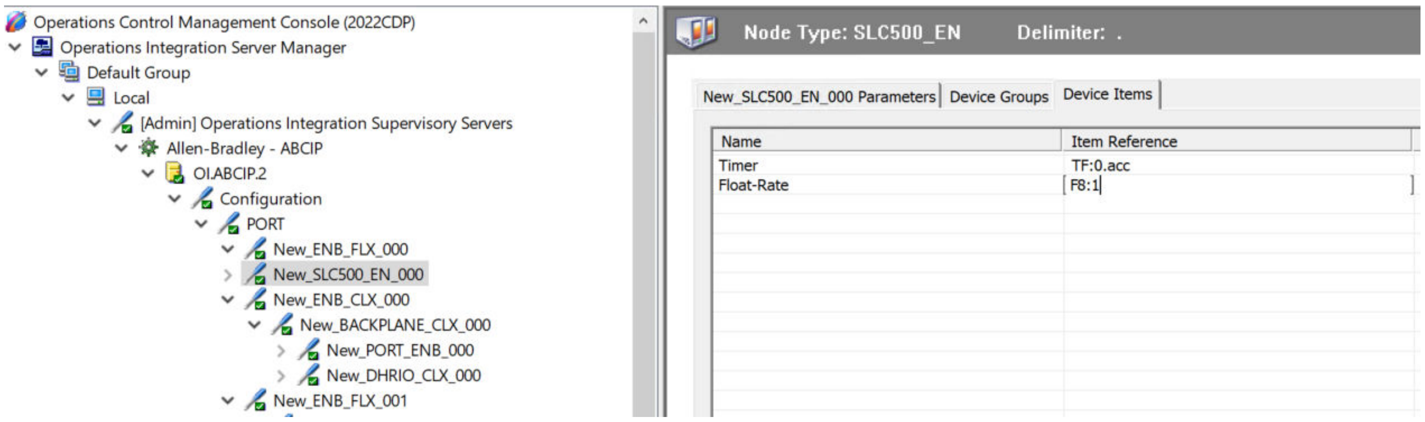
Device Item Definitions

To make it easier to remember lengthy or strictly structured item names, the Communication Driver enables you to create aliases for these item names. For example, it may be easier for you to remember the item syntax "T4:1.acc" as "Timer1."

The Device Items tab in the OI Server Manager user interface is used to create new, modify, delete, export, or import device item definitions for an object. The configuration is performed in the Device Items dialog box, which you can open by clicking the **Device Items** tab in the LOGIX5000_CLX, LOGIX_FLX, ML_EN, LOGIX_CPLX, SLC500_EN, PLC5_CN, SLC500_CN, PLC5_DHP, SLC500_DHP, ML_DH485 or SLC500_DH485 node configuration view.

Once the Device Items feature is used to configure item names, it provides the Communication Driver with the capability to perform OPC Item browsing. When the Communication Driver is running and an OPC client requests item information, the configured items will show up under the controller hierarchy node.

Note: Device items have the precedence in addressing items in the controller device at run time. Items request from the client would be searched from the Device Items Name list first before going out to the controller.



Name	Item Reference
Timer	TF:0.acc
Float-Rate	F8:1

To create or add device items

1. Right-click in the Device Items box.
2. Select the **Add** command from the shortcut menu.
3. Type the item name (symbolic name) of your choice in the Name column.
   - The device item name must be unique and is limited to 32 characters long.
4. Double-click the line on the Item Reference column and enter the correlated item reference (the actual I/O item name in the device) for the device item name you have just selected.
   - For example, "n7:0."
5. Click the **Save** icon (the floppy disk icon in the upper right corner).

Note: System items are not valid item references, but Communication Driver-specific system items are valid.

To rename device items

1. Right-click on the device item to be renamed.
2. Select **Rename** from the shortcut menu, then make the change.
3. Click the **Save** icon (the floppy disk icon in the upper right corner).

To delete device items

1. Right-click on the device item to be deleted from the list.
2. Select the **Delete** command from the shortcut menu.
3. Click the **Save** icon (the floppy disk icon in the upper right corner).

To clear all device items

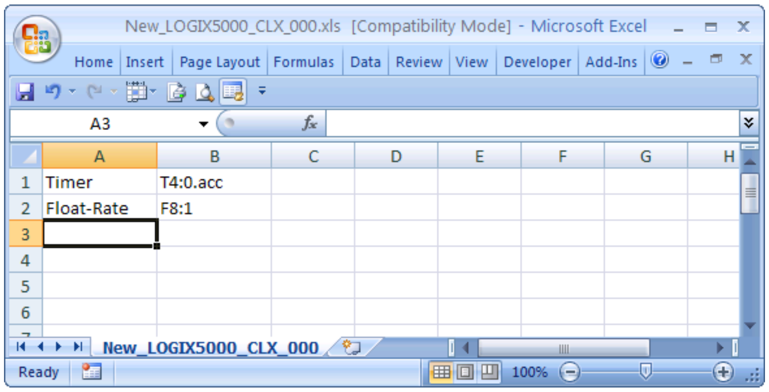
1. Right-click in the Device Items box.
2. Select the **Clear All** command from the shortcut menu.
   - The OI Server Manager confirmation box appears.
3. Click **Yes** to confirm the deletion.
   - All the device items listed will be cleared.

Exporting and Importing Communication Driver Item Data

The Export and Import commands on the shortcut menu enable you to export and import the Communication Driver item data to and from a CSV file, after the configuration of the Device Items has been completed. These commands will allow you to perform an off-line, large-scale edit on the item data configured for a controller, and import what has been edited back into the controller configuration.

To export Communication Driver item data to a CSV file

1. Right-click in the Device Items box.
2. Select the **Export** command from the shortcut menu.
   - The Save As dialog box appears.
   - The file name has defaulted into "PLCHierarchyNodeName.csv," within the current-system-configured default directory.
3. Accept the defaults to save the file or rename the file if appropriate.
   - The file is saved as New_PLC5_DHP_000.cs0v.
   - It is editable in Microsoft Excel.



	A	B	C	D	E	F	G	H
1	Timer	T4:0.acc						
2	Float-Rate	F8:1						
3								
4								
5								
6								

The file can now be edited off-line. It contains one row for each item configured with two columns, Name and Item Reference, respectively.

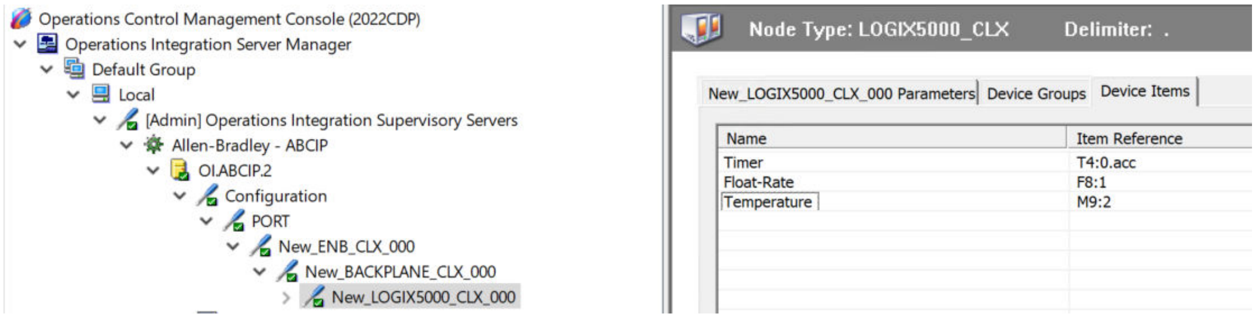
To import Communication Driver item data from a CSV file

1. Right-click in the Device Items box.
2. Clear all the item data you wish to replace with the edited.csv file by selecting the **Clear All** command.
   - The OI Server confirmation box is displayed.
3. Click **Yes** to confirm the deletion.
   - The data will be cleared.
4. Select the **Import command** from the shortcut menu.
   - The Open dialog box appears.
   - It defaults to the .csv file extension within the current-system-configured default directory.
5. Browse and select the specific CSV file you want to import, select it, then click **OK** for confirmation.

   The OI Server Manager will import the file and deposit it in the **Device Items** box.

During the imported file processing:

- New item references will be added based on unique names.
- If there are duplicate names, you will be provided with the ability to replace the existing entry with the new entry, or ignore the new entry.



Name	Item Reference
Timer	T4:0.acc
Float-Rate	F8:1
Temperature	M9:2

When the Communication Driver is running and an OPC client requests item information, the imported configured items will show up under the controller hierarchy node.

Note: When you select another part of the Communication Driver tree hierarchy, you are prompted to save the modifications to the configuration set.

Each configuration view associated with nodes in the Communication Driver hierarchy tree has a common feature, the Save button located on the upper-right corner of the configuration view.

When you modify any parameters in the Device Groups dialog box, click Save to implement the new modifications.

If you do not click Save, you will be prompted to save the new data to the configuration set.

Scan-Based Message Handling

The Communication Drivers are based on the concept of polling a hardware device for information. This polling is driven by a need that is expressed in the form of requests from one or more clients. After a particular piece of information has been requested by a client, the Communication Driver formulates its own request and sends that request to the hardware device. The Communication Driver then waits for a response to its request. After the information has been received, the Communication Driver passes that information back to the client, and repeats the process until all clients have ceased requesting information.

The rate at which the Communication Driver will poll a particular device for a specific piece of information is defined in the device group (topic definition) inside the Communication Driver, using a parameter called the Update Interval. When setting this parameter, there is always a trade-off between the update speed of the device group and the resulting decrease in system responsiveness.

Because you more than likely want a very fast response, the temptation is to set the Update Interval to a value close to 0 seconds. However, if every point is polled at this rate, the entire system will suffer due to slow response time. Therefore, you should compromise, and set the Update Interval to a more reasonable value. You could also create multiple device groups for each device, setting the Update Interval to different values, then assigning different items to different device groups depending on how quickly the values change and how quickly you want to see an update of those changes.

Some items, like alarms, change very infrequently but because of their importance require very fast updates. For those kinds of items, you should set the Update Interval at a very small value. If you desire an immediate response, set the Update Interval at 1.

Unsolicited Message Handling

In the world of controllers and Communication Drivers, it is obvious that a controller will know when a critical event has occurred before the Communication Driver will have a chance to poll for that data. Therefore, it would seem natural that if a critical event occurs, the controller should have the capability to inform the Communication Driver immediately, without having to wait for the Communication Driver to poll it.

This is the role of an unsolicited message. After a controller has determined that a critical condition exists, it can generate a message immediately sent to the Communication Driver without a prior request from the Communication Driver. The unsolicited message implementation requires both the messaging instructions properly programmed in the controller logic and the device group appropriately configured in the Communication Driver.

Note: The use of unsolicited messages requires configuration changes in the controller. Please refer to the related Rockwell Automation documentation for procedures to set up unsolicited messages from the supported controller processors.

The ABCIP Communication Driver supports unsolicited messages from the following processors:

- PLC-5 and SLC 5/04 processors on the Data Highway Plus network.

  The following non-Logix processor configuration does not support unsolicited message handling:
  MicroLogix with 1785-KA5 via ControlLogix Gateway (1756-DHRIO and 1756-ENB).

- Logix5000 and MicroLogix 1100 processors on the Ethernet network.

  The following Logix5000 processor configuration does not support unsolicited message handling:
  CompactLogix going through the EtherNet/IP interface module (1761-NET-ENI).

- Logix5000, PLC-5, and SLC500 processors on the ControlNet network.

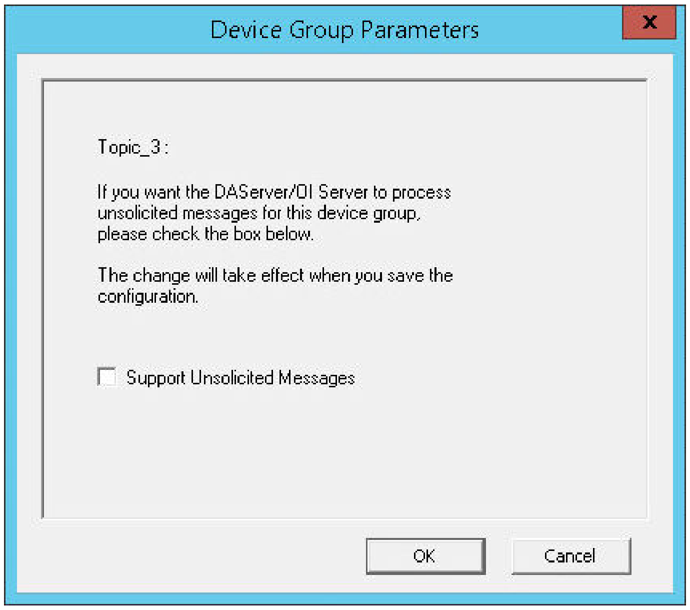
  The following SLC500 processor configuration does not support unsolicited message handling:
  SLC500 using 1747-KFC15 interface on ControlNet via the ControlLogix Gateway (1756-CNB and 1756-ENB) to RSLinx on Ethernet.

To configure the ABCIP Communication Driver to receive unsolicited messages

This option is available only to the controllers listed above.

1. Click on the target controller node under the controller branch of the Communication Driver hierarchy.
2. Select the **Device Group** tab of the configuration view.
3. Add a new device group or select an existing device group.
4. Right-click on the device group name, then select **Edit** from the shortcut menu.

   The Device Group Parameters dialog box is displayed.



Topic_3 :

If you want the DAServer/OI Server to process unsolicited messages for this device group, please check the box below.

The change will take effect when you save the configuration.

☐ Support Unsolicited Messages

5. In the **Device Group Parameters** dialog box, select the **Support Unsolicited Messages** check box.
6. Click **OK**.

Note: Because the status of Support Unsolicited Messages check box cannot be readily viewed from the Device Groups tab, proper naming of device groups that support unsolicited messages is strongly recommended.

7. If appropriate, you can modify the Update Interval to "0".
8. Click **Save** to save the configuration change.

Note: To enhance performance in message handling, the default for the device group is to not provide unsolicited message data; therefore, the Support Unsolicited Messages check box is not checked. The setting of this check box is hot-configurable. Unsolicited message handling will take effect in the Communication Driver as soon as the change made in the configuration view is saved.

Target-specific Unsolicited Messages

The Communication Driver supports target-specific unsolicited messages. This method involves sending messages to its target as specified by a given static IP address.

- Configure the appropriate message instructions in the controller with the proper path (including the destination IP address) for sending unsolicited messages.
- Configure the computer, in which the Communication Driver resides to receive unsolicited messages from the controller, with the corresponding IP address.
- Two instances of target-specific unsolicited messages are generated by the Communication Driver:
  - If the value of "Update Interval" for a topic is 0 (zero), the server will poll this topic only one time at the start. After that, only an unsolicited message will update the data.
  - If the value of "Update Interval" for a topic is >0 (greater than zero), the server will update the data for a particular item immediately upon receiving an unsolicited message for the item. The Communication Driver will also update the data at every Update Interval.

Note: The Communication Driver requires unsolicited messages to be configured as "Connected" in Logix5000 processors. For details in setting the Allen-Bradley hardware for unsolicited messaging, please refer to the related Rockwell Automation documentation.

To Receive Unsolicited Messages

1. Activate the Communication Driver.
2. Add the items, defined in the controller for unsolicited messages, for updates under the device group set up for receiving unsolicited messages.

To Access the Settings for Device groups

1. Click on the target controller node under the Configuration hierarchy of your Communication Driver.
2. Select the Device Groups tab of the configuration view pane at right.
3. Right-click on the device group name, then select **Edit** from the shortcut menu.

Note: Unsolicited PLC-5 Typed Write using CIP with Source ID method from the Logix5000 processor is not supported. Instead, configure the message instruction with CIP Data Table Write using the CIP method from the Logix5000 processor.

Chapter 3

Using Auto-Build with the ABCIP Communication Driver

About Auto-Build and the ABCIP Communication Driver

Auto-Build is an Engineering Efficiency feature that allows you to read the templates and instances in the controllers capable of running the Auto-Build feature. You can also replicate these structures to Application Server using a set of simple steps, that allows a one-to-one correspondence between the PLC and a Galaxy project.

This feature is accessible in the **Auto-Build** tab of the appropriate Logix controller hierarchy within the OI Server Manager.

Auto-Build browsing can be accomplished online if connected to the PLC, or offline if the PLC file is available. The Auto-Build feature supports only user-defined data types. It does not support other data types.

Prerequisites for Auto-Build Operation

Ensure the following prerequisites are met before configuring the Auto-Build.

Operating System Requirements

For a list of supported operating systems for the Communication Driver, refer to the **Technology Matrix**, available at the **Global Customer Support** (GCS) Site:

The **Technology Matrix** is a searchable database that contains the latest product information. Enter the product name in the search bar, then select the release to view:

- **Product Information:** version name, number, release date, etc.

- **Product Notes:** key release information, new features, and updates
- **OS Compatibility:** list of compatible Windows and Windows Server versions
- **Browser Compatibility:** list of compatible browsers
- **Virtualization Compatibility:** list of compatible virtualization software products and versions
- **Product Coexistence:** list of products that can be installed on the same computer
- **Product Interoperability/Compatibility:** list of products that can operate together and communicate with each other through a common message protocol.

Software Requirements

While most of the prerequisite software required for Auto-Build are installed during the Communication Drivers Pack installation, others need to be manually downloaded as listed below.

Software	Component	Installation Instruction
.NET Framework	.NET 4.0	Installed with Communication Drivers Pack as a prerequisite
	.NET 4.5+	• Pre-installed with the OS - Windows 2016, Windows 2012 R2, Windows 10 and Windows 8.1 • Requires manual download for Windows 2012, Windows 2008 R2, Windows 8 and Windows 7
Microsoft C Runtime Libraries	VC 2015-2019	Installed with Communication Drivers Pack as a prerequisite
XML	XML 6.0	Installed with Communication Drivers Pack as a prerequisite
ArchestrA Data Store	ADS 2.0.4	Installed with Communication Drivers Pack as a prerequisite
AdminUser	64-Bit AdminUser	Installed with Communication Drivers Pack as a prerequisite, on 64-bit OS only

Browser Requirements

Auto-Build requires Microsoft Edge browser for functioning.

Installation Requirements:

Auto-Build must be installed in the same node as an Application Server Galaxy Repository. You can only view the Auto-Build tab from the Application Server node. You cannot view the Auto-Build tab from a remote node.

Licensing Requirements

Auto-Build requires a Professional Communication Driver license or higher. Refer to the Centralized (Activation-Based) Licensing section of the Communication Drivers Pack Help for more details.

If the license server is not configured, a warning message is displayed:

Unable to obtain a license for Auto-Build

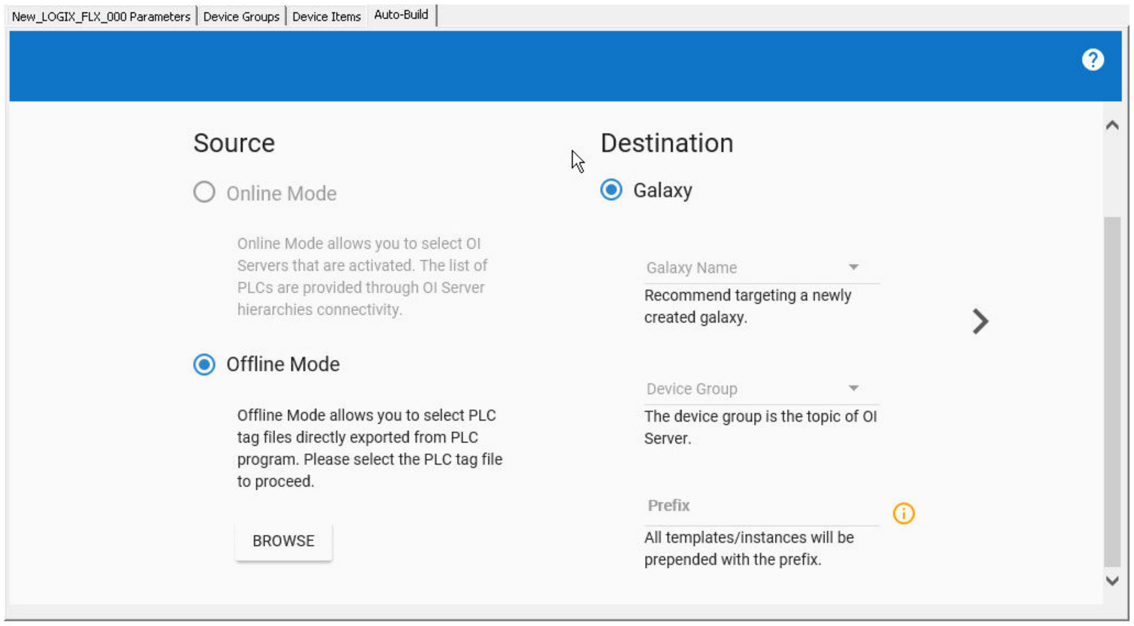
Note: A standard license, or no license will allow the user to exercise Auto-Build configuration steps, but not building the objects in Application Server.

Required User Privileges

The user who launches the OCMC must be part of the oiAdministrators Windows Local Users and Groups to access the Auto-Build functionality, else an error message is displayed saying the user or the user group is not part of oiAdministrators group. If the user is not a part of the oiAdministrators group, you must add the user to the oiAdministrators group manually and re-login your computer. The user who installs the Communication Drivers Pack will be added to the oiAdministrators group automatically.

Accessing the Auto-Build screen

Click the **Auto-Build** tab in the logix node configuration view to view the Auto-Build screen. Select the server-specific PLC Tag File (.L5X for ABCIP) to initiate the build operation.



Online Mode allows you to select OI Servers that are activated. The list of PLCs are provided through OI Server hierarchies connectivity.

Offline Mode allows you to select PLC tag files directly exported from PLC program. Please select the PLC tag file to proceed.

Recommend targeting a newly created galaxy.

The device group is the topic of OI Server.

All templates/instances will be prepended with the prefix.

Refer to the section "Using Auto-Build" in the Communication Drivers Pack Help for more details.

Note: If you make any changes to the configuration of the System Management Server (SMS) during run time, you must restart the **AVEVA Communications Backend Service** in the **Services** console of your machine. In the **Advanced Configuration** of the System Management Server (SMS), if you change the **HTTP Port** or the **HTTPS Port** then you must run the below command as an administrator in the command prompt:

If you change the HTTP Port field

```
netsh http add urlacl url=http://localhost:<https port configured in SMS>/oi/ user="NT Authority\Network Service"
```

If you change the HTTPS Port field

```
netsh http add urlacl url=https://localhost:<https port configured in SMS>/oi/ user="NT Authority\Network Service"
```

If you do not run the above commands after changing the ports and try to access the Auto-Build functionality, you will get a HTTP 503 error saying the service is unavailable.

Monitoring Auto-Build Progress

The progress bar displays the extent of completion of the building operation.

Open the Log Viewer within the OCMC to view the logs recorded during the upload.

Stopping the Auto Build Operation

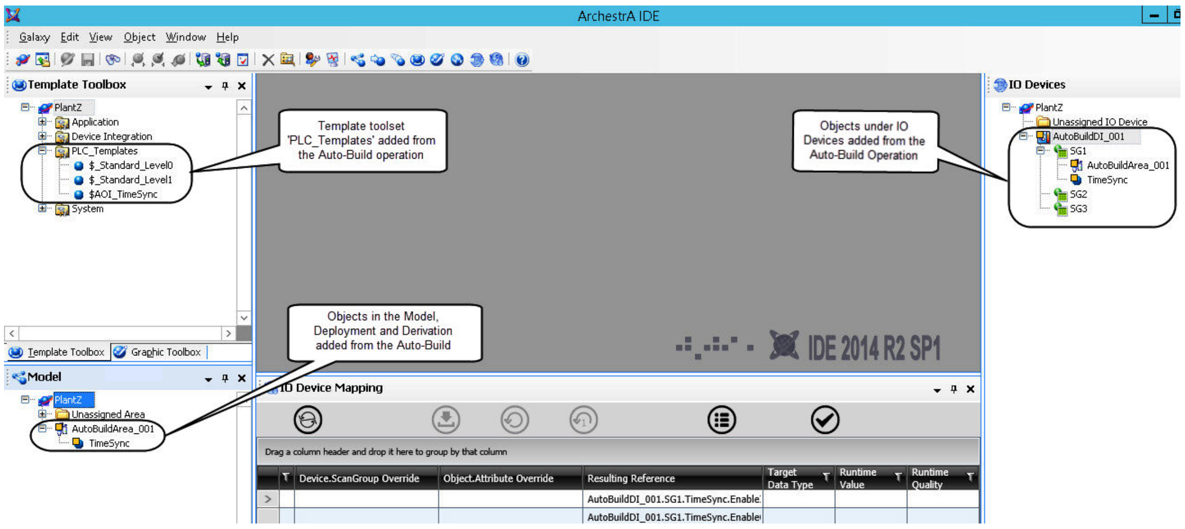
1. Click the **Stop** button during the progress to terminate the Auto Build operation.
2. The progress displays the terminating process.

The galaxy templates and instances created prior to the stop request will continue to remain in the Galaxy.

Note: Click anywhere on the screen during the building or build termination stages to minimize the progress to the title bar. Click the title bar again to bring back the progress bar.

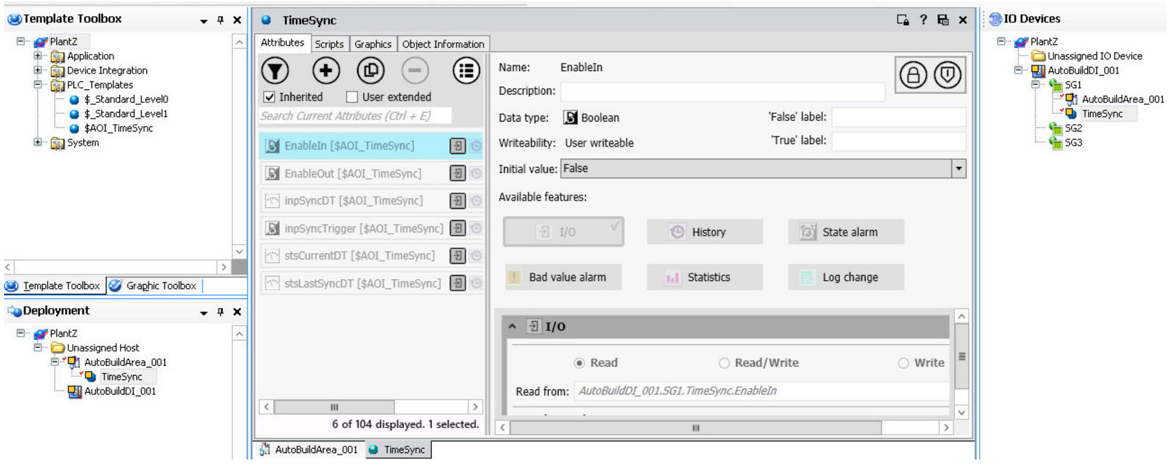
Template Generation in the Application Server

The instances submitted from Auto-Build are generated in the selected Galaxy within the IDE.



Device.ScanGroup Override	Object.Attribute Override	Resulting Reference	Target Data Type	Runtime Value	Runtime Quality
		AutoBuildDI_001.SG1.TimeSync.Enable			
		AutoBuildDI_001.SG1.TimeSync.Enable			

The attributes of the selected instance are displayed in the center of the IDE screen as shown below.



Name: EnableIn
Description:
Data type: Boolean
Writeability: User writeable
'False' label:
'True' label:
Initial value: False
Read from: AutoBuildDI_001.SG1.TimeSync.EnableIn

PLC Tag Database Feature Support

Allen-Bradley Logix TagDB Features Support Status

Category	**TagDB Item**	**Logix Program Feature**	**Auto-Build Feature**	**Online Mode**	**Offline Mode**
Data Type	User-Defined	User Defined (UDT) data structure	Template	Y	Y
Data Type	Add-On-Defined	Add-On-Defined (ADT) data structure	Template	Y	Y
Data Type	Module-Defined	I/O & Communication Module (MDT) data structure	Not Supported	N	N
Data Type	Pre-Defined	System Pre-Defined (PDT) data structure	Template	Y	Y
Scope	Controller	Global tag	Instance / Attribute	Y	Y
Scope	Program	Program tag	Not Supported	N	N
Usage	Local	Local tag for ADT	Not Supported	N*	N
Usage	Input	Parameter for ADT	Attribute (non-specific field)	Y	Y
Usage	Output	Parameter for ADT	Attribute (non-specific field)	Y	Y
Usage	InOut	Parameters for ADT	Not Supported	N	N
Tags	Tag Type	"Base" / "Alias"	Attribute (non-specific field)	Y	Y
Tags	Data Type	Refer to the entries under "Data Type" Category above	n/a	Y	Y
Tags	Dimension	Array dimension of "1" / "2" / "3"	Attribute enumeration	Y	Y
Tags	Description	Text field of 512 characters	Attribute (Description field)	N	Y
Tags	External Access	Tag properties for "Read/Write" / "Read Only" / "None"	Attribute writeability properties	Read/ Write	Y

Tags	Constant	Tag properties of "true" / "false"	Attribute (non-specific field)	Y	Y
Tags	Style	Value display in Binary / Octal / Decimal / Hex / ASCII	Attribute display style	N	N

Note:Auto-Build Online Mode does not hide the Local Tags from displaying, but they are **not** supported for item subscription with OI ABCIP.

ADT InOut parameters and local tags are not shown up in the actual tag list under RSLogix5000 TagDB.

The Auto-Build feature supports only user-defined data types. It does not support other data types.

Auto-Build Data Type Mapping

Logix Program Base Data Type	**Logix Data Type Description**	**AppServer Data Type**
BOOL	Boolean	MxBoolean
REAL	Real as 32-bit IEEE Floating-point number	MxFloat
SINT	Singular Integer as 8-bit signed integer	MxInteger
DINT	Double Integer as 32-bit signed integer	MxInteger
INT	Integer as 16-bit signed integer	MxInteger
LINT	Long Integer as 64-bit signed integer	MxInteger
STRING	String as character bytes	MxString

Chapter 4

ABCIP Communication Driver Reference

OPC Browsing

Two types of OPC browsing, namely off-line OPC browsing and on-line OPC browsing, are supported by the ABCIP Communication Driver.

Note: For tag items defined as array data types in an item addition request, the OPC_E_BADTYPE error is returned when an OPC client does not specify the array data type or the VT_EMPTY data type. The only exception is when an OPC client specifies VT_BSTR as the requested data type for an item that is defined as VT_ARRAY|VT_UI1. In this case, the Communication Driver accepts the item addition and returns the data as VT_BSTR.

Off-line OPC Item Browsing (Static Browsing)

The Communication Driver implements population of the namespace to enable OPC browsing of ControlLogix, CompactLogix, FlexLogix, GuardLogix, PLC-5, SLC500, MicroLogix, and SoftLogix processor items. Browsing can also be performed off-line using the .aacfg file for Device Items created and saved with the controller hierarchy node of the Communication Driver.

OPC browsing on item names is also provided to all controllers by means of importing a comma-separated-value (.csv) file, which provides symbolic names to tag names, into the .aacfg file.

Online OPC Item Browsing (Dynamic Browsing)

The online OPC browsing for ControlLogix, CompactLogix, FlexLogix, GuardLogix, PLC-5, SLC500, MicroLogix, and SoftLogix processor items is implemented by the Communication Driver. Using the information retrieved from the processor's tag database, the ABCIP Communication Driver will dynamically create a configuration hierarchy that allows the OI Engine to browse into it.

When it detects that the processor's tag database has changed while browsing, the ABCIP Communication Driver will update the internal tag database but not the configuration hierarchy until the $SYS$BrowseTags system tag is poked with "1."

Note: The OPC item browsing capability is available on-line only when the ABCIP Communication Driver is connecting to the corresponding processor and its tag database is available for access. Otherwise, only off-line items (system items and saved device items) will be displayed.

Note: By default, dynamic OPC browsing of tags from the Communication Driver is disabled. In order to browse tags online from the Communication Driver, a "1" must be written to the $SYS$BrowseTags system tag associated with the chosen processor hierarchy node. Subsequent OPC item browsing operation on this particular processor should be enabled.

Logix5000 Optimization Mode

Operation of the ABCIP Communication Driver per device will be such that it can operate in the same multi-request service (non-optimized) mode or in optimized mode for any device. If the device and firmware support optimization, the default mode for the device will be with optimization. You will have the capability of disabling optimization even though the device and firmware may support it.

The optimization will require a tag database upload from the Logix5000-series controller (ControlLogix, CompactLogix, FlexLogix, GuardLogix, SoftLogix). The tag database contains the data types and unique references that can be used to reference the physical tags available in the controller. The tag database kept in the controller is versioned. ABCIP Communication Driver provides the option that can be used to periodically probe the controller for any version changes and to obtain the changes in the tag database.

- The ABCIP Communication Driver supports only **No Optimization** and **Optimize for Read** mode on firmware versions 21 and higher. By default **Optimize for Read** is selected.
- The **Optimize for Startup** mode will only be used to access controllers operating on firmware versions 20 and lower.
- The ABCIP Communication Driver will switch to **Optimize for Read** mode from **Optimize for Startup** if firmware version 21 or higher is detected from the controller, even if the optimization setting is set to **Optimize for Startup**. In this instance, the ABCIP Communication Driver will generate a warning line in the logger to alert that the **Optimize for Startup** mode is not supported for firmware version 21 and higher controllers.

Three selectable options are:

1. **No optimization**

   All tags that communicate with the Logix processor will use the tag name. The tag database will be uploaded from the controller to validate the tag names. No optimization will have the fastest startup time, but will have the slowest read performance. It will create more messages for controller communication than the other two options. The length of the tag name will affect the number of messages created.

2. **Optimize for Read**

   All tags that communicate with the Logix processor will require a tag database to be available as a prerequisite. This operation also generates a memory buffer inside the controller and thus requires the longest startup time among the three options.

Despite the longer startup time, Optimize for Read provides the fastest read performance after the tag database upload operation has been completed. It will create fewer messages for controller communication.

3. **Optimize for startup time**

   Available only for firmware versions 20 and lower, this option provides the best overall performance among all three optimization options. All tags that communicate with the Logix processor will require a tag database to be available as a prerequisite.

   This option does not generate a memory buffer inside the controller and thus provides a faster startup time than the Optimize for read option. All tags communicating with the Logix processor will be using the physical tag address. It provides a faster read performance than the No Optimization option as multiple tags can be referenced in one request packet to the Logix processor. It will create a higher number of messages for controller communication than the Optimize for read option.

   Note: If this option is checked, the 'Auto Synchronize Tag' option is checked automatically and cannot be unchecked.

UDT Optimization

A UDT (User-Defined Type) is a data type defined by the user in the Logix5000 processor. A UDT can group various data types, such as integers, floats, and so on, into a single structure. When this feature is enabled, the Communication Driver will attempt to group requests for a UDT's elements into a request for the whole structure. In fact, this feature also works for system predefined structure.

If the size of the structure exceeds 488 bytes, the Communication Driver will send separate requests for each structure's element. If the UDT involved is a nested structure (a UDT containing other UDTs), the Communication Driver will determine the optimal UDT to retrieve.

Note: Optimization and UDT Optimization features are selectable from all Logix5000-series controllers.

UDT Optimization with None Access Attribute

Starting with ControlLogix firmware version 18.x, using the Rockwell RSLogix 5000 Programming Software, UDT tags, and their elements can be configured with an External Access property setting of Read/Write, ReadOnly or None. The None setting is specifically meant to define a private tag within the processor, which is not exposed to components outside of the controller, such as the ABCIP Communication Driver. This affects the UDT optimization capability in the ABCIP Communication Driver. This also affects Add On Instructions behavior, which makes extensive use of UDTs. For these reasons, UDTs with elements having the External Access property set to None is not supported. UDTs must not contain any elements with External Access property set to None when the UDT optimization option is checked in the ABCIP Communication Driver.

Important: You must reset, or deactivate and reactivate, the Communication Driver if you change the access rights of an element in a UDT from None to ReadOnly or Read/Write.

Logix5000 Write Optimization

The Poke Mode parameter in the configuration screen of the ABCIP Communication Driver (OI.ABCIP.x) in the OCMC controls how the Communication Driver treats pokes within a transaction with respect to optimization and folding.

You can select one of three modes:

- Control Mode

- Transition Mode
- Optimization Mode (ABCIP Default)

  Control Mode - preserves the poke order without folding. Typically used by batch and control applications that depend on the order of the pokes and processing every item poked.

  Transition Mode - preserves the poke order with minimum folding by keeping the first, second, and last poke values of an item. Typically used by batch and control applications that depend on the order of pokes but not processing every item poked.

  Optimization Mode (ABCIP Default) - does not preserve the poke order and has maximum folding by only poking the last value of an item.

  When Poke Mode is set to Optimized, the Communication Driver will attempt to group consecutive tag writes (array elements) into a single request. Depending on the timing situation, there is no guarantee that consecutive tag writes will be grouped into a single request.

  Note: For more information on all Communication Driver Global Parameters, see the Communication Drivers Pack Help.

Data Type Determination

When a client sends a read/write request to the ABCIP Communication Driver, the server needs to know if the tag is defined in the controller; it also needs to know the tag’s data type and size. To determine this information, the ABCIP Communication Driver internally builds the item table (tag database) in the server-specific code before any item is created.

- This table includes information on the item's name, data type, and size.
- If an item is a structure, it also includes its members and their data types.

To build the table, the ABCIP Communication Driver sends a request to the controller for all the tag information defined in the controller. The controller then returns all the information needed. The table is built one time for each controller, unless a "refresh" request is received from the client. The ABCIP Communication Driver does not rely on the Allen-Bradley .csv and .L5K files.

Important: The manual "refresh" tag database request for the Logix processor needs to be activated by your writing "true" (of type VT_BOOL) to the $Sys$UpdateTagInfo; it is not activated by selecting the option (check box) as was implemented in the ABCIP OI Server 1.1.

Tag Database Status

To provide the status of the tag database for the Logix processor cached in the ABCIP Communication Driver, this version of the Communication Driver will implement a predefined, read-only system variable, $Sys$TagDBStatus, of type VT_I2.

This system variable takes on any of the following values:

- 0 – No tag database
- 1 – Uploading tag database
- 2 – tag database uploaded
- 3 – tag database upload failed

The value of $Sys$TagDBStatus can only be changed by poking to the system variable $Sys$UpdateTagInfo. Poking a TRUE to $Sys$UpdateTagInfo while $Sys$TagDBStatus is 1 will not cause consecutive tag database uploads to the ABCIP Communication Driver.

Note: $Sys$UpdateTagInfo and $Sys$TagDBStatus are only available as item names associated with the Logix processor.

Regardless of the status of the tag database upload, the ABCIP Communication Driver periodically syncs the tag database from the controller. The Logix5000 controller has a journaling capability that keeps track of the changes in its tag database. Whenever the tag database in the Logix5000 controller is changed, a new journal and version are generated within the controller.

Tag Database Version

The ABCIP Communication Driver periodically checks for version changes and uploads the journal information from the controller, so that the tag database it maintains matches the corresponding database in the controller.

You can monitor changes in the Logix tag database version by subscribing to the tag database system item $SYS$TagDBVersion at the hierarchy of any CompactLogix, FlexLogix, or ControlLogix controller.

The ABCIP Communication Driver shows the new database major and minor versions presented as a number in addition to uploading the journal information from the controller.

Note: For tag items defined as array data types in an item addition request, the OPC_E_BADTYPE error is returned when an OPC client does not specify the array data type or the VT_EMPTY data type. The only exception is when an OPC client specifies VT_BSTR as the requested data type for an item that is defined as VT_ARRAY|VT_UI1. In this case, the Communication Driver accepts the item addition and returns the data as VT_BSTR.

Invalid Items Handling

Item syntax verification is based on the type of controllers associated with it. The PLC-5 and SLC500 controllers have predefined syntax on their item names. When an item is specified for these two types of controllers, its item syntax will be verified immediately. If the item syntax is incorrect, the item is rejected immediately and will not be added to the Communication Driver does periodically send messages to the Logix5000 controller for tag database update. The item that has a BAD syntax will be re-evaluated when a new tag database has been downloaded to the Logix5000 controller. If the item is subsequently matched to an item in the new tag database, the item will automatically switch to a GOOD quality with the proper data value.Communication Driver

Logix5000 Online Tag Management

The ABCIP Communication Driver can detect online changes to the Logix5000 processor tag database and automatically update the status of these tags in your application.

Note: Tag change detection and updates are dependent on the Auto Load Tags on Activation, Auto Synchronize Tags, and Use Persisted Tags setting for these Logix5000-compatible controllers. For more information on On these tag database options, see Loading Tag Database from File

Adding or Removing Tags

When tags are added or removed from the Logix5000 processor, the Communication Driver can detect the change and update its internal tag database. If the newly added tags have already been accessed in your application, the quality of these tags will be changed to GOOD and their values updated. In the case when the tags are removed from the processor, the tags' quality will be changed to BAD.

Because the detection is done through periodic pollings of your Logix5000's status, there will be a delay between the time when tags are modified and the time when tags' information is updated in your application. The delay can be a few seconds to minutes, depending on how busy your Communication Driver is.

Making PLC Program Routine Changes

If you import a routine containing new tags to an online PLC, the Auto Synchronize Tags option will not synchronize the new tags with the Communication Driver. The Communication Driver will reject the new tags as invalid even with the Auto Synchronize Tags option enabled until the tag database is re-reset by poking a 1 to the system item $SYS$UpdateTagInfo.

As a best practice, changes of PLC program routines should be done when the PLC is offline. Changing or importing the PLC program routines when the PLC is online is not supported.

Modifying Tags Through Downloaded Programs

Tag information can also be modified with an updated program. When a program is downloaded to the Logix5000 processor while data access is in progress, the Communication Driver can detect the change of state in your Logix5000 processor. A message will be displayed in the logger to inform you about the event and data access to the processor will be temporarily suspended.

As soon as the program downloading process has completed, the Communication Driver will re-upload all the tag database from the Logix5000 processor and resume your access to the processor. All tags in your application will be updated to reflect the change.

Loading Tag Database from File

The ControlLogix, GuardLogix, SoftLogix, CompactLogix, and FlexLogix controllerads have options to upload the Tag database from the file. Each option can improve the tag database upload time depending on your tag database management setup.

Auto Load Tags on Activation

When the Communication Driver is activated, it can perform a tag database upload.

If **Auto Load Tags on Activation** is selected, the Communication Driver will check the controller database version on startup. If it is different from the version stored in the file, it will read the tags from the controller and synchronize the file. The Tag Database from File Options Matrix explains the database upload feature from the file.

If this option is NOT selected, the Communication Driver will not perform an upload upon activation but will wait until an item has been advised by a client.

Auto Synchronize Tags

If **Auto Synchronize** is selected, the Communication Driver will periodically check the controller version number and perform an upload if a newer version is present.

Important: If the optimization option **Optimize for Startup time** is selected, the **Auto Synchronize Tags** is automatically selected and unchangeable. In this situation, the Communication Driver needs to synchronize physical address of tags from device.

For information about importing new tags to an online PLC, see Making PLC Program Routine Changes.

Persisted Tags

The ControlLogix, GuardLogix, SoftLogix, CompactLogix and FlexLogix controllers have an option to use Persisted Tags for uploading the tag database from the file. This feature will improve the tag database upload time.

When the Communication Driver is activated with the **Persisted Tags** option selected, it reads the tags from the controller and stores them into a file under the bin\CIPTagDB directory.

If the version of the tag database matches the tag database file persisted from the last run, the ABCIP Communication Driver will skip the tag database upload option and use the persisted file as the basis of the tag database.

If the Communication Driver detects the controller database version is different from the version stored in the file, it will read the tags from the controller and synchronize the file.

The subsequent restart of the Communication Driver will read the tag database from this file. This file will store the database major and minor version information.

Important: If secured controllers (password protected), are a part of your hierarchy, changes in the Persisted Tags functionality will occur. See Tag Database from File Options Matrix for a detailed description of each option.

Tag Database from File Options Matrix

Tag Database Options	Selected (checked)	Not Selected (Unchecked)
Auto Load Tags on Startup (Configurable parameter in the editor)	The tag database will be uploaded as soon as the Communication Driver is activated. The Communication Driver will attempt to connect to the device only one time. If the device is not connected, it will retry when the first item is subscribed.	The tag database will be uploaded as soon as the first device item is subscribed. The system item $SYS$UpdateTagInfo can not be used to trigger a tag database upload until the first device item is subscribed.

Tag Database Options	**Selected (checked)**	**Not Selected (Unchecked)**
Auto Synchronize Tags (Configurable parameter in the editor)	The tag database in the Communication Driver will be synchronized periodically with the device. If the device is secured, the Communication Driver will not be able to automatically synchronize the tag database. **Note:** If the optimization option is set for **Optimize for startup time**, the value is always True. In this case, the Communication Driver needs to synchronize the physical addresses of tags from the device.	The tag database in the Communication Driver will not be synchronized with the device. The system item $SYS$UpdateTagInfo can be used to synchronize the tag database manually.
Use Persisted Tags	The Communication Driver will read the tags from the tag database file. If the file does not exist, it will then read the tags from the controller and store them into a file under bin\CIPTagDB directory. If the controller is unsecured and the database version is different from the controller version, then the Communication Driver will read the tags from the controller and store them into a file. The system item $SYS$UpdateTagInfo can be used to force the tag database upload from the device. **Note:** If the optimization option is **Optimize for startup time**, the physical address of the tags will also be stored in the file.	The Communication Driver will always upload the tag database from the device and store them in to a file.

Tag Database Options	**Selected (checked)**	**Not Selected (Unchecked)**
$SYS$UpdateTagInfo (System item can be accessed by any client application)	The tag database will be uploaded from the device if value **True** is poked to this item. This system item is provided for manual synchronization of the tag database. If the device is secured, use this item to synchronize tag database. **Note:** If **Use Persisted Tags** is enabled, the original file will be renamed to <####>_temp.aaTDB (where ### represent the serial number of the device). If the Communication Driver fails to upload tags from the device, it will use the renamed file to recover the database. The temporary file (<####>_temp.aaTDB) will be deleted, after the tag database is uploaded successfully.	Poking the value **False** will not affect the tag database.

Manual Tag Synchronization

This system item ($SYS$UpdateTagInfo) is provided for manual synchronization of tag database. If the device is secured (Password protected), use this item to synchronize tag database.

The system item $SYS$UpdateTagInfo can be accessed by any client application.

The tag database will be uploaded from the device if value **True** is poked to this item.

Note: If **Use Persisted Tags** is enabled, the original file will be renamed to <####>_temp.aaTDB (where ### represent the serial number of the device). If the Communication Driver fails to upload tags from device, it will use the renamed file to recover the database. The temporary file (<####>_temp.aaTDB) will be deleted after the tag database is uploaded successfully.

Poking the value **False** will not affect the tag database.

Accessing Secured Logix5000-series Controllers

When Logix5000 controllers are secured (Password protected), accessing the program version number will fail. When the controllers are secured, the tag database in the ABCIP Communication Driver may not be in-sync with the controller tag database.

If the run-time tag database synchronization has not been turned off and the controller is unsecured, the ABCIP Communication Driver will re-sync the tag databases at the next re-synchronization interval.

When the controller is secured an error is returned to the ABCIP Communication Driver indicating that the controller is secured and a message will be logged indicating that tag database re-synchronization failed because the controller is secured.

Because re-syncing is still running at the re-syncing interval, if the controller goes from secured to unsecured, the tag databases will be re-synced if necessary and a message will be logged indicating that the controller is unsecured.

The system variable $Sys$DeviceSecurity will indicate if the controller security is On or Off.

You can turn on or off, through configuration, the tag database re-syncing, to minimize the traffic between ABCIP Communication Driver and the controller.

Note: The server can be started and in-sync with the controller, and the controller can be secured and un-secured with no changes, so that the server is still in-sync with the controller. Even though the controller is secured, the tag database can still be uploaded. The error returned from the controller when the controller is secured is only on the program version check.

The Auto Synchronize Tag Functionality Matrix and Persisted Tag Functionality Matrix show when a tag database upload will occur or not occur based upon security.

Auto Synchronize Tag Functionality Matrix

Auto Synchronize Tag Configuration	Server Runtime behavior:	
	For unsecured controller	**For secured controller**
Selected (Checked)	Tag database version in the controller will be checked periodically and the version changes will be uploaded to the ABCIP Communication Driver automatically.	Tag database version in the controller will be queried but no upload will be made automatically. Changes in the tag database in the controller will only be uploaded when the $SYS$UpdateTagInfo system tag in the ABCIP Communication Driver is written into.
Unselected (Unchecked)	Tag database version will not be checked. Changes in the tag database in the controller will only be uploaded if the $SYS$UpdateTagInfo system tag in the ABCIP Communication Driver is written into.	Same behavior as if the controller is unsecured.

Persisted Tag Functionality Matrix

Persisted Tags Configuration	**Server Runtime behavior:**	
	For unsecured controller	**For secured controller**
Selected (Checked)	1. The Communication Driver will read the tags from the file. If the file does not exist then it will read the tags from controller and store them into a file under bin\CIPTagDB directory. 2. If the file database version is different from the controller version, then the Communication Driver will read the tags from the controller and store them into a file.	The Communication Driver will read the tags from the file. If the file does not exist, then it will read the tags from controller and store them into a file under the bin\CIPTagDB directory.
Unselected (Unchecked)	The Communication Driver will always upload the tags from the controller and store them into a file.	

Controller Time Stamping

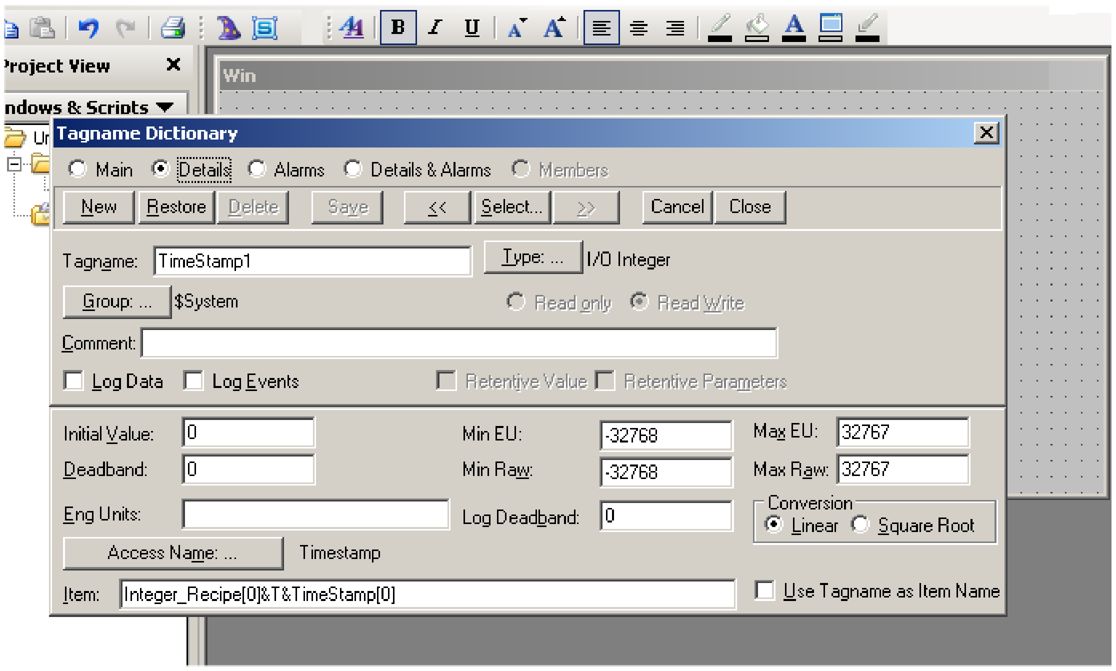
ABCIP Communication Driver has the capability to time stamp data changes with the controller's date and time as opposed to the PC's date and time. A new item syntax to time stamp data changes with the controller's date and time must be used.

Note: Controller Time Stamping is supported only in the Allen-Bradley Logix-family of controllers, version 16.x or later.

Important: The "TimeTag" in the controller must contain date and time as LINT type in UTC format. The logic behind the association between the specific DataTag & TimeTag pair is assumed to be user-defined in the controller program.

Specifying controller time stamping in native InTouch requires the time-stamping qualified DataTag plus both of its Date and Time Dotfield string tags.

The following sequence shows the Tagname Dictionary and the sample items.



Tagname Dictionary

Main [✓ Details] Alarms Details & Alarms Members

Tagname: TimeStamp1

Type: ... I/O Integer

Group: ... $System

Read only [✓ Read Write]

Comment:

Log Data
Log Events
Retentive Value
Retentive Parameters

Initial Value: 0
Deadband: 0
Eng Units:
Min EU: -32768
Min Raw: -32768
Log Deadband: 0
Max EU: 32767
Max Raw: 32767

Conversion: [✓ Linear] Square Root

Access Name: ... Timestamp

Item: Integer_Recipe[0]&T&TimeStamp[0]

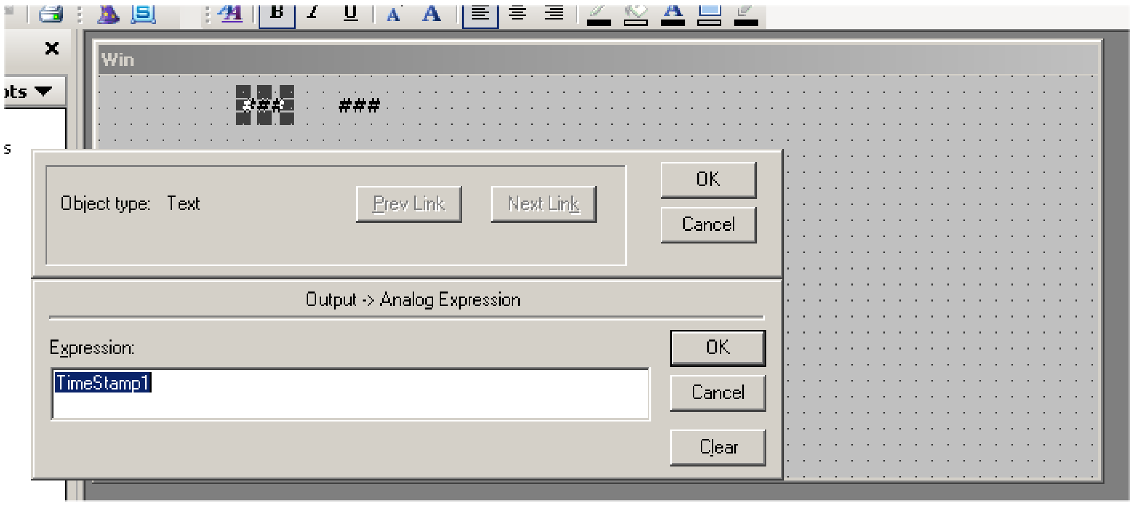
Use Tagname as Item Name

- DataTag: Can be almost any type including boolean, integer, string, and array. In the example, the Value tag is an integer for illustration purposes only.
- TimeTag: Must be string type.

When you enter an item name on the client side, you must enter an item name that is a Data tag and Time tag pair. You will use the "&T&" delimiter, to identify the time tag.

For example if you enter an item name such as DataTag&T&TimeTag, the Communication Driver will treat the item as two separate tags, "DataTag" and "TimeTag", and will validate each tag separately.

DataTag example "TimeStamp1":
```
Integer_Recipe(0)&T&TimeStamp(0)
```



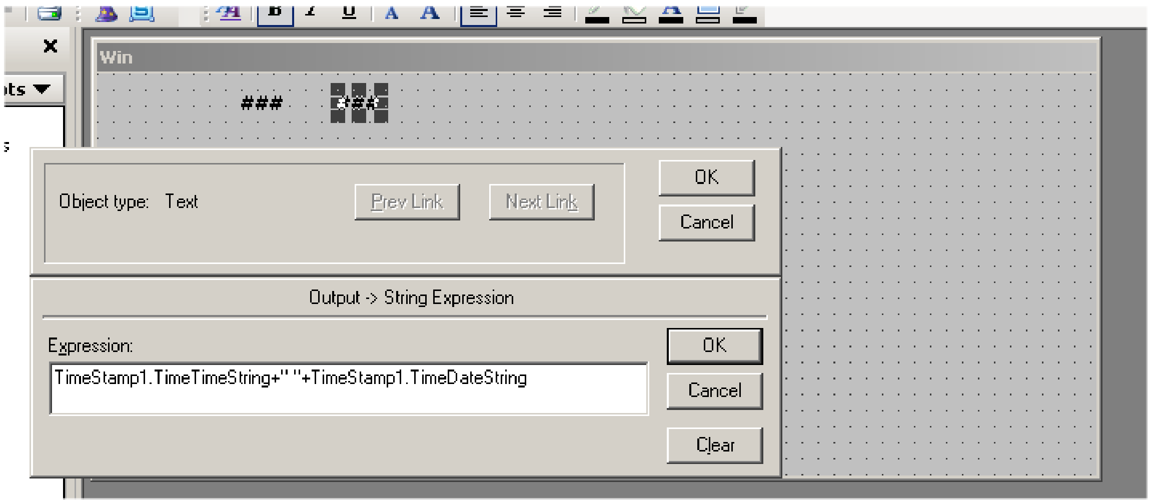
Object type: Text

Output -> Analog Expression

Expression:

TimeStamp1

TimeTag example, concatenated:
```
TimeStamp1.TimeTimeString+""+TimeStamp1.TimeDateString
```



TimeStamp1.TimeTimeString+" "+TimeStamp1.TimeDateString

The Communication Driver will read the data for the two tags from the controller.

When the pair of values is read by the server, the TimeTag value will be used to time stamp the DataTag value before sending the updates to the client.

If you enter an item name such as DataTag only, the value read from the controller is time stamped with the PC's date and time before sending the updates to the client.

Note: When advising an item using timestamping with "&T&", and communication is lost with the controller, the Communication Driver will timestamp the item and update its quality.

Item Names/Reference Descriptions

The ABCIP Communication Driver currently supports item names that follow the conventions described for the various Allen-Bradley ControlLogix, CompactLogix, FlexLogix, PLC-5, SLC500, and MicroLogix families of controllers.

- Logix5000 Item Naming
- PLC-5 Item Naming
- SLC500 Item Naming
- MicroLogix Item Naming
- OI Server-Specific System Item
- Generic OPC Syntax

Logix5000 Item Naming

The Logix5000 controllers (ControlLogix, CompactLogix, FlexLogix, GuardLogix and SoftLogix) store data in tags, whose names you create. This is in contrast to the traditional Allen-Bradley PLC-5, SLC500 or MicroLogix controllers which store data in data/section files, whose names must follow the vendor-predefined naming convention.

The Logix5000 tags uses arrays instead of file numbers in addressing a set of multiple items. That is, "[]" would be accepted as a valid symbol but ":" would be rejected for the tag name. The Logix5000 item syntax is shown in the following table. The Communication Driver will adhere to this syntax for native mode.

Note: A tagname can be up to 40 characters in length and cannot include a file number. File numbers are not applicable to Control Logix. File numbers are valid for PLC5, SLC500 and MicroLogix only.

Reference	**Syntax**
Program tag	Program:<Program_Name>.<Tag_Name>
IO tag	<Location>:<slot_#>:<Data_ Type><Member_Name>.<SubMember_Name>.[<bit_#>]
Entire tag	<Tag_Name>
Member of structure tag	<Tag_Name>.<Member_Name>
Array element	<Tag_Name>[<element_X>]
Two-dimensional array element	<Tag_Name>[<element_X>,<element_Y>]
Three-dimensional array element	<Tag_Name>[<element_X>,<element_Y>,<element_Z>]
Block reads/writes of one-dimensional arrays (supported types: BOOLS, SINTS, INTS, DINTS, REALS, LONG)	<Tag_Name>[<element_X>],L<number_of_items_#>
String tag	<String_Tag_Name>[.DATA[[<element_#>] *]]* *[*,SC<string_length_#>*]* <String_Tag_Name>[.DATA[[<element_#>] *]]* *[*,SP<string_length_#>*]* <String_Tag_Name>[.DATA[[<element_#>] *]]* *[*,SS<string_length_#>*]*
String tag array	<String_Tag_Name>[<element_X>][.DATA *[*[<element_#>]*]]* *[*,SC<string_length_#>*]* <String_Tag_Name>[<element_X>][.DATA *[*[<element_#>]*]]* *[*,SP<string_length_#>*]* <String_Tag_Name>[<element_X>][.DATA *[*[<element_#>]*]]* *[*,SS<string_length_#>*]*
Bit within integer	<Tag_Name or Member_Name>.<bit_#>
Read-only item syntax to read controller time-stamped data	<Tag_Name>&T&<Time_Tag> <Hierarchy_Node_Path><Tag_Name>&T&[.]<TimeTag>

Reference	Syntax
Note: When the data and timestamp are located in the same structure (e.g. UDT), the optional period following the &T& delimiter when entering the item name for structures reduces the need to retype the same structure name for the time tag.	Example: *A.B.C.D.DataTag&T&A.B.C.D.TimeTag* *A.B.C.D.DataTag&T&.TimeTag*
Note: [DT] qualifier is an option to subscribe the value (LINT) in date and time format.	Example: *A.B.C.D.TimeTag DT* **Note**: A space must be inserted between the <TimeTag> and DT qualifier. Using the [DT] qualifier causes the tag to become Read Only.

In the preceding table:

- *[ ]* italicized brackets designate element as optional.
- [ ] not italicized brackets denote array index.
- < > means user input (as defined in the controller program).
- String placeholder (start with uppercase): Location, Program_Name, Tag_Name, Data_Type, Member_Name, SubMember_Name, and String_Tag_Name.
- Numeric placeholder (all in lowercase): elemeny_#, element_X, element_Y, element_Z, string_length_#, slot_#, bit_#, and number_of_items_#.
- <Location>identifies network location as:
  LOCAL = Local rail or chassis
  <Adapter_Name> = Name of the remote module
- <Data_Type> is represented by a single letter as follows:
  I=input, O=output, C=configuration, and S=status.
- All others are predefined keywords or symbols.

Examples:

Tag Name	Example
String tag array	BatchRecipe[4], BatchRecipe[4].DATA BatchRecipe[4].DATA[0],sc82 (all of them return the same data)
Two-dimensional array tag	Mixer_StepTimer_Preset[3,5]
User-defined structure tag	ProductionUnit.AssemblyLine[2].Counter[4]

Program tag	Program:MainProgram.Tank[1,2,4].Level Program:UserProgram.OperationMode
Module tag	Local:6:O.Data.31 Remote_IO:2:C.ProgValue

Note: A STRING type member is implicitly a structure in the form of StringTag.DATA and StringTag.LEN (where the DATA member is an array of 82 elements and the LEN member defines the actual length of the string). Therefore, a string member consumes two nesting levels by default.
The length field of a string will be used to determine the length of the string to be returned if the DATA member is not explicitly included in the string specification when the string is put into subscription.

Note: The "DT" qualifier returns a UTC date/time for OPC Clients requesting a "VT_DATE" binary value. For DDE and SuiteLink clients, requesting a "VT_BSTR", the date is converted to a UTC Date/Time string. The dates supported by the Date/Time string include values from 1/1/1970 12:00:00AM (GMT) to 8/30/2920 5:19:59AM (GMT).

Module-Defined Data Types

Module-defined data types are created automatically in the RSLogix5000 software after their corresponding I/O or DeviceNet modules are defined.

- Module-defined tags do not allow user modification.
- Formats are fixed by the Logix5000 controller.

User-Defined Data Types

The ABCIP Communication Driver supports read and write of user-defined data types. The Logix user-defined data type is a custom-made structure consisting of members that can be atomic, arrays (single dimension only), or structures themselves.

The user-defined data-type tags can be atomic or arrays up to three dimensions. The members of the structure can be any data types supported by this Communication Driver. If a structure contains another structure as its member, the maximum nesting supported is up to 20 levels.

Each level of members in a structure or each array dimension within a user-defined tag consumes one nesting level. The individual bits that make up a structure member do not constitute a nesting level.

The Communication Driver supports the optimization of user-defined data types. For information on UDT optimization, see Logix5000 Read Optimization.

Block Reads and Writes of Arrays

The ABCIP Communication Driver supports Block Reads and Writes of one-dimensional arrays from the supported ControlLogix, FlexLogix, and CompactLogix controllers.

The following features are not supported by the Communication Driver:

- Block Reads/Writes of strings.
- Block Reads/Writes of structures (either predefined or user-defined).

Note: The requested block size cannot exceed 486 bytes.

There are five different data types that are supported, each of which requires a different allowance on the qualifier due to the block size limitation.

There are three optimization modes supported, each with a different maximum qualifier allowance as shown in the following table: Optimize for Reads, Optimize for Startup, and No Optimization.

Note: The number in the "Ln" qualifier should not need an offset, because it is the total number counting from 1 (one).

	Qualifier Allowance (n)	
Data Type	**Optimize for Read**	**Optimize for Startup No Optimization**
Boolean (VT_BOOL)	3840	3831
SINT (VT_I1)	486	478
INT (VT_I2)	243	239
DINT (VT_I4)	114	114
Real (VT_R4)	121	119
LINT (VT_I8)	60	59

Note: Boolean array tags may allow up to 3872 items in a block if the specified range of array elements fits exactly into a contiguous block of DINT-based (4-byte) memory units. That is, Boolean array item block starting from array index zero or at every quadruple of byte (32-bits) margin.
For example, index 0, 32, 64, 96, … can exploit this feature to the maximum.

The Block Reads and Writes of Arrays feature works differently for a DDE/SuiteLink client and OPC client.

- In an OPC client, the array of data is displayed as an array of values (a series of data) separated by ";" according to their data types.
- In a DDESuiteLink client, the array of data is expressed as a string of Hex data block, of which each unit occupies the same byte size as defined by the data types.
  - The Hex value contained in each unit of the data block is equivalent to the decimal quantity stored in each individual item in the controller.
  - The data in the array block can be parsed according to the byte size of the data type.
  - The Hex value can be converted to its equivalent decimal quantity for use in the application.

    For example:

    A DINT (double integer data type) item occupies 4 (four) bytes of data, which amounts to 8 (eight) Hex digits.
    An array block of DINT items from the InTouch HMI using DDESuiteLink should be parsed into individual units of 8 (eight) Hex characters.
    Then each unit of parsed data needs to be converted from Hex to its equivalent decimal value for usage.

PLC-5 Item Naming

The general format of item names for data from the PLC-5 controllers matches the naming convention used by the programming software. The following is the format:

[$] X [file] : element [.field] [/bit]

Note: The parts of the name shown in square brackets ([]) are optional.

Item Name	Description
$	Purely optional.
X	Identifies the file type. The following table summarizes the valid file types, the default file number for each type, and the fields allowed (if any).
file	File number (0 - 999 decimal). • File 0 must be Output. • File 1 must be Input. • File 2 must be Status.
element	Element number within the file. • For Input and Output files it is also called rack-and-group number and must be 0 - 777 octal. • For all other file types, it must be 0 - 999 decimal.
.field	Valid only for Counter, Timer, ASCII String, PID, SFC Status, Block Transfer, and Control files. Refer to the following table.
/bit	Valid for all file types except ASCII String and Floating Point. • For Input and Output files it must be 0 - 17 octal. • For all other file types it must be 0 - 15 decimal.

Identifier	File Type	Default File #	.fields
O	Output	0	N/A
I	Input	1	N/A
S	Status	2	N/A
B	Binary	3	N/A
T	Timer	4	.PRE .ACC .EN .TT .DN

Identifier	File Type	Default File #	.fields
C	Counter	5	.PRE .ACC .CU .CD .DN .OV .UN
R	Control	6	.LEN .POS .EN .EU .DN .EM .ER .UL .IN .FD
N	Integer	7	N/A
F	Floating Point	8	N/A
A	ASCII	None	N/A
D	BCD	None	N/A
ST	ASCII String*	None	.LEN
PD	PID*	None	.ADRF .ADRE .BIAS .CA .CL .CT .DB .DO .DVDB .DVN .DVNA .DVP .DVPA .EN .ERR .EWD .INI .KD .KI .KP .MAXI .MAXO .MAXS .MINI .MINO .MINS .MO .OLH .OLL .OUT .PE .PV .PVDB .PVH .PVHA .PVL .PVLA .PVT .SO .SP .SPOR .SWM .TIE .UPD
SC	SFC Status*	None	.DN .ER .FS .LS .OV .PRE .SA .TIM
BT	Block Transfer* (Read-Only)	None	.EN .ST .DN .ER .CO .EW .NR .RW .TO .RLEN .DLEN .FILE .ELEM
MG	Message	None	.NR .TO .EN .ST .DN .ER .CO .EW .ERR .RLEN .DLEN .DATA[0] through .DATA[51]
CT	CNet Message	None	.TO .EW .CO .ER .DN .ST .EN .ERR .RLEN .DLEN .FILE .ELEM

* Available only on certain PLC-5 models. Check the Processor Manual for the model being used.

Output File Items

O[n]:rg[/b]	n represents the file number and it is optional. If specified, it must be 0 (zero).
	r indicates the rack number (0 - 27 octal).
	g indicates the I/O group (0 - 7 octal).
	b specifies the bit (0 - 17 octal). /b may be omitted, if necessary, to treat the I/O group as a numeric value.

Examples:

O0:00/0

$O:177/17

O:3 4BCD (for 16-bit 7-segment display)

Input File Items

I[n]:rg[/b]	n represents the file number and is optional. If specified, it must be 1 (one).
	r indicates the rack number (0 - 27 octal).
	g indicates the I/O group (0 - 7 octal).
	b specifies the bit (0 - 17 octal). /b may be omitted, if necessary, to treat the I/O group as a numeric value.

Examples:

I1:0/0

I:177/17

I:3 4BCD (for 16-bit thumbwheel input)

Status File Items

S[n]:e[/b]	n represents the file number and is optional. If specified, it must be 2 (two).
	e indicates the element number in the file.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).

Note: Refer to the 1785 PLC-5 Family Processor Manual (Allen-Bradley Publication 1785-6.8.2) for a complete description of the Status file information.

Examples:

$S:18 (year)

$S2:18 (year)

S2:19 (month)

S2:10/0 (battery low status bit)

Binary File Items

B[n]:e[/b] or B[n]/m	n represents the file number and is optional. If not specified, it is assumed to be 3 (three). If specified, the file number must be 3 - 999 decimal.

	e specifies the element (word) number within the Binary file. It must be 0 - 999 decimal.
	b specifies the bit number within the word and is optional. In the first form (where :e is present), the bit number must be 0 - 15 decimal.
	m specifies the bit number within the file. However, in the second form, no word numbers are specified and the bit number may be 0 - 15999.

Examples:

B:33

B:6/4 (same bit as B/100)

B3/15999 (same bit as B:999/15)

Timer File Items

T[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 4 (four). If specified, the file number must be 3 - 999 decimal.
	e specifies the element number (three words per element) within the Timer file. It must be 0 - 999 decimal.
	f identifies one of the valid Timer fields. The valid fields for Timer Files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. All of the fields of a timer can be accessed by specifying the .f fields. However, it is possible to use /b to single out a bit in the .PRE or .ACC fields (which are words). If specified, the bit number must be 0 - 15 decimal.

Examples:

T4:0.ACC

T4:0.DN

T4:1.PRE

Counter File Items

C[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 5 (five). If specified, the file number must be 3 - 999 decimal.

	e specifies the element number (three words per element) within the Counter file. It must be 0 - 999 decimal.
	f identifies one of the valid Counter fields. The valid fields for the Counter files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a counter. However, it is possible to use /b to single out a bit in the .PRE or .ACC fields (which are words). If specified, the bit number must be 0 - 15 decimal.

Examples:

C5:0.ACC

C5:3.OV

C5:1.PRE

Control File Items

R[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 6 (six). If specified, the file number must be 3 - 999 decimal.
	e specifies the element number (three words per element) within the Control file. It must be 3 - 999 decimal.
	f identifies one of the valid Control fields. The valid fields for Control files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a Control file. However, it is possible to use /b to single out a bit in the .LEN or .POS fields (which are words). If specified, it indicates the bit (0 - 15 decimal).

Examples:

R6:0.LEN

R6:3.EM

R6:1.POS

Integer File Items

N[n]:e[/b]	n represents the file number and is optional. If not specified, it is assumed to be 7 (seven). If specified, the file number must be 3 - 999 decimal.
	e specifies the element number within the Integer file. It must be 0 - 999 decimal.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).

Examples:

N7:0

N7:0/15

N7:3

Floating Point File Items

F[n]:e	n represents the file number and is optional. If not specified, it is assumed to be 8 (eight). If specified, the file number must be 3 - 999 decimal.
	e specifies the element number within the Floating Point file. It must be 0 - 999 decimal.

Examples:

F8:0

F8:3

ASCII File Items

An:e[/b] An:x-y	n represents the file number (NOT optional) and must be 3 - 999 decimal.
	e specifies the element number within the ASCII file. It must be 0 - 999 decimal. Each element in an ASCII file contains two ASCII characters.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).
	x and y also specify element numbers. In this form, the item is an ASCII string occupying element x through element y. Each element contains two ASCII characters: the first character is the high-order byte and the second is the low-order, and so on.

Note: If reading only one word as a two-character string, the range must be "x-x." For example, A20:3-3.

Examples:

A20:3

A10:0/0

A9:0-19 (40-character ASCII string)

BCD File Items

Dn:e[/b]	n represents the file number (NOT optional) and must be 3 - 999 decimal.
	e specifies the element number within the BCD file. It must be 0 - 999 decimal. Each element in a BCD file contains a number between 0 - 9999.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).

Examples:

D20:3

D10:0/3

ASCII String Section Items

STn:e[.f]	n represents the file number (NOT optional) and must be 3-999 decimal.
	e specifies the element number within the String file. It must be 0 - 779 decimal. Each element in a String file contains an ASCII string with a maximum length of 82 characters.
	f identifies the following ASCII string field: .LEN. If .f is omitted, it is assumed to be the string.

Examples:

ST9:0

ST9:700

ST9:700.LEN

Block Transfer Section Items

BTn:e[.f][/b]	n represents the file number (NOT optional) and must be 3 - 999 decimal.
	e specifies the element number (three words per element) within the Block Transfer file (0 - 999 decimal).

	f identifies one of the valid Block Transfer fields. The valid fields for Block Transfer items are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a Block Transfer. However, it is possible to use /b to single out a bit in the .FILE or .ELEM fields (which are words). If specified, the bit number must be 0 - 15 decimal.

Note: Block Transfer files are read-only.

Examples:

BT9:0.EN

BT9:3.RLEN

BT9:3.FILE

PID Section Items

PDn:e.f[/b]	n represents the file number (NOT optional) and must be 3 - 999 decimal.
	e specifies the element number within the PID file. It must be 0 - 398 decimal.
	f identifies one of the valid PID fields. The valid fields for PID files are listed in the table. If PID field .ADDR is needed, use .ADRE for element and .ADRF for file.
	b is optional and is normally not used. All of the fields of a PID can be accessed by specifying the .f fields. If specified, it indicates the bit (0 - 15 decimal).

Important: Access to PID files may degrade the Communication Driver's performance due to the extreme size of the PID element (82 words each). If accessing only a few PIDs at a time, performance will not be greatly affected. If accessing a few fields of many PIDs at the same time, it may be faster to move the needed fields to an intermediate file (Floating Point or Binary) and let the Communication Driver access the intermediate files.

Examples:

PD9:2.SP

PD9:3.OLH

PD9:0.INI

SFC Status Section Items

SCn:e[.f][/b]	n represents the file number (NOT optional) and must be 3 - 999 decimal.
	e specifies the element number within the SFC Status file. It must be 0 - 999 decimal.
	f identifies one of the valid SFC fields. The valid fields for SFC files are listed in the table.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of an SFC. If specified, the bit number must be 0 - 15 decimal.

Examples:

SC9:0

SC9:0.PRE

SC9:0.SA

Message Section Items

MGn:e[.f] [/b]	n represents the file number (NOT optional) and must be 3 - 999 decimal.
	e specifies the element number within the SFC Status file. It must be 0 - 999 decimal.
	.f identifies one of the valid MSG fields. The valid fields for MSG files are listed in the table.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a .MG. However, it is possible to use /b to single out a bit in the word fields. If specified, the bit number must be 0 - 15 decimal.

Important: Access to MSG files may degrade the Communication Driver's performance, due to the extreme size of the MSG file element (56 words each). If accessing only a few MSG elements at one time, performance will not be affected greatly. However, if accessing a few fields of many MSG file elements at one time, it may be faster to move the needed fields to an intermediate file (Binary or Integer) and let the Communication Driver access the intermediate files.

Examples:

MG9:0.NR

MG255:1.DLEN

CNetMessage Control Block Items

CTn:e[.f][/b]	n represents the file number (NOT optional) and must be 3 - 999 decimal.

	e specifies the element number within the CT file. It must be 0 - 999 decimal.
	f identifies one of the valid CT fields. Valid CT fields are listed in the table.
	b is optional and normally not used. Specifying the .f fields can access all of the fields of a CT. If specified, the bit number must be 0 - 15 decimal.

Examples:

CT10:0

CT10:0.TO

CT10:0.ELEM

SLC500 Item Naming

The general format of item names for data from the SLC500 controllers matches the naming convention used by the programming software. The format is as follows:

[$] X [file] : element [.field] [/bit]

Note: The parts of the name shown in square brackets ([]) are optional.

Item Name	Description
$	Purely optional.
X	Identifies the file type. The following table summarizes the valid file types, the default file number for each type, and the fields allowed (if any).
file	Identifies the file number. • File numbers must be 0 - 255 decimal. • File 0 must be Output. • File 1 must be Input. • File 2 must be Status. • All other file numbers, 9 - 255 decimal, are open to all file types.
element	Identifies the element number within a file. • For Input and Output files it must be between 0 and 30 decimal. • For all other file types, the element number must be 0 - 255 decimal.

Item Name	Description
.field	Valid only for Counter, Timer, and Control files. See the following table.
/bit	Valid for all file types except ASCII String and Floating Point. • For Input and Output files it must be 0 - 17 octal • For all other file types it must be 0 - 15 decimal.

Identifier	File Type	Default File #	.fields
O	Output*	0	N/A
I	Input*	1	N/A
S	Status	2	N/A
B	Binary	3	N/A
T	Timer	4	.PRE .ACC .EN .TT .DN
C	Counter	5	.PRE .ACC. CU .CD .DN .OV .UN .UA
R	Control	6	.LEN .POS .EN .DN .ER .UL .IN .FD
N	Integer	7	N/A
F	Floating Point*	8	N/A
A	ASCII*	None	N/A
ST	ASCII String*	None	.LEN

*Available only on certain SLC500 models. Check the Processor Manual for the model being used.

Output File Items

O[n]:e[/b]	n represents the file number and is optional. If specified, it must be 0 (zero).
	e indicates the element number in the file (0 - 255).
	b specifies the bit (0 - 15 decimal). /b may be omitted, if necessary, to treat the I/O group as a numeric value.

Note: The elements in I/O modules are sequentially mapped into a memory table, and are different from the item names in the controller programming software. Refer to the *Addressing SLC I/O Modules* section.

Examples:

O0:0/0

$O:2/15

O:3 4BCD (for 16-bit 7-segment display)

Input File Items

I[n]:e[/b]	n represents the file number and is optional. If specified, it must be 1 (one).
	e indicates the element number in the file (0 - 255).
	b specifies the bit (0 - 15 decimal). /b may be omitted if necessary to treat the I/O group as a numeric value.

Note: The elements in I/O modules are sequentially mapped into a memory table and are different from the item names in the controller programming software. Refer to the *Addressing SLC I/O Modules* section.

Examples:

I1:0/0

I:2/15

I:3 4BCD (for 16-bit thumbwheel input)

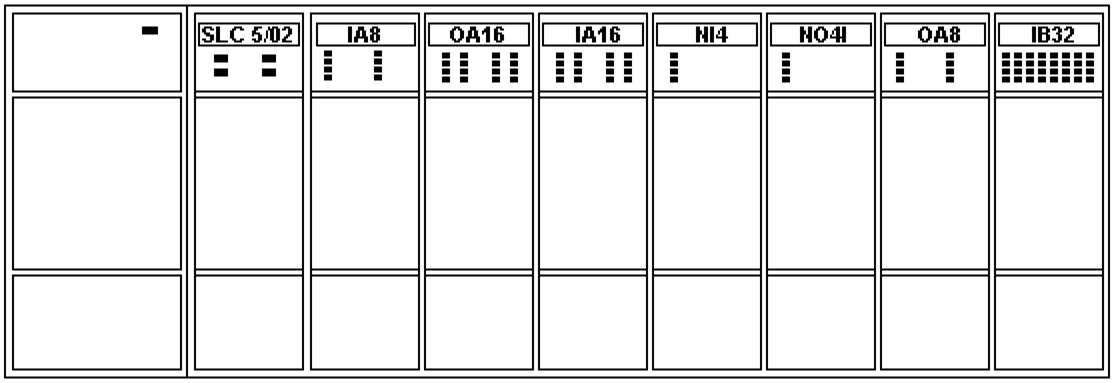
Addressing SLC I/O Modules

The elements (words) in I/O modules are mapped into a memory table. If the Analog I/O modules are being used, then the point naming will differ from the point naming in the programming software. The Communication Driver item name must be computed from the sum total of words used by the previous input or output blocks. The operator can use the programming software Data Monitor to look at the memory map of the I file or O file to verify your address. If the address is unsure, or if the controller configuration is likely to change, copy the points in question to the N table or B table, and access the data from there.

The naming conventions used in the Allen-Bradley programming software are not supported by the Allen-Bradley Ethernet Direct Communication Driver. The addressing convention is similar to that of the PLC-5 family processors. To derive the correct address for each I/O point, see the following Diagram System. Also see the following topics, Label I/O Modules with "Word Counts"," Sequentially Number the Input Modules, and Sequentially Number the Output Modules, to complete addressing the SLC I/O modules.

Diagram System

Addressing of the I/O points begins by drawing a schematic of the system. The following figure is a diagram of the SLC-5/02 system.



The far left unit is the power supply.

From left to right, the modules are:

1747-L524	SLC-5/02 Module Processor
1746-IA8	8-point 120VAC input module
1746-OA16	16-point 120VAC output module
1746-IA16	16-point 120VAC input module
1746-NI4	4-point 20mA analog input module
1746-NO4I	4-point 20mA analog output module
1746-0A8	8-point 120VAC input module
1746-IB32	32-point DC input module

Label I/O Modules with "Word Counts"

The address of any point within the I/O data table space, in an SLC processor, is the sum of the words occupied by previous modules (to the left in the rack) of the same type. Therefore, to determine the correct address for any particular point in the I/O data table, the number of words each module will consume must be known. Refer to the following list:

Number of Words	**Module**	
0	1747-L524	SLC-5/02 Module Processor
1	1746-IA8	8-point 120VAC input module
1	1746-OA16	16-point 120VAC output module
1	1746-IA16	16-point 120VAC input module
4	1746-NI4	4-point 20mA analog input module
4	1746-NO4I	4-point 20mA analog output module
1	1746-0A8	8-point 120VAC input module
2	1746-IB32	32-point DC input module

Note: In the preceding table, the minimum number of words which can be consumed by a module is 1 (16-bits). This is due to the memory scheme of all Allen-Bradley processors.

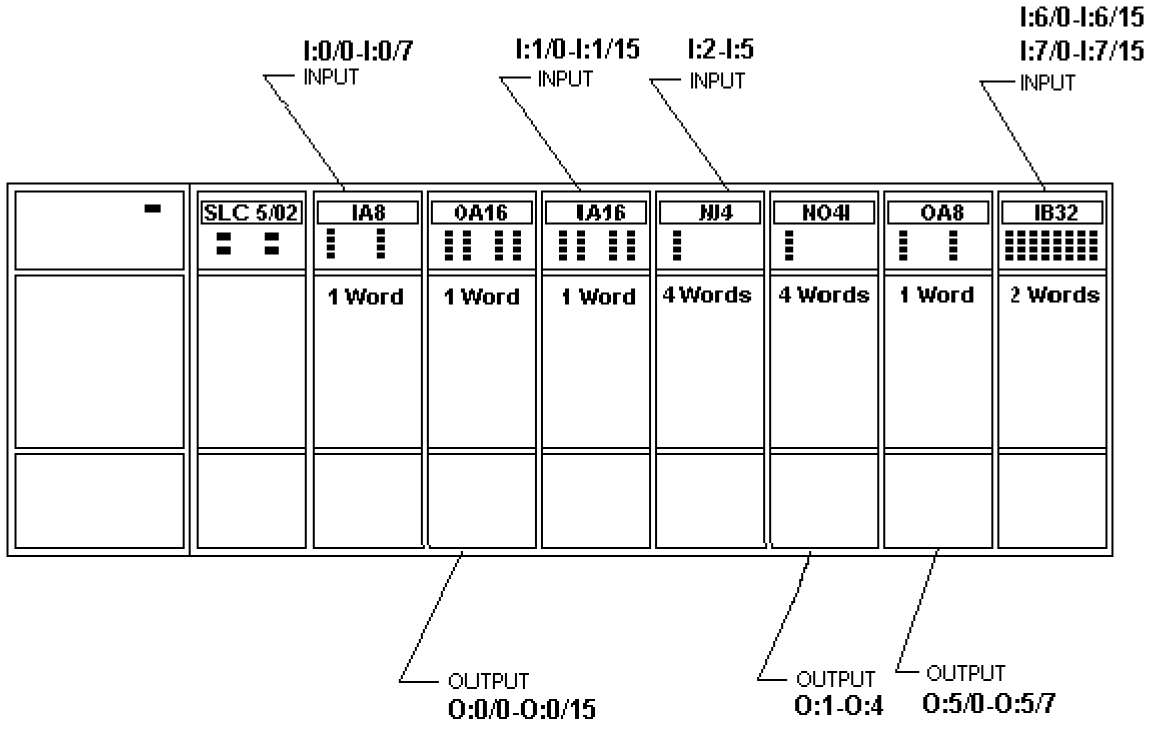
Sequentially Number the Input Modules

In the following I/O diagram, the first input module's addressing should start with "I:0." As previously noted, this module consumes one data table word. Therefore, the addressing of the next INPUT module encounter, moving from left to right, will begin with "I:1," regardless of the module's physical location.

Sequentially Number the Output Modules

In the following I/O diagram, the first output card encountered is the OA16. Although it is not in the first slot, its address will be "O:0" ("OHH, colon, ZERO"). This module consumes one data table word. Therefore, the addressing of the next OUTPUT module, moving from left to right, will begin with "O:1," regardless of the module's physical location.

I/O Diagram



Status File Items

S[n]:e[/b]	n represents the file number and is optional. If specified, it must be 2 (two).
	e indicates the element number in the file.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).

Note: Refer to the SLC500 Family Processor Manual (Allen-Bradley Publication) for a complete description of the Status file information.

Examples:

S2:6 (major error fault)

S2:13 (math register)

S:1/5 (forces enabled)

Binary File Items

B[n]:e/b or B[n]/m	n represents the file number and is optional. If not specified, it is assumed to be 3 (three). If specified, the file number must be 3 or 9 - 255 decimal.
	e specifies the element (word) number within the Binary file. It must be 0 - 255 decimal.
	b specifies the bit number within the word. In the first form (where :e is present), the bit number must be 0 - 15 decimal.
	m also represents the bit number. However, in the second form, no word numbers are specified and the bit number may be 0 - 4095.

Examples:

B:33

B:6/4 (same bit as B/100)

B3/4095 (same bit as B:255/15)

Timer File Items

T[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 4 (four). If specified, the file number must be 4 or 9 - 255 decimal.
	e specifies the element number (three words per element) within the Timer file. It must be 0 - 255 decimal.
	.f identifies one of the valid Timer fields. The valid fields for Timer Files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. All of the fields of a timer can be accessed by specifying the .f fields. However, it is possible to use /b to single out a bit in the .PRE or .ACC fields (which are words). The bit number must be 0 - 15 decimal.

Examples:

T4:0.ACC

T4:3.DN

T4:1.PRE

Counter File Items

C[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 5 (five). If specified, the file number must be 5 or 9 - 255 decimal.
	e specifies the element number (three words per element) within the Counter file. It must be 0 - 255 decimal.
	.f identifies one of the valid Counter fields. The valid fields for the Counter Files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a counter. However, it is possible to use /b to single out a bit in the .PRE or .ACC fields (which are words). The bit number must be 0 - 15 decimal.

Examples:

C5:0.ACC

C5:3.OV

C5:1.PRE

Control File Items

R[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 6 (six). If specified, the file number must be 6 or 9 - 255 decimal.
	e specifies the element number (three words per element) within the Control file. It must be 0 - 255 decimal.
	f identifies one of the valid Control fields. The valid fields for the Control files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.

	b is optional and is normally not used. All of the fields of a Control file can be accessed by specifying the .f fields. However, it is possible to use /b to single out a bit in the .LEN or .POS fields (which are words). The bit number must be 0 - 15 decimal.

Examples:

R6:0.LEN

R6:3.EN

R6:1.POS

Integer File Items

N[n]:e[/b]	n represents the file number and is optional. If not specified, it is assumed to be 7 (seven). If specified, the file number must be 7 or 9 - 255 decimal.
	e specifies the element number within the Integer file. It must be 0 - 255 decimal.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).

Examples:

N7:0

N7:0/15

N7:3

Floating Point File Items

F[n]:e	n represents the file number and is optional. If not specified, it is assumed to be 8 (eight). If specified, the file number must be 8 - 255 decimal.
	e specifies the element number within the Floating Point file. It must be 0 - 255 decimal.

Examples:

F8:0

F8:3

ASCII File Items

An:e[/b] An:x-y	n represents the file number (NOT optional) and must be 9 - 255 decimal.

	e specifies the element number within the ASCII file. It must be 0 - 255 decimal. Each element in an ASCII file contains two ASCII characters.
	b is optional. If specified, it indicates bit (0 - 15 decimal).
	x and y also specify element numbers. In this form, the item is an ASCII string occupying element x through element y. Each element contains two ASCII characters: the first character is the high-order byte and the second is the low-order, and so on.

Note: If reading only one word as a two-character string, the range must be "x-x." For example, A20:3-3.

Examples:

A20:3

A10:0/0

A9:0-19 (40-character ASCII string)

ASCII String Section Items

STn:e	n represents the file number (NOT optional) and must be 9 - 255 decimal.
	e specifies the element number within the String file. It must be 0 - 255 decimal. Each element in a String file contains an ASCII string with a maximum length of 82 characters.

Examples:

ST9:0

ST9:200

MicroLogix Item Naming

The general format of item names for data from the MicroLogix controllers matches the naming convention used by the programming software. The following is the format:

[$] X [file] : element [.field] [/bit]

Note: The parts of the name shown in square brackets ([]) are optional.

Item Name	Description
$	Purely optional.

Item Name	Description
X	Identifies the file type. The following table summarizes the valid file types, the default file number for each type, and the fields allowed (if any).
file	Identifies the file number. • File numbers must be 0 - 999 decimal. • File 0 (zero) must be Output. • File 1 (one) must be Input. • File 2 (two) must be Status. • All other file numbers, 9 - 255 decimal, are open to all file types.
element	Identifies the element number within a file. • For Input and Output files it must be between 0 and 777 octal. • For all other file types it must be 0 - 999 decimal.
.field	Valid only for Counter, Timer, ASCII String, PID, SFC Status, Block Transfer, and Control files. Refer to the following table.
/bit	Valid for all file types except ASCII String and Floating Point. • For Input and Output files it must be 0 - 17 octal. • For all other file types it must be 0 - 15 decimal.

Identifier	File Type	Default File #	.fields
O	Output	0	N/A
I	Input	1	N/A
S	Status	2	N/A
B	Binary	3	N/A
T	Timer	4	.PRE .ACC .EN .TT .DN
C	Counter	5	.PRE .ACC .CU .CD .DN .OV .UN
R	Control	6	.LEN .POS .EN .EU .DN .EM .ER .UL .IN .FD
N	Integer	7	N/A
F	Floating Point	8	N/A

Identifier	File Type	Default File #	.fields
A	ASCII	None	N/A
L	Long	None	N/A
ST	ASCII String*	None	.LEN
PD	PID*	None	.TM .AM .CM .OL .RG .SC .TF .DA .DB .UL .LL .SP .PV .DN .EN .SPS .KC .TI .TD .MAXS .MINS .ZCD . CVH .CVL .LUT .SPV .CVP
MG	Message	None	.IA .RBL .LBN .RBN .CHN .NOD .MTO .NB .TFT .T FN .ELE .SEL .TO .CO .EN .RN .EW .DN .ER .ST

* Available only on certain MicroLogix models. Check the Processor Manual for the model being used.

Output File Items

O[n]:e[/b]	n represents the file number and is optional. If specified, it must be 0 (zero).
	e indicates the element number in the file (0 - 255).
	b specifies the bit (0 - 15 decimal). /b may be omitted, if necessary, to treat the I/O group as a numeric value.

Note: The elements in I/O modules are sequentially mapped into a memory table, and are different from the item names in the controller programming software. MicroLogix and SLC500 adopt the same I/O addressing format. Refer to the *Addressing SLC I/O Modules* section for details.

Examples:

O0:0/0

$O:2/15

O:3 4BCD (for 16-bit 7-segment display)

Input File Items

I[n]:e[/b]	n represents the file number and is optional. If specified, it must be 1 (one).
	e indicates the element number in the file (0 - 255).
	b specifies the bit (0 - 15 decimal). /b may be omitted if necessary to treat the I/O group as a numeric value.

Note: The elements in I/O modules are sequentially mapped into a memory table and are different from the item names in the controller programming software. MicroLogix and SLC500 adopt the same I/O addressing format. Refer to the Addressing SLC I/O Modulessection for details.

Examples:

I1:0/0

I:2/15

I:3 4BCD (for 16-bit thumbwheel input)

Status File Items

S[n]:e[/b]	n represents the file number and is optional. If specified, it must be 2 (two).
	e indicates the element number in the file.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).

Note: Refer to the SLC500 Family Processor Manual (Allen-Bradley Publication) for a complete description of the Status file information.

Examples:

S2:6 (major error fault)

S2:13 (math register)

S:1/5 (forces enabled)

Binary File Items

B[n]:e/b or B[n]/m	n represents the file number and is optional. If not specified, it is assumed to be 3 (three). If specified, the file number must be 3 or 9 - 255 decimal.
	e specifies the element (word) number within the Binary file. It must be 0 - 255 decimal.
	b specifies the bit number within the word. In the first form (where :e is present), the bit number must be 0 - 15 decimal.
	m also represents the bit number. However, in the second form, no word numbers are specified and the bit number may be 0 - 4095.

Examples:

B:33

B:6/4 (same bit as B/100)

B3/4095 (same bit as B:255/15)

Timer File Items

T[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 4 (four). If specified, the file number must be 4 (four) or 9 - 255 decimal.
	e specifies the element number (three words per element) within the Timer file. It must be 0 - 255 decimal.
	.f identifies one of the valid Timer fields. The valid fields for Timer Files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. All of the fields of a timer can be accessed by specifying the .f fields. However, it is possible to use /b to single out a bit in the .PRE or .ACC fields (which are words). The bit number must be 0 - 15 decimal.

Examples:

T4:0.ACC

T4:3.DN

T4:1.PRE

Counter File Items

C[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 5 (five). If specified, the file number must be 5 (five) or 9 - 255 decimal.
	e specifies the element number (three words per element) within the Counter file. It must be 0 - 255 decimal.
	.f identifies one of the valid Counter fields. The valid fields for the Counter Files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a counter. However, it is possible to use /b to single out a bit in the .PRE or .ACC fields (which are words). The bit number must be 0 - 15 decimal.

Examples:

C5:0.ACC

C5:3.OV

C5:1.PRE

Control File Items

R[n]:e[.f][/b]	n represents the file number and is optional. If not specified, it is assumed to be 6 (six). If specified, the file number must be 6 (six) or 9 - 255 decimal.
	e specifies the element number (three words per element) within the Control file. It must be 0 - 255 decimal.
	f identifies one of the valid Control fields. The valid fields for the Control files are listed in the table. If .f is omitted, it is assumed to be the word containing the status bits.
	b is optional and is normally not used. All of the fields of a Control file can be accessed by specifying the .f fields. However, it is possible to use /b to single out a bit in the .LEN or .POS fields (which are words). The bit number must be 0 - 15 decimal.

Examples:

R6:0.LEN

R6:3.EN

R6:1.POS

Integer File Items

N[n]:e[/b]	n represents the file number and is optional. If not specified, it is assumed to be 7 (seven). If specified, the file number must be 7 (seven) or 9 - 255 decimal.
	e specifies the element number within the Integer file. It must be 0 - 255 decimal.
	b is optional. If specified, it indicates the bit (0 - 15 decimal).

Examples:

N7:0

N7:0/15

N7:3

Floating Point File Items

F[n]:e	n represents the file number and is optional. If not specified, it is assumed to be 8 (eight). If specified, the file number must be 8 - 255 decimal.
	e specifies the element number within the Floating Point file. It must be 0 - 255 decimal.

Examples:

F8:0

F8:3

ASCII String Section Items

STn:e	n represents the file number (NOT optional) and must be 9 - 255 decimal.
	e specifies the element number within the String file. It must be 0 - 255 decimal. Each element in a String file contains an ASCII string with a maximum length of 82 characters.

Examples:

ST9:0

ST9:200

Long Integer Section Items

Ln:e[/b]	n represents the file number. If not specified, it is assumed to be 0 (zero). If specified, the file number must be 0 - 255 decimal.
	e specifies the element number within the Long Integer file. It must be 0 - 255 decimal.
	b is optional. If specified, it indicates the bit (0 - 31 decimal).

Examples:

L15:3

PID Section Items

PDn:e[.f][/b]	n represents the file number. If not specified, it is assumed to be 0 (zero). If specified, the file number must be 0 - 255 decimal.
	e specifies the element number within the PID file. It must be 0 - 255 decimal.
	.f identifies one of the valid PID fields. The valid fields for PID files are listed in the table.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a PID. If specified, it indicates the bit (0 - 15 decimal).

Important: Access to PID files may degrade the Communication Driver's performance, due to the extreme size of the PID element (23 words each). If accessing only a few PIDs at one time, performance will not be affected greatly. However, if accessing a few fields of many PIDs at once, it may be faster to move the needed fields to an intermediate file (Floating Point or Binary) and let the Communication Driver access the intermediate files.

Examples:

PD:0.SP

PD9:3.LUT

PD1:0.CVP

Message Section Items

MGn:e[.f] [/b]	n represents the file number. If not specified, it is assumed to be 0 (zero). If specified, the file number must be 0 - 255 decimal.
	e specifies the element number within the String file. It must be 0 - 255 decimal.
	.f identifies one of the valid MSG fields. The valid fields for MSG files are listed in the table.
	b is optional and is normally not used. Specifying the .f fields can access all of the fields of a timer. However, it is possible to use /b to single out a bit in the .PRE or .ACC fields (which are words). For Timer files, the bit number must be 0 - 15 decimal.

Important: Access to MSG files may degrade the Communication Driver's performance, due to the extreme size of the MSG file element (56 words each). If accessing only a few MSG elements at one time, performance will not be affected greatly. However, if accessing a few fields of many MSG file elements at once, it may be faster to move the needed fields to an intermediate file (Binary or Integer) and let the Communication Driver access the intermediate files.

Examples:

MG9:0.NOD

MG255:1.ELE

Communication Driver-Specific System Item

The system items described in this section refers to specific information regarding the Communication Driver, OI Server Manager, and the controllers.

The following generic system items are supported for all Allen-Bradley controllers, unless otherwise noted.

System Item Name	**Type/ Access Rights**	**Description**	**Values**
$SYS$DeviceStatus	String/Read	Status of the processor.	RANGE: OK or faulted
$SYS$Mode	String/Read	Current mode of the processor.	RANGE: Run, Program, Remote Run, or Remote Program
$SYS$PLCType	String/Read	Name of the process type.	Descriptive text for the process type.
$SYS$ProcessorName	String/Read	Name of the program running in the processor.	Descriptive text for the corresponding processor name.
$SYS$Revision	String/Read	Firmware of the processor.	Descriptive text for the firmware revision.

System Item Name	Type/ Access Rights	Description	Values
$SYS$UpdateTagInfo	Boolean/ ReadWrite	Force update of the whole controller tag database. The OI Server returns WriteComplete for $SYS$UpdateTagInfo when poked. The transaction will be completed with no timeout. **Note:** The value in $SYS$Update TagInfo will return to "0" from "1" after the process is finished. **Note:** The OI Server will implement manual and automated updates of the ControlLogix tag database in the event that you add or delete items by direct controller programming. **Warning! Updating a tag database online consumes resources. During the updating process, the** Communication Driver **may be held up from updating the client application.**	RANGE: On or Off

The following tag-database-specific system items are supported for all Allen-Bradley controllers, unless otherwise noted.

System Item Name	Type/ Access Rights	Description	Values
$SYS$BrowseTags	Boolean/ ReadWrite	Indicates whether the controller tags are browsable from OPC client. If a TRUE value is written to this item, the controller tags will become browsable, provided that the tag database is ready at the time. If a FALSE value is written to this item, all controller tags will not be browsable.	RANGE: True or false

System Item Name	Type/ Access Rights	Description	Values
$SYS$TagDBStatus	String/ Read-Only	Indicates the status of the tag database as follows:	
		Uninitialized – The tag database is uninitialized, typically in a start state.	Uninitialized
		Uploading – The tag database is being uploaded from the controller	Uploading
		Uploaded – The tag database has been completely uploaded from the controller.	Uploaded
		Error – The tag database is not uploaded because of errors encountered during the upload	Error
$SYS$TagDBVersion	String/ Read-Only	Indicates the version of the Tag database. String format: MajorVersion. MinorVersion If version information cannot be acquired (for example, due to a bad PLC connection) the value initially displays "Uninitialized" as a string.	RANGE: Major version: 0...65535 (no padding) Minor version: 0...999 (no padding)
$SYS$UpdateTagInfo	Boolean/ ReadWrite	Forces update of the controller tag database upon adding the next item for advise or poking any value to an existing item (On or Off).	RANGE: True or false

The following Logix5000 system items are supported for ControlLogix, CompactLogix, and FlexLogix processors, unless otherwise noted.

System Item Name	Type/ Access Rights	Description	Values
$SYS$DeviceSecurity	Boolean/ Read-Only	Status of controller security	RANGE: True or False

System Item Name	Type/ Access Rights	Description	Values
$SYS$Optimization	Boolean/ Read-Only	Indicates the status of ControlLogix message optimization in handle mode as enabled with the ControlLogix Optimization check box under the Logix5000 node editor in OI Server Manager (True or False). If: FALSE - no message optimization will be used. TRUE - either the 'Optimized for read' option or the 'Optimized for startup' is being used.	RANGE: True or False
$SYS$UDTOptimization	Boolean/ Read-Only	Indicates the status of the ControlLogix user-defined data type optimization enabled with the User Defined Data Type Optimization check box under the Logix5000 node editor in the OI Server Manager (On or Off).	RANGE: True or False
$SYS$FreeMem	DWord/ Read-Only	Returns the current unused memory, in number of bytes, in the Logix processor (I/O + data table + general). This item is not supported on ControlLogix L8x controllers. For L8x controllers, quality will be 0x07 (CONFIG_ERROR and LIMIT_CONSTANT).	RANGE: 0…2147483647
$SYS$FreeMemDT	DWord/ Read-Only	Returns the unused data table memory in number of bytes. This item is not applicable to 1756-L1, and is not supported on ControlLogix L8x controllers. For L8x controllers, quality will be 0x07 (CONFIG_ERROR and LIMIT_CONSTANT).	RANGE: 0…2147483647
$SYS$FreeMemGM	DWord/ Read-Only	Returns the total available general memory in number of bytes. This item is applicable to 1756-L55M16 only, and is not supported on ControlLogix L8x controllers. For L8x	RANGE: 0…2147483647

System Item Name	Type/ Access Rights	Description	Values
		controllers, quality will be 0x07 (CONFIG_ERROR and LIMIT_CONSTANT).	
$SYS$FreeMemIO	DWord/ Read-Only	Returns the total available I/O memory in number of bytes. This item is not supported on ControlLogix L8x controllers	RANGE: 0…2147483647
$SYS$TotalMem	DWord/ Read-Only	Returns the total memory, in number of bytes, in the Logix processor. This item is not supported on ControlLogix L8x controllers	RANGE: 0…2147483647
$SYS$TotalMemDT	DWord/ Read-Only	Returns the total available data table memory in number of bytes. This item is not applicable to 1756-L1, and is not supported on ControlLogix L8x controllers.	RANGE: 0…2147483647
$SYS$TotalMemGM	DWord/ Read-Only	Returns the total available general memory in number of bytes. This item is applicable to 1756-L55M16 only, and is not supported on ControlLogix L8x controllers.	RANGE: 0…2147483647
$SYS$TotalMemIO	DWord/ Read-Only	Returns the total available I/O memory in n umber of bytes. This item is not supported on ControlLogix L8x controllers	RANGE: 0…2147483647

The following system items are supported by each communications node in the ABCIP Communication Driver, unless otherwise noted.

System Item Name	Type/ Access Rights	Description	Values
$SYS$OpenConnections	DWord/ Read-Only	Returns the number of open CIP connections.	RANGE: 0…2147483647
$SYS$ConnectionsInitiated	DWord/ Read-Only	Returns the number of CIP connections initiated by the server.	RANGE: 0…2147483647

System Item Name	Type/ Access Rights	Description	Values
$SYS$ConnectionsRefused	DWord/ Read-Only	Returns the number of CIP connections refused by the communications module.	RANGE: 0…2147483647
$SYS$RequestSent	DWord/ Read-Only	Returns the number of message requests originating from the communications module.	RANGE: 0…2147483647
$SYS$ReplyReceived	DWord/ Read-Only	Returns the number of reply packets received.	RANGE: 0…2147483647
$SYS$UnsolReceived	DWord/ Read-Only	Returns the number of unsolicited messages received by the communications module.	RANGE: 0…2147483647
$SYS$Unsolreplied	DWord/ Read-Only	Returns the number of replies sent in response to the unsolicited message.	RANGE: 0…2147483647
$SYS$RequestErrors	DWord/ Read-Only	Returns the number of errors for the requests sent.	RANGE: 0…2147483647
$SYS$RequestTimeout	DWord/ Read-Only	Returns the number of request timed out	RANGE: 0…2147483647
$SYS$ResetStatistics	Boolean/ Write-Only	The item $SYS$ResetStatistics is available at root hierarchy PORT_CIP and will reset statistic counters of all child nodes.	RANGE: True or False
$SYS$TotalPacketSent	DWord/ Read-Only	Returns the number of data packets sent.	RANGE: 0…2147483647
$SYS$TotalPacketReceived	DWord/ Read-Only	Returns the number of replies received.	RANGE: 0…2147483647
$SYS$RateSent	DWord/ Read-Only	Returns the number of packets sent per second	RANGE: 0…2147483647

System Item Name	Type/ Access Rights	Description	Values
$SYS$RateReceived	DWord/ Read-Only	Returns the number of packets received per second.	RANGE: 0…2147483647
$SYS$ItemUpdateRate	DWord/ Read-Only	Returns the number of read items received per second.	RANGE: 0…2147483647
$SYS$ItemWriteRate	DWord/ Read-Only	Returns the number of write items sent out per second.	RANGE: 0…2147483647

Important: The Redundant Hierarchy, including the Device Group, is not hot-configurable, and requires a Reset on the Redundant Hierarchy to effect a configuration change.

Generic OPC Syntax

Communication Driver serves as a container for the OPC Groups, which provide the mechanism for containing and logically organizing OPC items. Within each OPC Group, an OPC-compliant client can register OPC items, which represent connections to data sources in the field device. In other words, all access to OPC items is maintained through the OPC Group.

The fully qualified name for an OPC item is called the Item ID (equivalent to the union of Link Name and Item Name). The syntax for specifying a unique Item ID is OI Server-dependent. In OPC data acquisition Communication Drivers, the syntax can be as follows:

```
AREA10.VESSEL1.TIC1.PLC.N7:11
```

where each component (delimited by a hint, that is, a period in case of Communication Driver) represents a branch or leaf of the field device's hierarchy.

In this example:

- AREA10.VESSEL1.TIC1 is the link name for Communication Driver.
- PLC is the name of the target controller.
- N7:11 is the specific data point (Item) desired.
- An item is typically a single value such as an analog, digital, or string value.

Where Item ID describes the syntax for defining the desired data point, OPC provides for another parameter, called Access Path, that defines optional specifications for obtaining that data.

In Communication Drivers, Access Paths are equivalent to Device Groups; it is this parameter that is used to define the update interval between the Communication Driver and the field device for accessing the values of data points in the controller.

Chapter 5

Troubleshooting the ABCIP Communication Driver

Troubleshooting with Windows Tools

Windows has two tools that may be useful in troubleshooting performance problems.

You can find quick verification that the Communication Driver process is running by looking at the Windows Task Manager. It also provides information on the user, CPU, and memory usage of the processes.

If you need more information, or need to gather data while not logged in, you can use the Performance and Alerts application. For more information, see the Microsoft Management Console (MMC) help files on the Performance application. The Performance application is one of the administrative tools found in the Windows Control Panel.

Troubleshooting with the OI Server Manager

The OI Server Manager has information that may be useful in troubleshooting problems. When the Communication Driver is active, a diagnostic node is present below the configuration node in the console tree of the Operations Control Management Console.

Each diagnostic leaf contains information about Communication Driver activity. For more information, see the AVEVA™ Communication Drivers Pack – Core Help.

Finding Version Information

If you contact Technical Support, you may need to supply version information.

To determine the OI Server Manager version

- In the OI Server Manager, right-click **OI Server Manager**, and then click **About OI Server Manager**. An **About** box appears showing the version and build date of the OI Server Manager.

To determine version information for Communication Driver components

- In the OI Server Manager, select the Communication Driver name in the console tree. The version information for each Communication Driver component is shown in the details pane.

Using the Log Viewer

Error messages are created by the Communication Driver and logged by the Logger. You can view these messages with the Log Viewer. The Log Viewer help files explain how to view messages and how to filter which messages are shown.

Log Flags are categories of messages. The Log Flag Editor User Guide contains an explanation of the categories. Using the Log Flag Editor, you can specify which log flags the Communication Driver creates.

Note: Generating large numbers of diagnostic messages can impact Communication Driver performance. You should not run in production with any more flags than those set when the Communication Driver is installed. To troubleshoot you can turn on more flags, but there is a performance impact. For more information, see the Log Flag Editor User Guide.

To open the Log Flag Editor

1. In the Operations Control Management Console, expand **Log Viewer** and then expand the log viewer group.
2. Select **Local**.
3. On the **Action** menu, click **Log Flags**.

In general, look at error and warning messages to determine if a problem exists. To determine whether the Communication Driver is communicating with a device, you can enable the DASSend and DASReceive log flags. From these you can determine whether or not the device is responding.

Basic Log Flags

The basic log flags for all AVEVA components are:

- Error: A fatal error, the program cannot continue. By default set on by logger.
- Warning: The error is recoverable. A client called with a bad parameter, or the result of some operation was incorrect, but the program can continue. By default set on by logger.
- Start-Stop: Each main component logs a message to this category as it starts and stops.
- Info: General diagnostic messages.
- Ctor-Dtor: C++ classes of interest log messages to this category as they are constructed and destructed.
- Entry-Exit: Functions of interest log messages to this category as they are called and return.
- Thread Start-Stop: All threads should log messages to this category as they start and stop.

Communication Driver Log Flags

Messages created for these log flags are for Communication Driver common components and contain information about internal Communication Driver activities.

- DACmnProtFail: Some failure occurred in the common components while sending a message, updating an item, or otherwise moving data. Typically, this represents some unexpected behavior in the server-specific DLL.
- DACmnProtWarn: Some problem occurred that interfered with sending messages, updating items, or otherwise moving data. Common examples are slow poll, value limiting during type conversion, and transaction timeout messages.
- DACmnTrace: Normal processing of client program requests and data movement to and from the server-specific DLL are traced on this log flag. Use this in conjunction with DACmnVerbose to get the most information.
- DACmnVerbose: Many log flags used by the DAS common components are modified occasionally by DACmnVerbose. When DACmnVerbose is set, the logging of messages on other log flags includes more information.
- DACmnSend: Operations within the DAS Engine DLL that revolve around sending messages to the server-specific DLL.
- DACmnReceive: Events surrounding messages that are returned to the DAS Engine by the server-specific DLL, including the blocking and unblocking of hierarchies.

Communication Driver-Device Interface Log Flags

Messages created for the following log flags are specific to an individual Communication Driver and contain information about communications between the Communication Driver and device.

- DASProtFail: An error in the protocol occurred, for example, device disconnected. The program can continue, and, in fact, this category is expected during normal operation of the program. Must be set on by the generic DAS code when the Communication Driver starts.
- DASProtWarn: Something unexpected occurred in the protocol, for example, a requested item with an otherwise valid item name is not supported by this device. Must be set on by the generic DAS code when the Communication Driver starts.
- DASStateCat1: General diagnostic messages of a protocol-specific nature. For example, you can provide the number of items in a message for a specific protocol, then optimize based on the number.
- DASVerbose: Modifies all other DAS logging flags. When on, provides detailed messages.
- DASSend: Protocol messages sent to the device are logged to this category.
- DASReceive: Protocol messages received from the device are logged to this category.
- DASStateCat1, DASStateCat2, DASStateCat3, DASStateCat4: These are general categories for use by the server developer. As DeviceEngine-generated state machines are created by the Communication Driver, they can be told to log state machine messages to one of the following: DASStateCat1, DASStateCat2, DASStateCat3, or DASStateCat4. These messages indicate when a state is made the active state, when a state handler is run, when a state handler completes, and when a timeout occurs for a state machine.
- DASStateMachine: By default, DeviceEngine-generated state machines created by the Communication Driver log to this category unless specifically told to log to one of the DASStateCatN categories. In addition, general

state machine messages are logged to this category. These messages indicate when a state machine is created and deleted.

Error Codes and Error Messages

This section describes the various error codes and error messages that might appear when working with the ABCIP Communication Driver.

ABCIP Communication Driver Error Codes

There are two server-specific error codes, shown in the following table, that augment those provided by the Operations Integration Toolkit.

Error Code	Logger Message	Log Flag
-10001	PLC not connected	DASProtFail
-10002	PLC times out	DASProtFail

Logix5000 Error Codes

The Logix5000 processor generates error conditions. The following tables show these errors and the server-specific strings generated by the Communication Driver to the logger.

Note: All of the error messages shown in the following table apply to the **DASProtFail l**og flag.

General Allen-Bradley Error Code (High byte = 00)	Logger Message
00	Success
01	Connection failed
02	Insufficient Connection Manager resources
03	Invalid connection number
04	IOI could not be deciphered. Either it was not formed correctly or the match tag does not exist
05	The particular item referenced could not be found
06	The amount of data requested would not fit into the response buffer. Partial data transfer has occurred.
07	Connection has been lost

General Allen-Bradley Error Code (High byte = 00)	**Logger Message**
08	Requested service not supported
09	Error in data segment or invalid attribute number
0A	An error has occurred trying to process one of the attributes
0C	Service cannot be performed while object is in current state
10	Service cannot be performed while device is in current state
11	Response data too large
13	Not enough command data/parameters were supplied in the command to execute the service requested
14	Attribute not supported
15	Too much data was received
1C	An insufficient number of attributes were provided compared to the attribute count
1E	Errors encountered with the items in the message
26	The IOI word length did not match the amount of IOI which was processed
None of the above codes	Unknown Status

Extended Allen-Bradley Error Code (Hex)	**Logger Message**
2104	The beginning offset was beyond the end of the template.
2105	You have tried to access beyond the end of the data object.
2106	Data in use.
2107	The abbreviated type does not match the data type of the data object.

Extended Allen-Bradley Error Code (Hex)	**Logger Message**
0100	Connection in Use or Duplicate Forward Open.
0103	Transport Class and Trigger combination not supported.
0106	Ownership Conflict.
0107	Connection not found at target application.
0108	Invalid Connection Type. Indicates a problem with either the Connection Type or Priority of the Connection.
0109	Invalid Connection Size.
0110	Device not configured
0111	RPI not supported. May also indicate problem with connection time-out multiplier.
0113	Connection Manager cannot support any more connections.
0114	Either the Vendor ID or the Product Code in the key segment did not match the device.
0115	Product Type in the key segment did not match the device.
0116	Major or Minor Revision information in the key segment did not match the device.
0117	Invalid Connection Point.
0118	Invalid Configuration Format.
0119	Connection request fails since there is no controlling connection currently open.
011A	Target Application cannot support any more connections.
011B	RPI is smaller than the Production Inhibit Time.
0203	Connection cannot be closed since the connection has timed out.

Extended Allen-Bradley Error Code (Hex)	**Logger Message**
0204	Unconnected Send timed out waiting for a response.
0205	Parameter Error in Unconnected Send Service.
0206	Message too large for Unconnected message service.
0207	Unconnected acknowledge without reply.
0301	No buffer memory available.
0302	Network Bandwidth not available for data.
0303	No Tag filters available.
0304	Not Configured to send real-time data.
0311	Port specified in Port Segment Not Available.
0312	Link Address specified in Port Segment Not Available.
0315	Invalid Segment Type or Segment Value in Path.
0316	Path and Connection not equal in close.
0317	Either Segment not present or Encoded Value in Network Segment is invalid.
0318	Link Address to Self Invalid.
0319	Resources on Secondary Unavailable.
031A	Connection already established.
031B	Direct connection already established.
031C	Miscellaneous.
031D	Redundant connection mismatch.
None of the above codes	Unknown Extended Status.

Note: For more information about the general and extended Allen-Bradley error codes, please refer to the Allen-Bradley controller documentation.

Data Highway Plus Error Conditions

The Data Highway Plus generates error conditions. These error conditions and the server-specific strings are generated by the Communication Driver to the logger.

Note: All of the error messages shown in the following table apply to the **DASProtFail l**og flag.

Allen Bradley Error Code	**Logger Message**
DHPERR_DP_FNC (0x20000001)	Dual-port memory functionality test error
DHPERR_RAM (0x20000002)	Unknown random access memory test error
DHPERR_RAM (0x20000003)	Failure of Z80 RAM 0
DHPERR_RAM (0x20000004)	Failure of dual-port RAM
DHPERR_RAM (0x20000005)	Failure of Z80 RAM 1
DHPERR_RAM01 (0x20000006)	Failure of both Z80 RAM 0 and RAM 1
DHPERR_RAM1_DP (0x20000007)	Failure of both RAM 1 and Dual-Port RAM
DHPERR_CTC (0x20000008)	Unknown counter timer circuit test error
DHPERR_CTC_TMR (0x20000009)	Failure of CTC timer module
DHPERR_CTC_CNT (0x2000000A)	Failure of CTC counter module
DHPERR_CTC_TC (0x2000000B)	Failure of both CTC timer and counter modules
DHPERR_SIO (0x2000000C)	Unknown serial input output test error
DHPERR_SIO_INT (0x2000000D)	Failure of CIO channel: no interrupt
DHPERR_SIO_LOOP (0x2000000E)	Failure of SIO channel A: Loopback failure
DHPERR_PROT_LOAD (0x2000000F)	Protocol file download error
DHPERR_LOAD_BLK (0x20000010)	Block too large error
DHPERR_RAM_FULL (0x20000011)	Z80 RAM too full for next block
DHPERR_BD_WRITE (0x20000012)	Cannot write to adapter card memory
DHPERR_OPEN_LOADPCL (0x20000013)	Cannot open file LOADPCL.BIN

Allen Bradley Error Code	Logger Message
DHPERR_OPEN_KLPCL (0x20000014)	Cannot open file KLPCL.BIN
DHPERR_OPEN_KLST0 (0x20000015)	Cannot open file KLST0.BIN
DHPERR_OPEN_KLST1 (0x20000016)	Cannot open file KLST1.BIN
DHPERR_OPEN_KLST2 (0x20000017)	Cannot open file KLST2.BIN
DHPERR_OPEN_PROT (0x20000018)	Cannot open protocol file
TIMEOUT_ERR (0x01)	Timeout error
CANCELLED_ERR (0x02)	Cancelled error code

ABCIP Communication Driver Error Messages

The following table lists the error messages produced by the Communication Driver that are logged to the Log Viewer.

- <Message ID> corresponds to the message ID displayed in the Communication Driver's Diagnostics root in the OI Server Manager.
- <Device> refers to the node name of the device.

Note: All of the error messages shown in the following table apply to the **DASProtFail** log flag.

Error Message	Explanation	Probable Cause	Solution
A PLC (IP:<IPAddress>) attempted to send us an unsolicited data packet. But the maximum number of simultaneous unsolicited data connections [MAX socket] has already been reached. Data packet ignored.	The maximum number of sockets used for unsolicited data communications has been reached. No more unsolicited data packages will be accepted.	The maximum number of sockets used for unsolicited data communications has been reached.	Decrease the number of unsolicited data to communicate to the socket.
ABCIPAcceptedSocket: Initialize unable to associate an event with a handle	Unable to associate the event with a valid handle within the internal state computer.	Software internal error.	Restart the Communication Driver and try again.

Error Message	Explanation	Probable Cause	Solution
Array index error found while formatting message for block <Block Number>	The Bit number specified in the item is out of range	Bit number is out of range	Verify and specify the correct bit number appropriate for the data type of the item tag.
Attempt to resolve remote hostname <HostName> failed	Failed to resolve the HostName.	The HostName cannot be translated to a valid IP address internally.	Check if the HostName is configured correctly in the server.
Cannot create optimize structure for item <Item Name> message <MessageID>	The server failed to create internal structure.	It is an internal error.	
Connection to <Target Address> on port <Port Number> failed with error <Error Code>.	Error is returned from the OS while trying to establish the socket connection.	Indicated by OS returned <Error Code>.	
Connection to <Target Address> on port <Port Number> refused.	No connection can be made because the target device actively refused it.	The target address is not a ControlLogix Ethernet Module.	Check the device configured with the address.
Encountered the following error in reply message <Message ID> when reading from <Device>	Error codes are returned in the poll-message response from the device; further explanations follow.	Depends on the CIP errors returned (refer to the tables in Logix5000 Error Codes).	Check to see if there are other error messages in the logger. Check the Communication Driver diagnostics, if necessary.
Encountered the following error in reply message <Message ID> when writing to <Device>	Error codes are returned in the poke-message response from the device; further explanation follows.	Depends on the CIP errors returned (refer to the tables in Logix5000 Error Codes).	Check to see if there are other error messages in the logger. Check the Communication Driver diagnostics, if necessary.
Encountered the following error when reading block <Block Item Name> in message <Message ID> from <Device>	Error codes are returned in the request block for Multi-Request messages. Further explanation of the errors will be listed.	Depends on the CIP errors returned (refer to the tables in Logix5000 Error Codes).	Check to see if there are other error messages in the logger. Check the Communication Driver diagnostics, if necessary.

Error Message	Explanation	Probable Cause	Solution
Encountered the following error when reading optimized block <Internal Block Address> in message <Message ID> from <Node>	The error code is returned from the ControlLogix controller when the server tries to read the optimization structure in the controller.	This is an internal error. See the CIP Service error code for details.	
Encountered the following error when writing block <Block Item Name> in message <Message ID> from <Device>	Error codes are returned in the request block for Multi-Request messages. Further explanation of the errors will be listed.	Depends on the CIP errors returned (refer to the tables in Logix5000 Error Codes).	Check to see if there are other error messages in the logger. Check the Communication Driver diagnostics, if necessary.
Error encountered initializing Unsolicited Data Port. No direct (i.e.: peer-to-peer) unsolicited data will be accepted.	There is a failure to create a listening socket for the peer-to-peer unsolicited data used. As a result, no unsolicited data can be accepted.	Another application has already been listening at the same port. The network communications is having a problem. The controller is having a problem communicating.	Make sure that no other application is running and listening at the same port (such as RSLinx). Make sure the network is functioning. Make sure the controller is functioning.
ExtSTS=<Extended Error Code>: <Description>	The error message shows the extended CIP error code and description, if there is one.	CIP-error dependent (refer to the tables in Logix5000 Error Codes).	
Failed to add block <block number> with base name <UDT base name> in <PLC address>, tag does not exist in the PLC or some of its UDT members are configured for External Access=None in the PLC	The error message shows when advising a UDT member with external access set to NONE or the PLC tag does not exist.	Either the tag does not exist in the PLC or one or more UDT members are configured for an External Access attribute of "None".	Set the UDT member external access from NONE to Read Only or Read/Write, or create the tag.

Error Message	Explanation	Probable Cause	Solution
Failed to initialize Listen Socket (CIP port = <Port Number>)	Listening socket with the indicated port number is being used by another process. ABCIP Communication Driver cannot receive unsolicited message from controllers.	There may be third party products already listening on the same CIP port.	Shut down any third-party product (such as RSLinx) listening on the same CIP port and restart ABCIP Communication Driver.
Host EtherNet/IP <IP address> connect host failed, maximum number of socket <MAX socket> exceeded	The maximum number of sockets allowed was exceeded.	The maximum number of sockets allowed was exceeded.	System limit on TCP sockets on the machine hosting the Communication Driver is reached. Check if there are other programs on same the machine consuming a large number of sockets.
Inconsistent message type encountered for <Device Name>	The Communication Driver encounters an internal error.	This is an internal error.	
Invalid item <Item Name> fields required for structure item	UDT member was not defined in the item syntax	Invalid Item syntax	Specify the member of the UDT structure in the item tag
Invalid item <Item Name> bit number not allowed or invalid	The Bit number specified in the item is out of range or invalid	The bit number is specified for a non-integer type.	Verify and specify the correct bit number appropriate for the data type of the item tag.
Invalid item <Item Name> structure not found	Specified UDT structure does not exist in the controller	Invalid item syntax	Verify if the UDT structure name is correct and exists in the controller.
Invalid item <Item Name>, invalid item format <format>	"," was used to specify the item's format. However, the format was found to be invalid for the item.	An incorrect format was used for the item.	Check the item's format.
Invalid item <Item Name>, invalid index	An invalid index is specified in the item.	An invalid index is found in the item.	Check the index specified. Only integer is accepted as an index.

Error Message	Explanation	Probable Cause	Solution
Invalid item <Item Name>, member <Member Name> not found in structure	The member of the structure cannot be found.	The member is not defined in the structure.	Check the structure's definition in the device.
Invalid item <Item Name>, not defined in the processor	The item's definition cannot be found in the tag database.	The item is not defined in the Logix processor.	Check the item's definition in the device.
Invalid item <Item Name>, offset dimension exceeded	The internal limit of the item's nesting level (20) was exceeded. In general, each "." and the index [x] increment the item's nesting level by 1.	The item's nesting level exceeds the server's limit.	Reduce the item's nesting level.
Invalid item <Item Name>, retrieving whole structure not supported	The item points to a structure other than a string. The Communication Driver does not support this type of item.	A structure item is specified.	Retrieve an individual member of the structure instead of the whole structure.
Invalid item <Item Name>, structure not found	The structure definition for the item cannot be located.	Invalid item encountered.	Check the item's definition in the device.
Invalid item <Item Name>, syntax error	There is a syntax error in the item.	An invalid item is specified.	Check the item's syntax.
Invalid item <Item Name>, unknown suffix	The suffix specified after "," is not recognized by the Communication Driver.	The suffix is not supported.	Check the suffix specified for the item.
Invalid item <Item Name>, dimension mismatch	The dimension specified in the <Item Name> is different from what has been defined in the device.	The item's array dimension is different from the definition in the device.	Check the item's definition in the device.
Invalid item <Item Name>, index bracket mismatch	The bracket for the index is mismatched. This is an item syntax error.	An invalid item is entered.	Correct the item's syntax.
Invalid item <Item Name>, index out of range	The index specified in the <Item Name> is	The item's index is too large.	Check the item's definition in the device.

Error Message	Explanation	Probable Cause	Solution
	outside the range defined in the device.		
Invalid item <Item Name>, invalid bit number	The bit number specified in the item is invalid.	An invalid item is specified.	The bit number specified cannot go beyond the range allowed for the item's data type. For example, INT range is 0-15 and DINT range is 0-31.
Item <ItemName> cannot be created, out of memory.	ABCIP Communication Driver failed to obtain memory during item creation.	The system ran out of memory.	Reduce the number of tags in the Communication Driver. Close other applications.
Message <Message ID> for <Device> timed out.	The Communication Driver did not receive the message's response from the device within the <Reply Timeout> specified.	The device is off line.	Check the device's network connection.
Mismatched bracket found while formatting message for block <Block Number>	Invalid item syntax	Missing bracket in the item tag	Specify the missing bracket in the item tag.
Received packet from [HostName] too big on port [PortNumber] ([#of bytes received] bytes)	The received packet from the controller exceeds the maximum packet size allowed for this type of protocol.	Incorrect data packet was read from the socket.	Check if the server is configured properly for the target ControlLogix controller.
Received incomplete response packet for message <Message ID> from <Device>	The response packet is incomplete or corrupted.	Bad connection or there is a Communication Driver problem.	Check if there are other error messages in the logger. Check the OI Server diagnostics if necessary.
recv() for <HostName> on port <PortNumber> failed	Failed to read from the Window Socket specified.	Failed to read from the Winsock.	Repeat the operation by restarting the Communication Driver.

Error Message	Explanation	Probable Cause	Solution
Register Session encountered the following error: recd packet from [HostName] too big on port [PortNumber] ([#of bytes received] bytes)	The Communication Driver encountered an error while trying to establish an EtherNet/IP session with the controller. The received packet from the controller exceeds the maximum packet size allowed for this type of protocol.	Not communicating with a ControlLogix controller. Incorrect data packet was read from the socket.	Check the controller configuration. Check if the server is configured properly for the target ControlLogix controller.
Rejected %s ITEM = %s on plc %s	The item cannot be added or subscribed from the OI Server. The time tag portion in the &T& syntax is missing or invalid.	The time tag portion in the &T& syntax is missing or invalid.	If the &T& syntax is used, make sure that it consists of a valid data tag followed by the "&T&" and a valid time tag.
Rejected <PLC Type Name> ITEM = <Item Name> on plc <PLC Node Name>. Time Tags not supported on Firmware Revision less than 16.0	The &T& time tag syntax cannot be used with controller firmware prior to version 16.0.	Controller firmware version is not compatible with &T& time tag requirement.	Upgrade the controller firmware version to 16 or above if the &T& time tag syntax is to used.
Rejected <PLC Type Name> ITEM = <Item Name> on plc <PLC Node Name>. Invalid data type for time tag	The data type of the time tag is not correct	The data type of the time tag is not correct	The data type for time tag must be LINT
Response service code <ServiceCode> different from command <Service Code> for message <Message ID>	The service code in the message sent does not match the one in the reply.	Packet corrupted or OI Server problem encountered.	Check if there are other error messages in the logger. Check the Communication Driver diagnostics, if necessary.
Response service code <ServiceCode> not handled	Unexpected service code was encountered in the reply packet from the controller.	This is an internal error.	The service code received by the Communication Driver is not supported. Verify the controller firmware version against version supported by the Communication Driver.

Error Message	Explanation	Probable Cause	Solution
Session error <Error Code>, packet ignored	Communications errors encountered.	CIP-error dependent (refer to the tables in Logix5000 Error Codes).	
Socket <SocketID> send() returned <ErrorNumber>, connection to be closed	This is an internal Winsock error.	This is an internal Winsock error.	
Socket <SocketID> sending packet with buffer size <BufferSize> larger than <MAX buffer size>	Failed to send on socket due to the data packet size.	This is an internal error.	
STS=<CIP Error Code>: <Description>	The error message lists the CIP error code and description.	CIP-error dependent (refer to the tables in Logix5000 Error Codes).	
Timeout waiting for an unknown event from PLC on an unsolicited data port connected to <PLC Host Name>	Timeout occurred while waiting for unsolicited data packet from a controller.	Failed to receive unsolicited data from a controller.	Make sure the controller is configured to send out unsolicited data correctly. Make sure the Communication Driver is functioning correctly.
Timeout waiting for data packet from PLC on an unsolicited data port connected to <PLC Host Name>	Timeout occurred while waiting for unsolicited data packet from a controller.	Failed to receive unsolicited data from a controller.	Make sure the controller is configured to send out unsolicited data correctly. Make sure the Communication Driver is functioning correctly.
Timeout waiting for initialization packet from PLC on an unsolicited data port connected to <PLC Host Name>	Timeout occurred while waiting for unsolicited data header from a controller.	Failed to receive unsolicited data from a controller.	Make sure the controller is configured to send out unsolicited data correctly. Make sure the Communication Driver is functioning correctly.
Unsolicited socket not open for device <Device>	Unsolicited socket was not open when the Communication Driver	This is an internal error.	Reset the node hierarchy to restart the connection

Error Message	Explanation	Probable Cause	Solution
	tried to send a reply message to the device.		

PLC-5 Error Messages

The error messages generated specifically for the PLC-5 family controllers are listed in the following table.

Error Message	Explanation	Possible Cause	Solution
BCD file number must be greater than 2	Incorrect format for the item. The BCD Item's File Number was smaller than 3 for PLC-5.	The BCD Item's File Number must be 3 or larger.	Only access the BCD Item with File Number equal to 3 or larger.
BINARY file number must be greater than 2	Incorrect format for this item. The Binary Item's File Number was smaller than 3 for PLC-5.	The Binary Item's File Number must be 3 or larger.	Only access the Binary Item with File Number equal to 3 or larger.
BT file number must be > 8	Incorrect format for the Item. The BT item's File Number was 8 or smaller for PLC-5.	The BT Item's File Number must be 9 or larger.	Only access the BT Item with File Number equal to 9 or larger.
Cannot write to file BT[FileNumber]	Failed to write to a BT item for PLC-5.	For PLC-5, write operation to a BT item is not permitted.	Do not attempt to write to a BT item for PLC-5.
CONTROL file number must be greater than 2	Incorrect format for the item. The Control Item's File Number was smaller than 3 for PLC-5.	The Control Item's File Number must be 3 or larger.	Only access the Control Item with File Number equal to 3 or larger.
INTEGER file number must be greater than 2	Incorrect format for the item. The Integer Item's File Number was smaller than 3 for PLC-5.	The Integer Item's File Number must be 3 or larger.	Only access the Integer Item with File Number equal to 3 or larger.

Error Message	Explanation	Possible Cause	Solution
item <ItemName> not valid, PLC does not have PID feature	PID feature is not supported for this PLC-5 configuration.	The PLC-5 configuration indicates that the PID feature is not supported.	Select the "Support PID" feature option for PLC-5, if the controller supports the feature.
PD file number must be > 8	Incorrect format for the item. The PID Item’s File Number was 8 or smaller for PLC-5.	The PID Item’s File Number must be 9 or larger.	Only access the PID Item with File Number equal to 9 or larger.
SC file number must be > 4	Incorrect format for the Item. The SC item’s File Number was 4 or smaller for PLC-5.	The ST Item’s File Number must be 5 or larger.	Only access the ST Item with File Number equal to 5 or larger.
ST file number must be > 8	Incorrect format for the Item. The ST item’s File Number was 8 or smaller for PLC-5.	The ST Item’s File Number must be 9 or larger.	Only access the ST Item with File Number equal to 9 or larger.
TIMER file number must be greater than 2	Incorrect format for the item. The Timer Item’s File Number was smaller than 3 for PLC-5.	The Timer Item’s File Number must be 3 or larger.	Only access the Timer Item with File Number equal to 3 or larger.

SLC500 and MicroLogix Error Messages

The following table lists all the SLC500 and MicroLogix controller-specific error messages.

Error Message	Explanation	Possible Cause	Solution
BINARY file number must be 3 or 9-255	Incorrect format for this item. The Binary Item’s File Number was not 3 or 9-255 for SLC500 and MicroLogix.	Binary Item’s File Number must be 3 or 9-255.	Only access Binary Item with valid File Number.
CONTROL file number must be greater than 6 or 9-255	Incorrect format for the item. The Control Item’s File Number was not 6 or 9-255.	The Control Item’s File Number must be 6 or 9-255.	Only access the Control Item with the valid File Number.

Error Message	Explanation	Possible Cause	Solution
COUNTER file number must be greater than 5 or 9-255	Incorrect format for the item. The Counter Item's File Number was not 5 or 9-255.	Counter Item's File Number must be 5 or 9-255.	Only access Counter Item with valid File Number.
FLOAT file number must be 8 or 8-255	Incorrect format for the item. The Float Item's File Number was not 8 or 8-255.	The Float Item's File Number must be 8 or 8-255.	Only access the Float Item with the valid File Number.
INTEGER file number must be 7 or 9-255	Incorrect format for the item. The Integer Item's File Number was not 7 or 9-255.	The Integer Item's File Number must be 7 or 9-255.	Only access the Integer Item with the valid File Number.
TIMER file number must be 4 or 9-255	Incorrect format for the item. The Timer Item's File Number was not 4 or 9-255.	Timer Item's File Number must be 4 or 9-255.	Only access Timer Item with valid File Number.

PLC-5, SLC500, and MicroLogix Error Messages

The error messages listed in the following table pertain to the PLC-5, SLC500, and MicroLogix controllers.

Error Message	Explanation	Possible Cause	Solution
[Sub-Element] not valid for type [FileType] files.	Incorrect format for the item. The Sub-Element is not valid for this File Type.	Wrong Item format with a wrong Sub-Element type.	Only access the valid item format with the correct Sub-Element type.
ASCII file number must be greater than 2	Incorrect format for the item. The ASCII Item's File Number was smaller than 3.	The ASCII Item's File Number must be 3 or larger.	Only access the ASCII Item with a File Number equaling to 3 or larger.
BINARY file, bit>15 and element>0	Incorrect format for the item. The Binary Item contained an element number, but its bit number was larger than 15.	The valid format for a PLC-5 Binary Item is: B[FileNumber]: [Element]/[Bit], where Bit is from 0 to 15. In this case, the Bit field was larger than 15.	Only access the Binary Item with the valid range.

Error Message	**Explanation**	**Possible Cause**	**Solution**
File numbers must be between 0 and 999	Incorrect format for the Item. The Item’s File Number was out of range.	A bad item File Number was used.	Use a valid range for the Item’s File Number.
FLOATING POINT file cannot have bit number	Incorrect format for the item. The Floating Point Item contained a bit number field.	The Floating Point Item must not contain a bit number field.	Only access the Floating Point Item without a bit number field.
FLOATING POINT file number must be greater than 2	Incorrect format for the item. The Floating Point Item’s File Number was smaller than 3.	The Floating Point Item’s File Number must be 3 or larger.	Only access the Floating Point Item with a File Number that equals to 3 or larger.
INPUT file number must be 1	Incorrect format for the item. The Input Item’s File Number was not 1.	The Input Item’s File Number must be 1.	Only access the Input Item with a File Number that equals to 1.
OUTPUT file number must be 0	Incorrect format for the item. The Output Item’s File Number was not 0.	The Output Item’s File Number must be 0.	Only access the Output Item with a File Number equaling to 0.
STATUS file number must be 2	Incorrect format for the item. The Status Item’s File Number was not 2.	The Status Item’s File Number must be 2.	Only access the Status Item with a File Number equaling to 2.
Unsupported file type [File Type]	Incorrect format for the item. An invalid Item Type was used.	There was no such Item Type name.	Use the valid Item Type.
[Sub-Element not valid for type [FileType] section.	Incorrect format for the item. The Sub-Element is not valid for this section.	Wrong Item format.	Use only the valid item format.

Error Message	**Explanation**	**Possible Cause**	**Solution**
Attempt to write read only item in file [FileNumber] element [Element#] subelement [Sub-Element#] ignored	Write operation failed due to an attempt to write to a read-only item.	An attempt to write to a read-only item caused the failure.	Do not attempt a write operation to a read-only item.