AVEVA™ Communication Drivers Pack – Siemens - SIDIRECT Driver

User Guide

Publication date: Tuesday, May 9, 2023

Publication ID: 868888

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

AVEVA

Chapter 1

Introduction to the SIDirect Communication Driver

About the SIDirect Communication Driver

The SIDirect Communication Driver is a Microsoft® Windows® application program that acts as a communications protocol server to provide access to data within the Siemens SIMATIC S7-1500 PLC series.

The Communication Driver provides access to a Siemens SIMATIC S7-1500 PLC through an off-the-shelf standard Ethernet network interface card in the computer. Multiple network interface cards can be supported by the Communication Driver.

The SIDirect Communication Driver connects via Ethernet to the S7-1500 series PLC. It does **not** require any of the following items:

- Siemens PC adapter cards
- Siemens CP (Communications Processor) cards that reside in the PC
- Siemens SIMATIC NET library

The SIDirect Communication Driver contains two objects that can be configured to interface with the Siemens S7-1500 PLC:

- Legacy object (see SIDirect Legacy Object: Non-Symbolic Namespace on page 8) – supports SIMATIC tag names.
- Symbolic object (see SIDirect Symbolic Object: Symbolic Namespace on page 8) – supports symbolic tag names.

While the SIDirect Communication Driver can support other PLCs in Siemens S7 family of PLCs, it is specifically designed to support the S7-1500 . For the complete list of supported hardware and firmware, see "Tested Communication Driver Hardware and Firmware".

SIDirect Legacy Object: Non-Symbolic Namespace

The SIDirect Legacy object provides an interface to support traditional SIMATIC tag names (for example, "`DB1,INT0`"). Use the Legacy object if you are using non-symbolic names (absolute addressing) to communicate with the PLC. To use absolute addressing, check the following settings in the S7_1500 PLC:

- PUT/GET communication must be enabled.
- Optimized block access must be disabled (non-optimized mode).

See SIDirect Legacy Object Pre-Configuration for additional information.

SIDirect Symbolic Object: Symbolic Namespace

The SIDirect Symbolic object provides an interface to support symbolic tag names (for example, "`Tank4.Lvl`"). Use the Symbolic object if you are using symbolic names to communicate with the PLC. Symbolic addressing works whether or not optimized block access has been enabled in the PLC.

Supported Client Protocols

The SIDirect Communication Driver uses the following different communications protocols to communicates with clients:

- OPC
- SuiteLink
- DDE/FastDDE

For more information refer to "Support Client Protocols" in the AVEVA Communication Drivers Core Help.

Supported Device Protocols

The SIDirect Communication Driver uses only the TCP bus communications protocols over the Ethernet to communicate with the Siemens S7-300/400/1200/1500 family of controllers. The SIDirect Communication Driver does not support MPI, Profibus, and other non-Ethernet protocols.

Licensing for SIDirect Communication Driver

The SIDirect Communication Driver supports the activation-based licensing to acquire the license both locally and remotely. For more information on activation-based licensing, see "Licensing" section in the Communication Drivers Pack Help.

Chapter 2

SIDirect Communication Driver Configuration

Determining the SIDirect Communication Driver Hierarchy

Each Communication Driver is identified by a unique program name (ProgID) under the OCMC. The ProgID for the SIDirect Communication Driver is: OI.SIDIR.1. You can find it in the local node of the default group of the OI Server Manager, on the computer where the Communication Driver is installed.

When you access the Communication Driver remotely, you will not find the Communication Driver node under the local node. You must locate and identify the Communication Driver on a computer in one of the node groups.

To find the Communication Driver

1. Start the Operations Control Management Console (From the Windows **Start** menu, point to **Programs**, **AVEVA**, and then click the **Operations Control Management Console** icon).
2. In the OI Server Manager tree, under the **Local** node, navigate to the SIDirect Communication Driver. For general information about working in this snap-in environment, see the Communication Drivers Pack Help.

To determine the SIDirect Communication Driver Hierarchy

Before configuring your Communication Driver, you should determine the hierarchical structure of your network/PLC environment to establish communications to each of the controllers. The SIDirect hierarchy in the Communication Driver starts with the TCP/IP PORT object, followed by supported SIDirect controllers.

1. Start the **Operations Control Management Console**.
2. In the OI Server Manager tree, under the **Local** node, navigate to the SIDirect Communication Driver.
3. Expand the Communication Driver, and then click **Configuration**.

   The **Global Parameters** tab appears in the details pane.

4. Configure all the global parameters as required for this Communication Drivers Pack Help.

   Note: Any global parameters fields that appear disabled are not supported for the Communication Driver.

5. Configure the PORT objects.
   - For step-by-step procedures on how to configure the Legacy Object, see Configuring the SIDirect Legacy Object.
   - For step-by-step procedures on how to configure the Symbolic Object, see Configuring the SIDirect Symbolic Object.
6. When the SIDirect hierarchy build is completed, you can start configuring the respective devices for communications.
   - Optionally, the desired device groups can be created under the **Device Groups** section with each of the PLC objects.
   - Desired device items can also be optionally created under the **Device Items** section with each of the PLC objects.

SIDirect Preconfigured Hierarchy

The server-specific configuration portion of the SIDirect Communication Driver hierarchy tree under the OI Server Manager starts at the PORT object. It represents the network board in the computer that communicates with the PLC. Usually it is an ordinary network card identified by the local IP address.

When you install the SIDirect Communication Driver, it is installed with a preconfigured device hierarchy in the **Operations Control Management Console**. The SIDirect Communication Driver uses a two-tier hierarchy for modeling the S7 PLC objects.This hierarchy consists of a Port object and two CPU objects:

- **PORT:** This is the CpS7 communications port object that communicates with a CpS7 node.
- **Legacy Object**: The Legacy object is subordinate to the PORT object. Use the legacy object to support absolute addressing (for example, "`DB1,INT0`"). Optimized Block Access must be disabled (non-optimized mode) in the S7 PLC in order to use absolute addressing.
- **Symbolic Object:** The Symbolic object is subordinate to the PORT object, and works with symbolic addressing to permit the use of enhanced tag names (for example, "`Tank4.Lvl`"). Symbolic addressing works whether or not optimized block access has been enabled in the S7 PLC.

To view the preconfigured hierarchy, click **Operations Integration Supervisory Servers -> Siemens - SIDirect -> OI.SIDIR.1 -> Configuration -> Port**

Note: If a configuration view is in an open state and you open the same server the second time, the Communication Driver locks the second instance of this same-server access for any update or configuration activities. Access to this second server resumes after the first one is closed.

Setting Up the SIDirect Communication Driver for the First Time

If you are setting up a Communication Driver for the first time, perform the following tasks in the order listed:

1. Review the items described in Before Configuring the SIDirect Communication Driver.
2. In the OI Server Manager tree, under the **Local** node, navigate the SIDirect Communication Driver in the Operations Control Management Console (OCMC).

3. Configure the global parameters. See "Configuring Global Parameters" in the Communication Drivers Pack help.
4. Add a port. See Configuring Port Parameters.
5. Add one or more device groups. See Device Group Definitions.
6. Add device items. See Device Item Definitions.
7. Activate the Communication Driver. See "Activating/Deactivating the OI Server" in the Communication Drivers Pack help
8. Access data from the client, see "Accessing Items Using the OPC Communications Protocol" in the Communication Drivers Pack help.
9. Troubleshoot any problems. See Troubleshooting the SIDirect Communication Driver.

Before Configuring the SIDirect Communication Driver

Before configuring the Communication Driver, verify the following items:

- A PC is set up with the necessary network cards, and is connected to the necessary networks.
- The Windows administration account is created or identified.
- The Communication Driver and any other AVEVA software such as the OI Server Manager is installed with the proper licenses. For more information, Communication Drivers Pack Help.
- The client software is installed.
- The device(s) is/are connected (networked) and, if necessary, programmed.

Before configuring the Communication Driver, you should know:

- The device network configuration and addresses.
- Which data items are needed for the client application.
- The device name/topic name/group name.
- The desired update intervals.

Special Pre-Configuration for S7-1200 and S7-1500 PLCs

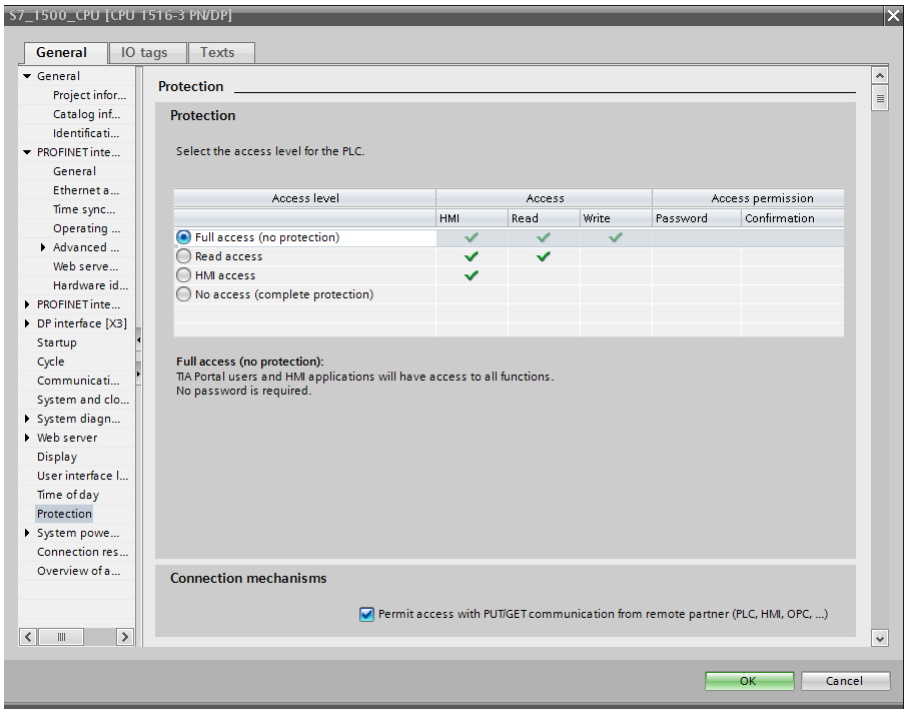
The SIDirect Legacy object requires some pre-configuration through the Siemens TIA portal to allow access to data inside the S7-1500 PLC. This pre-configuration consists of:

- Enabling PUT/GET communication
- Disabling optimized block access

Note: Pre-configuration is not required for the SIDirect Symbolic object.

To enable PUT/GET communication:

1. In the navigation pane of the TIA software, locate the S7-1500 CPU.
2. Right-click the S7-1500 CPU and then click **Properties**. The properties dialog box appears.



3. In the left pane, click **Protection**.
4. Select the required access level for the PLC.

   Note: If you select **No Access**, you will not be able to use the SIDirect **Legacy** object to access tags in the PLC.

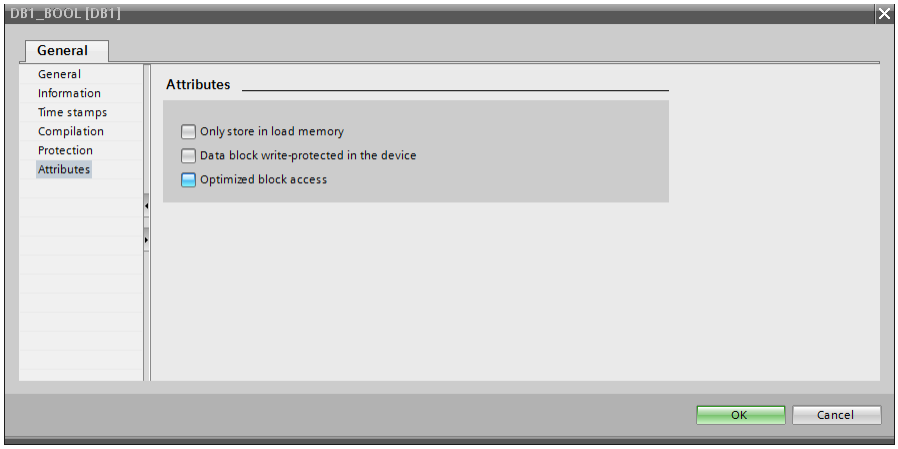
5. In the **Connection mechanisms** area, select the **Permit access with PUT/GET communication from remote partner** check box.

   Note: This field will be disabled if you select the access level as **NO Access**.

6. Click **OK**.

To disable optimized block access:

1. In the TIA software, locate the data block in the left tree.
2. Right-click the data block, and then click **Properties**. The properties dialog box appears.



3. In the left pane, click **Attributes**.
4. Click to clear the **Optimized block access** check box.
5. Click **OK**.

If the legacy communication is still not established, perform the following steps:

1. Go offline.
2. Compile-hardware rebuild (all).
3. Download to the PLC.

Configuring Port Parameters

When you first install the SIDirect Communication Driver, by default a PORT node with a Legacy object and a Symbol object is added. Only one PORT object is allowed per instance SIDirect Communication Driver. By default the port type will be TCP/IP and is not editable. You can disable or enable the PORT node. By default it will be enabled.

To disable the PORT node:

- Under **Configuration** node, right click **PORT** and select **Disable PORT**.

To enable the PORT node:

- Under **Configuration** node, right click **PORT** and select **Enable PORT**.

Configuring the SIDirect Legacy Object

The SIDirect Legacy object supports "traditional" absolute addressing of PLC items. If you are using symbolic addressing, use the SIDirect Symbolic object.

To add more Legacy objects

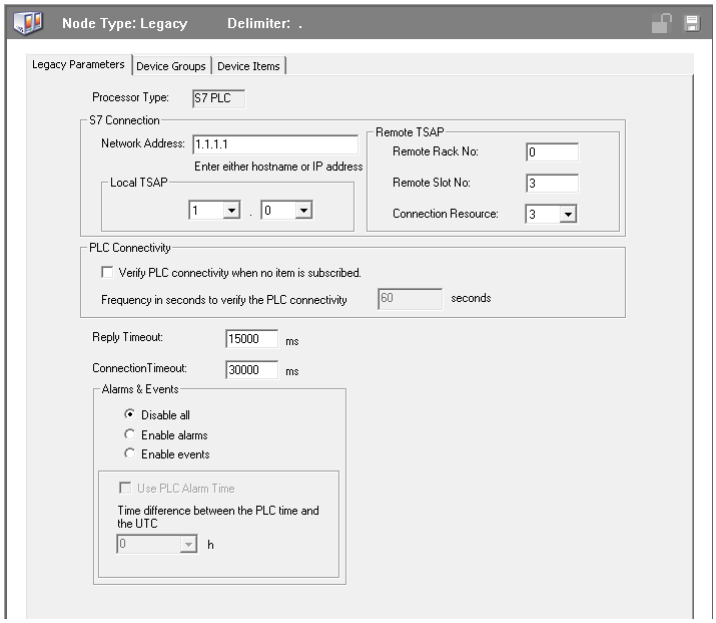
1. Right-click the PORT object and select **Add Legacy Connection**.
2. A new object **New_Legacy_xxx** is added to the hierarchy.Rename as needed.

Note: Up to 1024 Legacy and Symbolic objects (combined total) can exist in the hierarchy under PORT.

SIDirect Legacy Object Parameters

To view or edit the SIDirect Legacy object

- Click the **Legacy** object. The **Legacy Parameters** configuration view appears.



Edit the elements in the **Parameters** page as needed. Five of the six elements in the page are configurable.

Note: The Legacy object is preconfigured with three Device Groups and a default network address. The default network address and preconfigured Device Groups are not present in Legacy objects ("New_Legacy_000") that you add.

- **Processor Type** (not configurable): S7 PLC.
- **S7 Connection:** This has three configurable settings:a
  - **Network Address:** The IP address or host name of the remote S7 CP.

    Enter network address where the PLC is located (for example, "10.11.12.13"), or a host name if one is defined in the local hostlist. The address field cannot be blank and the IP address or host name cannot be more than 255 characters.

    Note: The preconfigured Legacy object includes a default network address: "1.1.1.1". Any additional Legacy objects that you add do not contain the default address (network address is blank).
  - **Local TSAP:** The local TSAP of the computer. Select the Hex numbers for the connection resources from the menu. The Local TSAP consists of two (2) Hex numbers. The first number corresponds to the connection resource. Each number ranges from 00 to FF. The default values are 01 and 00, respectively.
  - **Remote TSAP:** This Remote TSAP corresponds to what you configured in the TSAP for the S7 CP. Configure the Remote TSAP by typing in the decimal numbers for the Remote Rack and Remote Slot, and by selecting the Hex number for the Connection Resource from the menu. The values for the Remote Rack No. and Remote Slot No. range from 0 to 255, with the default values of 0 and 3, respectively. The value for the Connection Resource ranges from 00 to FF. The default value is 03.
- **PLC Connectivity:** The watchdog scheme for detecting the connectivity status to the PLC when there are no activities (no items are subscribed to).
  - Select the **Verify connectivity when no item is subscribed** check box to turn on the watchdog.
  - Specify the watchdog time interval, in seconds, in the **Frequency in seconds to verify the PLC connectivity** box.

- **Reply Timeout:** Enter a value, in milliseconds, beyond which messages time out.
  - Allowable range is 0 to 100,000 milliseconds.
  - The default value is 15,000 milliseconds. If you decrease this value, the SIDirect Communication Driver reacts faster to a communications failure.
- **Connection Timeout:** Enter a value, in milliseconds, beyond which a pending request to initiate a connection times out.
  - Allowable range is 0 to 100,000 milliseconds.
  - The default value is 30,000 milliseconds.
- **Alarms and Events:** Enable Alarms or Events or disable both for this connection by selecting:
  - Disable all
  - Enable alarms
  - Enable events

  On any one connection, you can configure Alarms, Events, or none. If you need to access both Alarms and Events, create two different connections.
- **Use PLC Alarm Time:** The alarm time in the PLC used to timestamp alarm and event-related data.

  You can enable this feature to time stamp data in the alarm and event blocks with the timestamps provided in the alarm and event blocks. The Communication Driver, in this case, will not generate its timestamp for the received data. This feature is only available for data in the alarm and event blocks.
- **Time Difference Between the PLC Time and the UTC:** The time difference, in hours, between the PLC time and the UTC time.
  - The value range is from +12 hours to -12 hours.
  - The default value is 0.

Configuring the SIDirect Symbolic Object

The SIDirect Symbolic object supports symbolic addressing of PLC items. If you are using absolute addressing, use the SIDirect Legacy object.

To add more SIDirect Symbolic objects

1. Right-click on the **PORT** object and select **Add Symbolic Connection**.
2. A new object **New_Symbolic_xxx** is added to the hierarchy. Rename as needed.

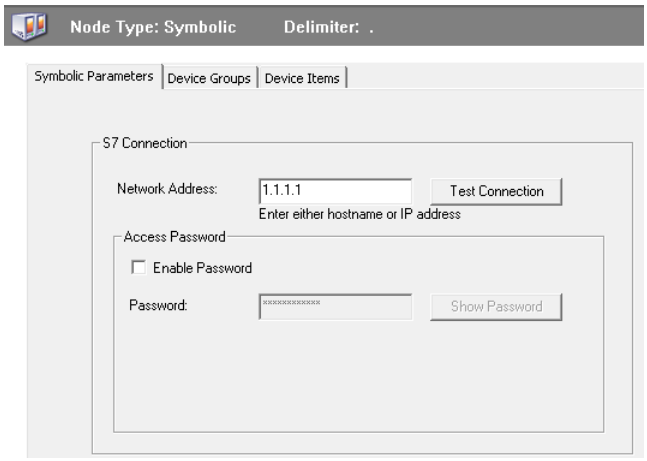
Note: Up to 1024 S7 objects (combined total of Legacy and Symbolic objects) can be added under PORT.

SIDirect Symbolic Object Parameters

To view or edit the SIDirect Symbolic object

- Click the **Symbolic** object. The **Symbolic Parameters** configuration view appears.

  Edit the elements in the **Parameters** page as needed. Configurable elements are network address and password settings.



Note: The SIDirect Symbolic object is preconfigured with three Device Groups and a default network address. The default network address and preconfigured Device Groups are not present in additional Symbolic objects ("New_Symbolic_000") that you add.

- **S7 Connection:** This has two configurable settings:
  - **Network Address:** The IP address or host name of the remote S7 PLC.

    Enter the network address (for example, "10.11.12.13"), or a host name if one is defined in the local hostlist. The address field cannot be blank and the IP address or host name cannot be more than 255 characters.

    Note: The preconfigured SIDirect Symbolic object includes a default network address: "1.1.1.1". Any additional Symbolic objects that you add do not contain the default address (network address is blank).
  - **Enable Password:** You can enable password protection for access to the PLC.
    i. Select the **Enable Password** check-box to activate the password entry field.
    ii. Click **Show Password** to view the password you entered.

    To help ensure security, passwords for Symbolic node objects can only be set on the local node and cannot be transferred from one Configuration File (.aaCfg) to another. Therefore, if Password is enabled on any Symbolic node object, each must be re-entered after any of the following operations:
    - Copying the Configuration File to another computer
    - Cloning the base server
    - Creating an instance of the base server

Device Redundancy

The OI Server Manager provides the ability to assign redundant device for fail-over protection in the event of device failure. Two devices must be configured in the same Communication Driver having identical item syntax.

Primary and secondary devices will be set up in the REDUNDANT_DEVICE object in the OCMC, along with a common item name (ping item) shared by each device to determine device status.

Note: Unsolicited message configuration is not supported from the device redundant hierarchy.

Run-time Behavior of Redundant Devices

The Communication Driver starts with the active device. The OI Engine switches to the standby device when the active device fails to communicate. The value of the SYSStatus determines the communication failure.

Note: The value of the SYSStatus of the standby device must be TRUE in order to switch over to the standby device. Otherwise, there will not be any failover.

When SYSStatus shows a FALSE value at both active and standby devices, the OI Engine considers a complete communication failure and mark all the items subscribed to the redundancy device hierarchy with the current time and the appropriate OPC quality. The OI Engine activates the slow-poll mechanism to retry the communication to both devices until either one of the Ping Items returns to a good quality and update its SYSStatus item to TRUE.

When the OI Engine switches to the standby device, the standby device becomes active and the originally active device becomes the standby.

When the active device becomes the standby device the Ping Item will not be deleted from that the standby device. This ensures the standby is able to recover the communication again.

Note: To allow the failover to function properly, the Ping Item must be a valid PLC item that has not been rejected the server. System items (items beginning with SYS) cannot be used as the Ping Item. See SIDirect Driver Diagnostic Info Items for the list of system items.

The Communication Driver logs any failover activities. All other functionality such as diagnostics, enable/disable, and reset is performed exactly same as it is performed for any other hierarchy node.

Note: Unsolicited message configuration is not supported in the Redundant Device Object itself. You can still receive unsolicited messages directly from device groups defined in the regular server hierarchy.

This feature allows the Communication Driver to provide fail over support by providing one node which switches between two other nodes. The Redundant device is configured with a redundancy node which directs itself to one of the two nodes and switches to the other based on lack of communications to a common user-configured controller item. In this manner the Redundant Device Object can be used to direct client requests to the redundant node, which switches between device or communication pathway failure without intervention.

In both stand-alone and redundant configurations, the SIDirect Communication Driver supports subscription and poking. Block services are not supported.

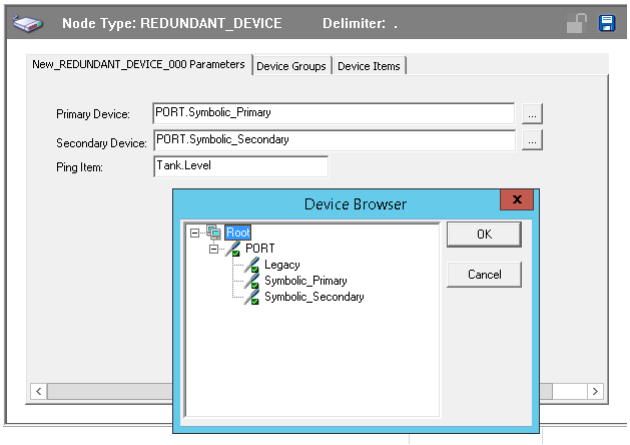
Configuring Device Redundancy

To setup up a REDUNDANT_DEVICE from the configuration branch

1. Configure a primary device in the OI Server Manager in the OCMC. You must enter a valid network address in the configuration parameters, and add an item reference that can be shared with the the device's secondary partner.
2. Configure a secondary device. Enter a valid network address and add the same item reference that was added for the primary device.

   Important: You cannot mix Legacy and Symbolic objects when configuring a redundant pair, since these objects use different item syntaxes. Both primary and secondary devices must be the same type.
3. Select and right-click on the configuration node under the OI.SIDIR.1 object.
4. Select Add REDUNDANT_DEVICE Object. An object called New_REDUNDANT_DEVICE_000 is created.

5. Rename the newly created object as appropriate. The New_REDUNDANT_DEVICE_000 configuration view is displayed in the Configuration branch of the hierarchy.
6. Enter or use the device browser to select the primary and secondary devices. Open the device browser by clicking on the ellipses (...) button and expand the device hierarchy that is displayed in the browser.



7. Enter one device item as the Ping Item that can be shared between the primary and secondary devices to determine device status.
8. Save the hierarchy node configuration by clicking on the save icon.

Important: A Ping item must be specified and be a valid tag in both the primary and secondary controllers to determine the connection status for SYSStatus. The Ping item can be a static item in the device such as a firmware version or processor type. If the Ping item is invalid or does not exist in the controller, the failover operation may not work correctly as the value of SYSStatus may continue to stay as FALSE in the standby device.

AVEVA

Chapter 3

Device Groups and Device Items

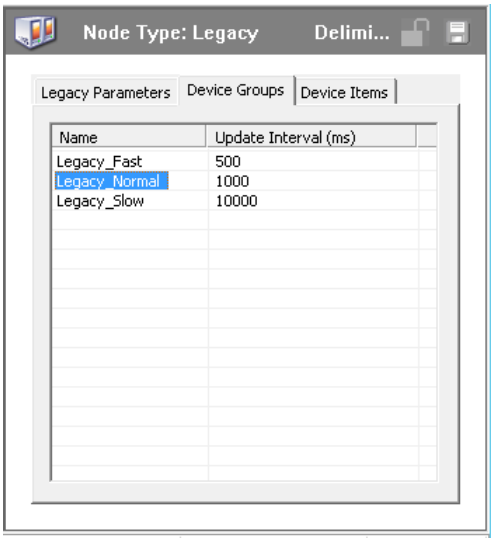
The **Device Groups and Device Items** a**llow you to** create new, modify, or delete device-group and device-item definitions for a SIDirect Symbolic object.

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

Each configuration view associated with objects in the SIDirect Communication Driver hierarchy tree has a **Save** button. When you modify the **Parameters**, **Device Groups**, or the **Device Items** dialog box, click **Save** to implement the new modifications. If you try to open another configuration dialog box you are prompted to save the new data to the configuration set.

Device Group Definitions

The **Device Groups** dialog box, which appears by clicking the **Device Groups** tab in the **Legacy** or the **Symbolic** configuration view, is the place where you create, add, delete, and define device groups. You can also configure default update intervals for the objects and edit update intervals in this dialog box.



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

To create or add device groups

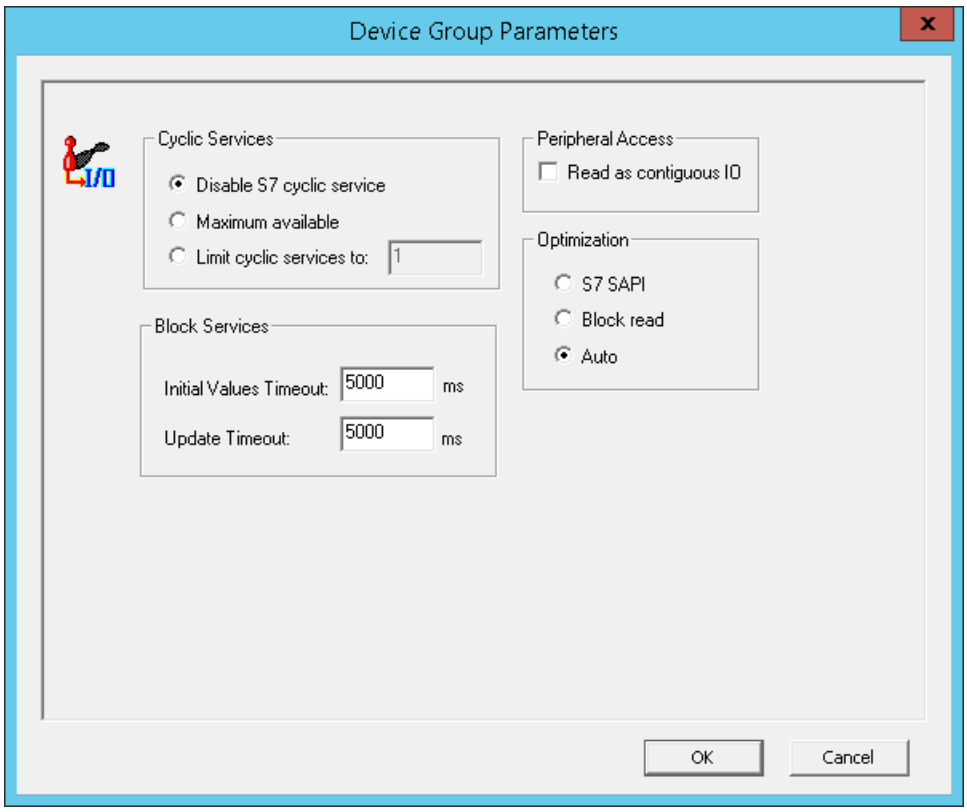
1. Right-click in the **Device Groups** box and click **Add**.
2. Enter a unique name for the device group. Device group names are case-insensitive.

To delete device groups

- Right-click on the device group to be deleted from the list and select **Delete**.

To edit device groups

1. Right-click on the device group to be edited.
2. Select **Edit** from the menu to open the **Device Group Parameters** dialog box.



3. Make the necessary edits.

Note: The **Edit** option is enabled only for Legacy connection.

The **Edit** dialog box contains four configurable elements:

- **Cyclic Services**
  If you use Cyclic Services (the **Disable S7 cyclic service** option is not selected), configure two additional settings:
  - Maximum available
  - Limit cyclic services to

    If you know how many services the remote PLC can handle, you can limit the use of cyclic services in this device group and distribute the available cyclic services among the device groups associated with this connection. Otherwise, you can use the maximum available services.
  - Allowable range for **Limit cyclic services to** is 1 to 150.
  - The default is 1.

    Click on **Disable S7 cyclic service** to disable the S7 cyclic services for the device group.
  - Cyclic services have a reliable update frequency and need less bus access. They are a limited resource in the PLC and/or Communications Processor.

    If you select the **Disable S7 cyclic service** option, the SIDirect Communication Driver to read input and output blocks (also peripherals) if their address spaces are not contiguous.
    Select this check box if you have some holes in the input-address or output-address space in your PLC.

- **Optimization**
  Select one of the following settings to configure the optimization mode the SIDirect needs to use to acquire data from the PLC:
  - S7 SAPI
  - Block read
  - Auto

    The default is Auto. For more detailed information, see "Optimization Considerations".

    Note: The S7-1200 and S7-1500 PLCs do not support SAPI.
- **Block Services**
  Communication DriverIf the Block Services function is required, you have to configure two settings in the **Block Services** box:
  - **Initial Values Timeout**
    Allowable range for Initial Values Timeout is 0 to 65,535 milliseconds; the default value is 5,000.
  - **Update Timeout**
    Allowable range for Update Timeout is 0 to 65,535 milliseconds. The default value is 5,000.

    The Block Services function needs time-outs to supervise reading initial values and updating the block items to this connection. A time-out value of 0 (zero) disables the time supervision of block messages.

    Block services are unconfirmed services. If the remote station does not send data within this time range, the Block Services is reinitialized and an error message appears in the Logger.

To configure default update intervals

- To configure a default update interval for the object, right-click in the **Device Groups** box and click **Config Default Update Interval** on the menu.

To edit update intervals

- To edit the update interval for an object, double-click its value in the **Update Interval** column and make the edits.
  - Update Interval is the frequency in milliseconds that the SIDirect Communication Driver acquires data from the topics associated with that device group.
  - Different topics can be polled at different rates in a PLC by defining multiple device-group names for the same PLC and setting a different Update Interval for each device group.

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

Optimization Considerations

The SIDirect Communication Driver uses the following same optimization considerations as the S7 Communication Driver:

- Use different Poke modes.
- Use different reading optimization modes.
- Use cyclic services to minimize traffic.

- Use block services to minimize traffic.

The Communication Driver can also optimize its performance in getting data from the PLC by using the optimization mode. The following options are available:

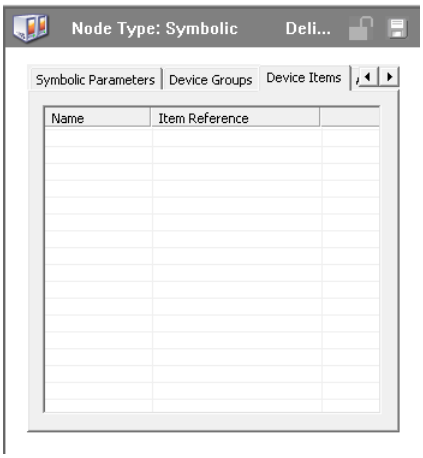
- S7 SAPI
  The S7 SAPI mode is the same optimization mode used in the pre-release of the former Wonderware Siemens SIMATIC NET S7 I/O Server. This mode is implemented to keep the server compatible to the former server. This mode is the less-preferred optimization mode.
- Block Read
  The Block Read mode always registers a whole byte array containing some items. If you frequently switch items (activating and deactivating) that have similar addresses, this mode is the best selection. In this mode, there are less activations and deactivations on the protocol.
- Auto
  By default, the Auto mode is best to use when exploiting the whole PDU (Protocol Data Unit). The Auto mode has the best performance in cases where you do not make many activations and deactivations. The S7-300 family of controllers can process PDU lengths of 480 to 960 bytes, depending on the controller. An S7-400 controller can process PDU lengths of 960 bytes.

Note: The S7-1200 and 1500 PLCs do not support cyclic services or SAPI.

Device Item Definitions

The predefined item syntax/name for the S7 PLC cannot be changed. However, to make it easier to remember item names, you can create aliases for these item names.
For example, it may be easier for you to remember the item syntax "mb80" as "Temperature."

Select the **Device Item**s tab in the OI Server Manager user interface 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 appears when you click the **Device Items** tab in the **New_S7Cp_000 Parameters** configuration view.



After you configure item names, the Communication Driver can perform OPC Item browsing. When the Communication Driver is running and an OPC client requests item information, the configured items appear under the PLC hierarchy node. User-defined data types appears at the lowest level in the hierarchy when browsed via the OPC client.

To create or add device items

1. Right-click in the **Device Items** box and click **Add**.
2. Type the item name in the **Name** column.
   For example, "Temperature."
3. When you add a new device item, enter a unique name.
4. Double-click the line in the **Item Reference** column and enter the correlated item reference for the name you selected.
   For example, "mb80."

   Note: If the name and the item reference are the same, it is only necessary to enter a name. The Communication Driver assumes that the item reference is the same. This is necessary if you want to add some items for browsing via the OPC, even if they do not have a symbolic name.

To rename device items

- Right-click on the device item to be renamed and click **Rename**. Make the changes.

To delete device items

- Right-click on the device item to be deleted from the list and click **Delete**.

To clear all device items

- Right-click in the **Device Items** box and click **Clear All**. All the device items listed are cleared after you confirm their deletion.

Exporting and Importing Communication Driver Item Data

You can export and import the Communication Driver item data to and from a CSV file after you configure the Device Items. This lets you perform an off-line, large-scale edit on the item data configured for a PLC and import that data back into the PLC configuration.

To export Communication Driver item data to a CSV file

1. Right-click in the **Device Items** box and click **Export**. The **Save As** dialog box appears and the file name defaults to "PLC Hierarchyname.csv," within the current-system-configured default directory.
2. Accept the defaults to save the file. The file is saved as New_S7Cp_000.csv. Now you can edit it in Microsoft Excel.

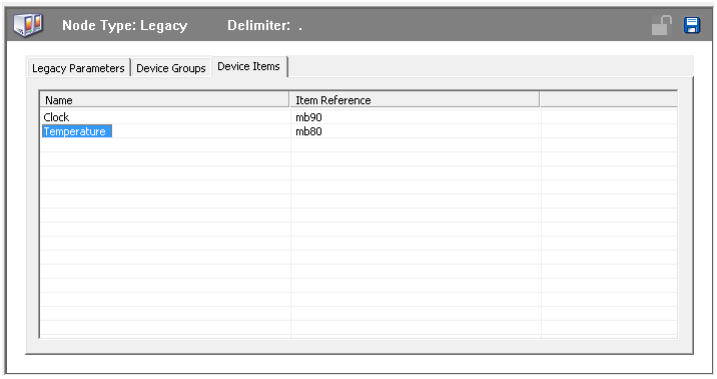
   The file contains one row for each item configured with two columns: Name and Item Reference.

After you are done editing, you are ready to import the edited file into the OI Server Manager.

To import Communication Driver item data from a CSV file

1. Right-click in the **Device Items** box.
2. Click **Clear All** to clear all the item data you want to replace with the edited .csv file. The data is cleared after you click **Yes** to confirm the deletion.
3. Click **Import** on the menu. The **Open** dialog box appears.
4. Browse for the .csv file you want to import, select it, then click **OK**.
   The OI Server Manager imports the file. You see the data in the **Device Items** box.

When the file is imported, new item references are added based on unique names. If there are duplicate names, you can replace the existing entry with a new entry or ignore the new entry.



When the Communication Driver is running and an OPC Client requests item information, the imported configured items appear under the PLC hierarchy node.

Note: The SIDirect Communication Driver does not support importing preconfigured items (alias names of items that work with OPC) in an output file generated by the Siemens Step7 software.

SIDirect Scan-Based Message Handling

AVEVA Communication Drivers poll hardware devices for information. This polling is requested by one or more clients.

After a client requests a particular piece of information, the SIDirect Communication Driver creates its own request and sends that request to the hardware device. The SIDirect Communication Driver then waits for a response to its request. After the SIDirect Communication Driver receives the information, it passes that information back to the client and repeats the process until all clients stop requesting information.

You define the rate at which the SIDirect Communication Driver polls a particular device for a specific piece of information in the device group (topic definition) inside the SIDirect Communication Driver. You use 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 very fast response, the temptation is to set the Update Interval to a value close to 0 (zero) seconds. However, if every item is polled at this rate, the entire system suffers due to slow response time. Therefore, compromise and set the Update Interval to a more reasonable value.

You can 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.

Using the S7 Tag Creator

The SIDirect Communication Driver supports the S7 Tag Creator to create the tags. You can also use the S7 Tag Creator editor for categorizing tags into different views, importing tags into different sections and areas, and generating tags automatically.

The SIDirect S7 Tag Creator is similar to the current S7 I/O Server S7 Tag Creator, except that the infrastructure for the S7 Tag Creator allows you to create S7 tags for the SIDirect DIObjects, while the I/O Server S7 Tag Creator

works only with the S7 I/O Server. Additionally, the SIDirect S7 Tag Creator has numerous other features that are not found in the I/O Server S7 Tag Creator.

Note: The S7 Tag Creator does not read the current memory/register layout from the S7 PLC online.

Using the SIDirect DIObject S7 Tag Creator, you can:

- Generate alias names and tag names from the Siemens Step 7 .awl, .asc, .dif, .sdf, and .seq output files.
- Categorize tags into three different views for your selection: S7 tag hierarchy, alias name, and tag data type.
- Continuously apply user-defined filter criteria for alias names.
- Automatically generate tags.
- Import tags into the Scan Group, Block Read, and Block Write DA Groups on the S7C_S7 DIObject editor.

The S7 Tag Creator module generates tags for ArchestrA and SIDirect DIObjects. You can either directly specify the tags or import tag files generated by the Siemens S7 Step 7 programming software.

Unsolicited Message Handling

The SIDirect Communication Driver processes the following three types of unsolicited messages sent by the S7 PLCs:

- Alarms
- Events
- Block Services

Block Services

In addition to unsolicited messages based on critical conditions or events, S7 PLCs can also handle another type of unsolicited messages called Block Services.

You can use Block Services to send blocks of data up to 64KBytes within one send job and trigger it by using a timer, an event, an I/O activity, or initiate it via a program code.

Chapter 4

Using Auto-Build with the SIDirect Communication Driver

About Auto-Build and the SIDirect 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 under the **Symbolic** configuration node.

Auto-Build browsing can be accomplished online if connected to the PLC.

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 Server 2012 R2, Windows 10 and Windows 8.1 Update • Requires manual download for Windows Server 2008 R2 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 3.0	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 Internet Explorer 11 or higher for functioning. For IE versions lower than 11, an error message is displayed in the Auto-Build tab:

IE version X is not supported. Please upgrade to IE 11

IE 11 is pre-installed with Windows 2016 (not as default browser), Windows Server 2012 R2, Windows 10 and Windows 8.1. IE 11 should be manually downloaded for Windows Server 2008 R2 and Windows 7

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 the Licensing section of the Communication Driver Pack help for more details.

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

Unable to obtain license for Auto-Build

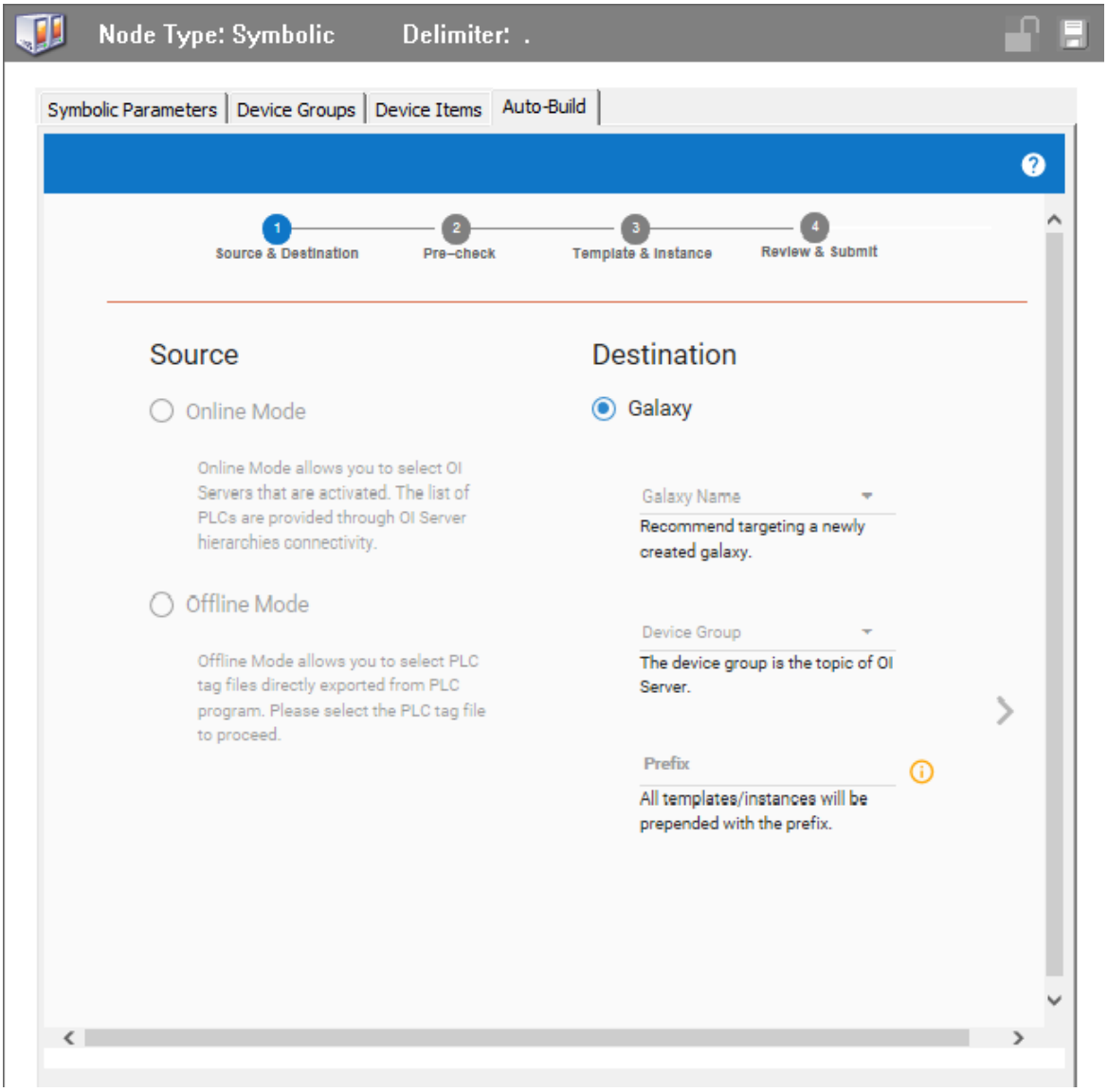
A standard license, or no license will allow the user to exercise Auto-Build configuration steps, but not building the objects in Application Server.

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

To view the Auto-Build screen, under the **Symbolic** configuration node, click the **Auto-Build** tab.



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

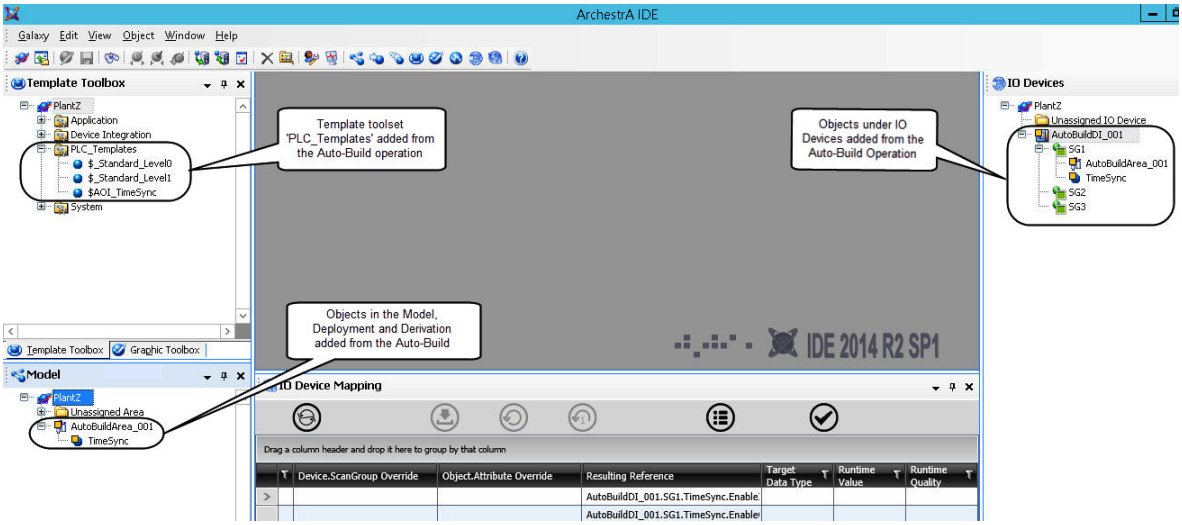
1. To terminate the Auto Build operation, in the progress bar, click the **Stop**.
2. The progress displays the terminating process.

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

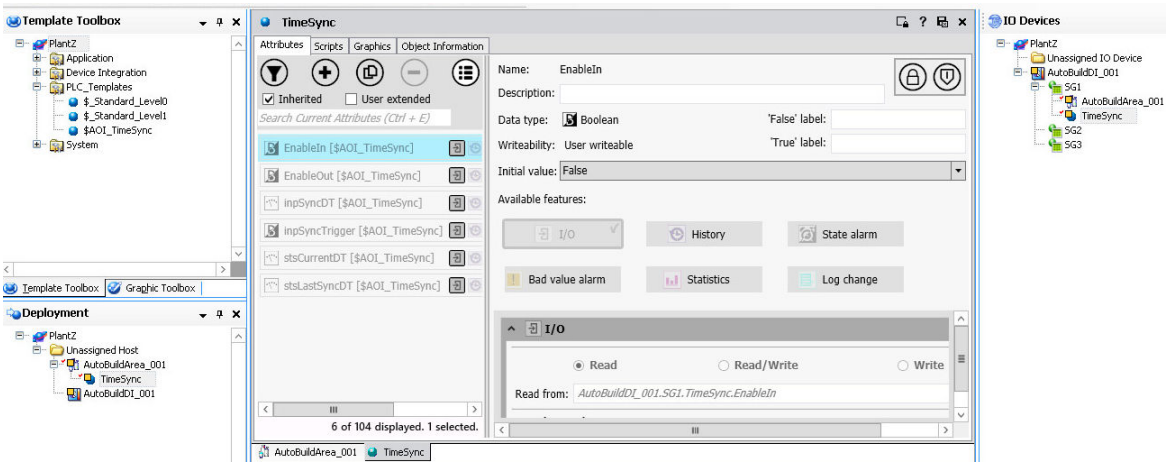
Note: Click anywhere on the screen during the building or build termination stages to minimize the progress to the title bar. Click the title bar again to bring back the progress bar.

Template Generation in the Application Server

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



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



PLC Tag Database Feature Support

Siemens PLC TagDB features support status:

Category	TagDB Item	Siemens Program Feature	Auto-Build Feature
Data Type	Pre-defined data types	System Pre-Defined (PDT) data structure	Template
Data Type	PLC Data Types	User Defined (UDT) data structure	Template
Scope	PLC Tags	Global tags	Instance/Attribute
Usage	Accessible from HMI/OPC UA/	Parameter for UDT	Not Supported
Usage	Writable from HMI/OPC UA	Parameter for UDT	Not Supported
Usage	Visible in HMI engineering	Parameter for UDT	Not Supported
Tags	Tag Table	User-defined Tag table	Template
Tags	Tag Table	Default Tag table	Not Supported
Tags	Data Type	Refer to the entries under "Data Type" Category above	Attribute (non-specific field)
Tags	Address	PLC address for the Tag	Not Supported
Tags	Retain	Tag properties for Retain/Non retain	Not Supported
Tags	Supervision	Tag properties for different supervisions.	Not Supported
Tags	Comment	Description for the Tag	Not Supported

Auto-Build Data Type Mapping

Data Type	Data Type Description	AppServer Data Type
BOOL	Boolean	MxBoolean
BYTE, UINT8, USINT, BITSET8	Integer as 8-bit unsigned integer	MxInteger
SINT, INT8	Integer as 8-bit signed integer	MxInteger

INT, INTEGER, INT16	Integer as 16-bit signed integer	MxInteger
UINT, UINT16, WORD, BITSET16	Integer as 16-bit unsigned integer	MxInteger
DINT, INT32	Integer as 32-bit signed integer	MxInteger
UDINT, UINT32, DWORD, BITSET32, TIMER	Integer as 32-bit unsigned integer	MxInteger
LINT, INT64	Integer as 64-bit signed integer	MxInteger
LWORD, BITSET64, UINT64, ULINT	Integer as 64-bit unsigned integer	MxInteger
REAL, REAL32	Real as 32-bit IEEE Floating-point number	MxFloat
DOUBLE, LREAL, REAL64	Real as 64-bit IEEE Floating-point number	MxDouble
CHAR, STRING, STRING2	String as character bytes	MxString
WCHAR, WSTRING, WSTRING2	Wide character string	MxString
TIMESPAN	Elapsed time	MxElapsedTime
LTIME, S5TIME, SINTTIME, TIME_OF_DAY, LTIME_OF_DAY,LDT, DATE	Time	MxTime

AVEVA

Chapter 5

SIDirect Communication Driver Reference

SIDirect Legacy Object Reference (Absolute Addressing)

This section provides reference information specific to the SIDirect Legacy object. The Legacy object is used for absolute addressing (non-symbolic names).

SIDirect Absolute Naming Convention

The SIDirect Legacy interface included with the SIDirect Communication Driver supports only "traditional" absolute addressing of PLC items. To use absolute addressing:

- Optimized Block Access must be disabled (non-optimized mode).
- GET/PUT must be enabled.

Note: Symbolic addressing is only supported through the S7_1500_Plus interface.

In general, the item syntax takes the following format:

```
D[DataBlock#],[DataTypePrefix]
[OffsetAddress]<.BitAddress>,<ArraySize><DataDisplayFormatSuffix>
```

Example:

`DB10,INT2,8`	Data Block #10 Integer array item of 8 elements starting at 2 bytes offset
`DB25,DWORD4BCD`	Data Block #25 Double Word item shown in BCD starting at 4 bytes offset

Note: The common “D” data block identifier above maybe replaced by a prefix letter such as F (Flag), M (Memory), I (Input), Q (Output), PI (Peripheral Input), PQ (Peripheral Output), T (Timer), C (Counter) for register items from various special Areas in the S7 PLC program.

Item Naming

The SIDirect Communication Driver uses an item-naming convention based on the two-letter data-type identifiers used in programming the Siemens PLCs. The server accepts both the English and German standard identifiers.

The tables in this section describe the item naming convention for the Siemens S7-1500 PLCs. The ranges specified in those tables vary according to the type of the controller used.

Note: PDU (Protocol Data Unit) size is factored in determining Read/Write size limits. The S7-300 family of controllers can process PDU lengths of 480 to 960 bytes, depending on the controller. An S7-400 controller can process PDU lengths of 960 bytes.

Data Blocks and Instance Blocks

The following table summarizes the data format, item or point, suffix, data type, and range for Data Blocks and Instance Blocks. See the S7-1200 and S7-1500 sections in this documentation for specific item naming conventions and/or limitations for those controllers.

Data Format	Item/Point	Suffix	Data Type	Range
Bit	D<B,I>*d*,X*x.y*		VT_BOOL	0 or 1
String	D<B,I>*d*,S*x,v* D<B,I>*d*,STRING*x,v*		VT_BSTR VT_BSTR	String String
S7String	D<B,I>*d*,S7S*x,w* D<B,I>*d*,S7STRING*x,w*		VT_BSTR VT_BSTR	String String
Byte	D<B,I>*d*,B*x* D<B,I>*d*,BYTE*x*	DT	VT_UI1 VT_UI1 VT_BSTR	0 to 255 0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
Byte Array	D<B,I>*d*,B*x,v* D<B,I>*d*,BYTE*x,v*		VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1	0 to 255 for each element* 0 to 255 for each element*
USINT	D<B,I>*d*,USINT*x*	DT	VT_UI1 VT_BSTR	0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
USINT Array	D<B,I>*d*,USINT*x,v*		VT_ARRAY|VT_UI1	0 to 255 for each element*
Char	D<B,I>*d*,CHAR*x*	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
Char Array	D<B,I>*d*,CHAR*x,v*		VT_ARRAY|VT_UI1	-128 to 127 for each element*

SINT	D<B,I>*d*,SINT*x*	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
SINT Array	D<B,I>*d*SINT*x,v*		VT_ARRAY|VT_UI1	-128 to 127 for each element*
Word	D<B,I>*d*,W*n* D<B,I>*d*,WORD*n*	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 65535 0 to 9999*** 0.0 to 999.3*** 0ms to 2h46m30s0ms*** 0.0 to 9990.0 (s)*** 1990-1-1 to 2168-12-31
Word Array	D<B,I>*d*,W*n,v* D<B,I>*d*,WORDn,v		VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2	0 to 65535 for each element* 0 to 65535 for each element*
UINT	D<B,I>*d*,UINT*n*	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 9999*** 0.0 to 999.3*** 0ms to 2h46m30s0ms*** 0.0 to 9990.0 (s)*** 1990-1-1 to 2168-12-31
UINT Array	D<B,I>*d*,UINT*n,v*		VT_ARRAY|VT_UI2	0 to 65535 for each element*
Integer	D<B,I>*d*,INT*n*	BCD D	VT_I2 VT_I2 VT_BSTR	-32768 to 32767 -999 to 999*** 1990-1-1 to 2168-12-31
Integer Array	D<B,I>*d*,INT*n,v*		VT_ARRAY|VT_UI1	-32768 to 32767 for each element*
Double Word	D<B,I>*d*,D*m* D<B,I>*d*,DWORD*m*	BCD TOD T	VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Word Array	D<B,I>*d*,D*m,v* D<B,I>*d*,DWORD*m,v*		VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4	0 to 4294967295 for each element*

				0 to 4294967295 for each element*
UDINT	D<B,I>*d*,UDINT*m*	BCD TOD T	VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
UDINT Array	D<B,I>*d*,UDINT*m,v*		VT_ARRAY|VT_UI4	0 to 4294967295 for each element*
Double Integer	D<B,I>*d*,DINT*m*	BCD TOD T	VT_I4 VT_I4 VT_BSTR VT_BSTR	-2147483648 to 2147483647 -9999999 to 9999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Integer Array	D<B,I>*d*,DINT*m,v*		VT_ARRAY|VT_I4	-2147483648 to 2147483647 for each element*
Real	D<B,I>*d*,REAL*m*		VT_R4	+/-1.2e-38 to +/-3.4e+38
Real Array	D<B,I>*d*,REAL*m,v*		VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element*

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big endian format when data is returned to the DDE/SuiteLink layer.

**: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

***: The Communication Driver provides suffixes (S5T, TR, DT, KT, and BCD) that are used to interpret raw bytes as having a special data format. Review the PLC processor documentation, as not all processors have ladder logic internally to interpret those data formats the same way (for example, the S7-1200 PLC).

Where:

d is the data block number, with a range from 1 to 65535.

x is the start address, with a range from 0 to 65535.

y is the bit position, with a range from 0 to 7.

0 is the LSB (Least Significant Bit).

7 is the MSB (Most Significant Bit).

n is the start address of 2-byte data/2-byte data arrays.

m	is the start address of 4-byte data/4-byte data arrays.
v	is the length of data in elements (an item in an array), with a range from 0 to 65534.
w	is the length of the net S7 string data in characters (size in S7 message is w+1, size of string representation in S7 PLC is w+2).

Note: All data blocks are **Read/Write**. The longest string or array that can be read in a cyclic service has the length of the PDU size minus 32 bytes. The longest string the InTouch software can process is 131 bytes. The longest string that can be poked is 256 bytes or the PDU size minus 28 bytes, whichever is less. The Communication Driver processes a write (**POKE**) to a Data Block.

Examples for S7-300/400/1200/1500 PLCs:

DB123,W24

DB23,DINT10BCD

DI5,X2.0

DI6,BYTE4,10

Flag Bytes

The following table summarizes data format, item or point, suffix, data type, and range for Flag Bytes. See Conversions and Suffixes of Items (Absolute Addressing) for suffix definitions.

Data Format	Item/Point	Suffix	Data Type	Range
Bit	FX*x.y*		VT_BOOL	0 or 1
	MX*x.y*		VT_BOOL	0 or 1
String	FS*x,v*		VT_BSTR	String
	MS*x,v*		VT_BSTR	String
	FSTRING*x,v*		VT_BSTR	String
	MSTRING*x,v*		VT_BSTR	String
Byte	FB*x*	DT	VT_UI1	0 to 255
	MB*x*		VT_UI1	0 to 255
	FBYTE*x*		VT_UI1	0 to 255
	MBYTE*x*		VT_UI1	0 to 255
			VT_BSTR	1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999

Byte Array	FB*x,v* MB*x,v* FBYTE*x,v* MBYTE*x,v*		VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1	0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element*
USINT	D<B,I>*d,*USINT*x*	DT	VT_UI1 VT_BSTR	0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
USINT Array	D<B,I>*d*,USINT*x,v*		VT_ARRAY|VT_UI1	0 to 255 for each element*
Char	FCHAR*x* MCHAR*x*	DT	VT_I1 VT_I1 VT_BSTR	-128 to 127 -128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Char Array	FCHAR*x,v* MCHAR*x,v*		VT_ARRAY|VT_I1 VT_ARRAY|VT_I1	-128 to 127 for each element* -128 to 127 for each element*
SINT	D<B,I>*d*,SINT*x*	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
SINT Array	D<B,I>*d*SINT*x,v*		VT_ARRAY|VT_UI1	-128 to 127 for each element*
Word	FW*n* MW*n* FWORD*n* MWORD*n*	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 65535 0 to 65535 0 to 65535 0 to 9999 0.0 to 999.3 0ms to 2h46m30ms 0.0 to 9990.0 (s) 1990-1-1 to 2168-12-31
Word Array	FW*n,v* MW*n,v* FWORD*n,v* MWORD*n,v*		VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2	0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element*

UINT	D<B,I>*d*,UINT*n*	BCD KT S5T TR D 	VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 9999*** 0.0 to 999.3*** 0ms to 2h46m30s0ms*** 0.0 to 9990.0 (s)*** 1990-1-1 to 2168-12-31
UINT Array	D<B,I>*d*,UINT*n,v*		VT_ARRAY|VT_UI2	0 to 65535 for each element*
Integer	FINT*n* MINT*n*	BCD D	VT_I2 VT_I2 VT_I2 VT_BSTR	-32768 to 32767 -32768 to 32767 -999 to 999 1990-1-1 to 2168-12-31
Integer Array	FINT*n,v* MINT*n,v*		VT_ARRAY|VT_I2 VT_ARRAY|VT_I2	-32768 to 32767 for each element* -32768 to 32767 for each element*
Double Word	FD*m* MD*m* FDWORD*m* MDWORD*m*	BCD TOD T	VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 9999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Word Array	FD*m,v* MD*m,v* FDWORD*m,v* MDWORD*m,v*		VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4	0 to 4294967295 for each element* 0 to 4294967295 for each element* 0 to 4294967295 for each element* 0 to 4294967295 for each element*
UDINT	D<B,I>d,UDINT*m*	BCD TOD T	VT_UI4 VT_UI4 VT_BSTR	0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999

			VT_BSTR	-24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
UDINT Array	D<B,I>d,UDINT*m,v*		VT_ARRAY|VT_UI4	0 to 4294967295 for each element*
Double Integer	FDINT*m* MDINTm	BCD TOD T	VT_14 VT_14 VT_14 VT_BSTR VT_BSTR	-2147483648 to 2147483647 -2147483648 to 2147483647 -9999999 to 9999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Integer Array	FDINT*m,v* MDINT*m,v*		VT_ARRAY|VT_I4 VT_ARRAY|VT_I4	-2147483648 to 2147483647 for each element* -2147483648 to 2147483647 for each element*
Real	FREAL*m* MREAL*m*		VT_R4 VT_R4	+/-1.2e-38 to +/-3.4e+38 +/-1.2e-38 to +/-3.4e+38
Real Array	FREALm,v MREALm,v		VT_ARRAY|VT_R4 VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element* +/-1.2e-38 to +/-3.4e+38 for each element*

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

**: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

***: The Communication Driver provides suffixes (S5T, TR, DT, KT, and BCD) that are used to interpret raw bytes as having a special data format. Review the PLC processor documentation, as not all processors have ladder logic internally to interpret those data formats the same way (for example, the S7-1200 PLC).

Where:

x	is the start address, with a range from 0 to 65535.
y	is the bit position, with a range from 0 to 7. 0 is the LSB (Least Significant Bit). 7 is the MSB (Most Significant Bit).
n	is the start address of 2-byte data/2-byte data arrays, with a range from 0 to 65534.

m	is the start address of 4-byte data/4-byte data arrays, with a range from 0 to 65532.
v	is the length of data in elements (an item in an array), with a range from 1 to (net PDU data size/type size - header information).

Note: All flags are **Read/Write**. The longest string or array that can be read in a cyclic service is the length of the PDU size minus 32 bytes. The longest string the InTouch software can process is 131 bytes. The longest string that can be poked is 256 bytes or the PDU size minus 28 bytes, whichever is less. The Communication Driver processes a write (**POKE**) to a Flag Byte.

Input Bytes

The following table summarizes the data format, item or point, suffix, data type, and range for Input Bytes. For suffix definitions, see Conversions and Suffixes of Items (Absolute Addressing).

Data Format	Item/Point	Suffix	Data Type	Range
Bit	I*x.y*		VT_BOOL	0 or 1
	E*x.y*		VT_BOOL	0 or 1
	IX*x.y*		VT_BOOL	0 or 1
	EX*x.y*		VT_BOOL	0 or 1
String	IS*x,v*		VT_BSTR	String
	ES*x,v*		VT_BSTR	String
	ISTRING*x,v*		VT_BSTR	String
	ESTRING*x,v*		VT_BSTR	String
Byte	IB*x*	DT	VT_UI1	0 to 255
	EB*x*		VT_UI1	0 to 255
	IBYTE*x*		VT_UI1	0 to 255
	EBYTE*x*		VT_UI1	0 to 255
			VT_BSTR	1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Byte Array	IB*x,v*		VT_ARRAY|VT_UI1	0 to 255 for each element*
	EB*x,v*		VT_ARRAY|VT_UI1	0 to 255 for each element*
	IBYTE*x,v*		VT_ARRAY|VT_UI1	0 to 255 for each element*
	EBYTE*x,v*		VT_ARRAY|VT_UI1	0 to 255 for each element*

USINT	D<B,I>*d*,USINT*x*	DT	VT_UI1 VT_BSTR	0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
USINT Array	D<B,I>*d*,USINT*x,v*		VT_ARRAY|VT_UI1	0 to 255 for each element*
Char	ICHAR*x* ECHAR*x*	DT	VT_I1 VT_I1 VT_BSTR	-128 to 127 -128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Char Array	ICHAR*x,v* ECHAR*x,v*		VT_ARRAY|VT_I1 VT_ARRAY|VT_I1	-128 to 127 for each element* -128 to 127 for each element*
SINT	D<B,I>*d*,SINT*x*	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
SINT Array	D<B,I>*d*SINT*x,v*		VT_ARRAY|VT_UI1	-128 to 127 for each element*
Word	IW*n* EW*n* IWORD*n* EWORD*n*	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 65535 0 to 65535 0 to 65535 0 to 9999 0.0 to 999.3 0ms to 2h46m30s0ms 0.0 to 9990.0 (s) 1990-1-1 to 2168-12-31
Word Array	IW*n,v* EW*n,v* IWORD*n,v* EWORD*n,v*		VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2	0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element*
UINT	D<B,I>*d*,UINT*n*	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4	0 to 65535 0 to 9999*** 0.0 to 999.3*** 0ms to 2h46m30s0ms*** 0.0 to 9990.0 (s)***

			VT_BSTR	1990-1-1 to 2168-12-31
UINT Array	D<B,I>*d*,UINT*n,v*		VT_ARRAY|VT_UI2	0 to 65535 for each element*
Integer	IINT*n* EINT*n*	BCD D	VT_I2 VT_I2 VT_I2 VT_BSTR	-32768 to 32767 -32768 to 32767 -999 to 999 1990-1-1 to 2168-12-31
Integer Array	IINT*n,v* EINT*n,v*		VT_ARRAY|VT_I2 VT_ARRAY|VT_I2	-32768 to 32767 for each element* -32768 to 32767 for each element*
Double Word	ID*m* ED*m* IDWORD*m* EDWORD*m*	BCD TOD T	VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 99999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Word Array	ID*m,v* ED*m,v* IDWORD*m,v* EDWORD*m,v*		VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4	0 to 4294967295 for each element* 0 to 4294967295 for each element 0 to 4294967295 for each element* 0 to 4294967295 for each element
UDINT	D<B,I>*d*,UDINT*m*	BCD TOD T	VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS

UDINT Array	D<B,I>*d*,UDINT*m,v*		VT_ARRAY|VT_UI4	0 to 4294967295 for each element*
Double Integer	IDINT*m* EDINT*m*	BCD TOD T	VT_I4 VT_I4 VT_I4 VT_BSTR VT_BSTR	-2147483648 to 2147483647 -2147483648 to 2147483647 -9999999 to 9999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_ 648MS to 24D_20H_31M_23S_647MS
Double Integer Array	IDINT*m,v* EDINT*m,v*		VT_ARRAY|VT_I4 VT_ARRAY|VT_I4	-2147483648 to 2147483647 for each element* -2147483648 to 2147483647 for each element*
Real	IREAL*m* EREAL*m*		VT_R4 VT_R4	+/-1.2e-38 to +/-3.4e+38 +/-1.2e-38 to +/-3.4e+38
Real Array	IREAL*m,v* EREAL*m,v*		VT_ARRAY|VT_R4 VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element* +/-1.2e-38 to +/-3.4e+38 for each element

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

**: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

***: The Communication Driver provides suffixes (S5T, TR, DT, KT, and BCD) that are used to interpret raw bytes as having a special data format. Review the PLC processor documentation, as not all processors have ladder logic internally to interpret those data formats the same way (for example, the S7-1200 PLC).

Where:

x	is the start address, with a range from 0 to 65535.
y	is the bit position, with a range from 0 to 7. 0 is the LSB (Least Significant Bit). 7 is the MSB (Most Significant Bit).
n	is the start address of 2-byte data/2-byte data arrays, with a range from 0 to 65534.
m	is the start address of 4-byte data/4-byte data arrays, with a range from 0 to 65532.
v	is the length of data in elements (an item in an array), with a range from 1 to (net PDU data size/type size - header information).

Note: All inputs are **Read-Only**. The longest string or array that can be read in a cyclic service is the length of the PDU size minus 32 bytes. The longest string the InTouch software can process is 131 bytes. The Communication Driver does **not** process a write (**POKE**) to an Input Byte.

Output Bytes

The following table summarizes data format, item or point, suffix, data type, and range for Output Bytes. See Conversions and Suffixes of Items (Absolute Addressing) for suffix definitions.

Data Format	Item/Point	Suffix	Data Type	Range
Bit	Ox.y		VT_BOOL	0 or 1
	Ax.y		VT_BOOL	0 or 1
	Qx.y		VT_BOOL	0 or 1
	OXx.y		VT_BOOL	0 or 1
	AXx.y		VT_BOOL	0 or 1
	QXx.y		VT_BOOL	0 or 1
String	OSx,v		VT_BSTR	String
	ASx,v		VT_BSTR	String
	QSx,v		VT_BSTR	String
	OSTRINGx,v		VT_BSTR	String
	ASTRINGx,v		VT_BSTR	String
	QSTRINGx,v		VT_BSTR	String
Byte	OBx	DT	VT_UI1	0 to 255
	ABx		VT_UI1	0 to 255
	QBx		VT_UI1	0 to 255
	OBYTEx		VT_UI1	0 to 255
	ABYTEx		VT_UI1	0 to 255
	QBYTEx		VT_UI1	0 to 255
			VT_BSTR	1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999

Byte Array	OBx,v ABx,v QBx,v OBYTEx,v ABYTEx,v QBYTEx,v		VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1	0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element*
USINT	D<B,I>d,USINTx	DT	VT_UI1 VT_BSTR	0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
USINT Array	D<B,I>d,USINTx,v		VT_ARRAY|VT_UI1	0 to 255 for each element*
Char	OCHARx ACHARx QCHARx	DT	VT_I1 VT_I1 VT_I1 VT_BSTR	-128 to 127 -128 to 127 -128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999

Char Array	OCHARx,v ACHARx,v QCHARx,v		VT_ARRAY|VT_I1 VT_ARRAY|VT_I1 VT_ARRAY|VT_I1	-128 to 127 for each element* -128 to 127 for each element* -128 to 127 for each element*
SINT	D<B,I>d,SINTx	DT	VT_I1 VT_BSTR	128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
SINT Array	D<B,I>dSINTx,v		VT_ARRAY|VT_UI1	-128 to 127 for each element*

Word	OWn	BCD	VT_UI2	0 to 65535
	AWn	KT	VT_UI2	0 to 65535
	QWn	S5T	VT_UI2	0 to 65535
	OWORDn	TR	VT_UI2	0 to 65535
	AWORDn	D	VT_UI2	0 to 65535
	QWORDn		VT_UI2	0 to 65535
			VT_UI2	0 to 9999
			VT_BSTR	0.0 to 999.3
			VT_BSTR	0ms to 2h46m30s0ms
			VT_R4	0.0 to 9990.0 (s)
			VT_BSTR	1990-1-1 to 2168-12-31
Word Array	OWn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element*
	AWn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element
	QWn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element
	OWORDn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element*
	AWORDn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element
	QWORDn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element
UINT	D<B,I>d,UINTn	BCD	VT_UI2	0 to 65535
		KT	VT_UI2	0 to 9999***
		S5T	VT_BSTR	0.0 to 999.3***
		TR	VT_BSTR	0ms to 2h46m30s0ms***
		D	VT_R4	0.0 to 9990.0 (s)***
			VT_BSTR	1990-1-1 to 2168-12-31

UINT Array	D<B,I>d,UINTn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element*
Integer	OINTn AINTn QINTn	BCD D	VT_I2 VT_I2 VT_I2 VT_BSTR	-32768 to 32767 -32768 to 32768 -32768 to 32768 -999 to 999 1990-1-1 to 2168-12-31
Integer Array	OINTn,v AINTn,v QINTn,v		VT_ARRAY|VT_I2 VT_ARRAY|VT_I2 VT_ARRAY|VT_I2	-32768 to 32767 for each element* -32768 to 32767 for each element* -32768 to 32767 for each element*

Double Word	ODm ADm QDm ODWORDm ADWORDm QDWORDm	BCD TOD T	VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 99999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Word Array	ODm,v ADm,v QDm,v ODWORDm,v ADWORDm,v QDWORDm,v		VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4	0 to 4294967295 for each element* 0 to 4294967295 for each element* 0 to 4294967295 for each element* 0 to 4294967295 for each element* 0 to 4294967295 for each element* 0 to 4294967295 for each element*
UDINT	D<B,I>d,UDINTm	BCD TOD T	VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS

UDINT Array	D<B,I>d,UDINTm,v		VT_ARRAY|VT_UI4	0 to 4294967295 for each element*
Double Integer	ODINTm ADINTm QDINTm	BCD TOD T	VT_I4 VT_I4 VT_BSTR VT_BSTR	-2147483648 to 2147483647 -9999999 to 9999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Integer Array	ODINTm,v ADINTm,v QDINTm,v		VT_ARRAY|VT_I4	-2147483648 to 2147483647 for each element*
Real	OREALm AREALm QREALm		VT_R4	+/-1.2e-38 to +/-3.4e+38

Real Array	OREALm,v AREALm,v QREALm,v		VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

**: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

***: The Communication Driver provides suffixes (S5T, TR, DT, KT, and BCD) that are used to interpret raw bytes as having a special data format. Review the PLC processor documentation, as not all processors have ladder logic internally to interpret those data formats the same way (for example, the S7-1200 PLC).

Where:

x	is the start address, with a range from 0 to 65535.
y	is the bit position, with a range from 0 to 7. 0 is the LSB (Least Significant Bit). 7 is the MSB (Most Significant Bit).
n	is the start address of 2-byte data/2-byte data arrays, with a range from 0 to 65534.
m	is the start address of 4-byte data/4-byte data arrays, with a range from 0 to 65532.
v	is the length of data in elements (an item in an array), with a range from 1 to (net PDU data size/type size - header information).

Note: All outputs are Read/Write. The longest string or array that can be read in a cyclic service is the length of the PDU size minus 32 bytes. The longest string the InTouch software can process is 131 bytes. The longest string that can be poked is 256 bytes or the PDU size minus 28 bytes, whichever is less. The Communication Driver processes a write (POKE) to an Output Byte.

Peripheral Input Bytes

The following table summarizes the data format, item or point, suffix, data type, and range for Peripheral Input Bytes. See Conversions and Suffixes of Items (Absolute Addressing) for suffix definitions.

Data Format	Item/Point	Suffix	Data Type	Range

Bit	PIx.y PEx.y PIXx.y PEXx.y		VT_BOOL VT_BOOL VT_BOOL VT_BOOL	0 or 1 0 or 1 0 or 1 0 or 1
String	PISx,v PESx,v PISTRINGx,v PESTRINGx,v		VT_BSTR VT_BSTR VT_BSTR VT_BSTR	String String String String
Byte	PIBx PEBx PIBYTEx PEBYTEx	DT	VT_UI1 VT_UI1 VT_UI1 VT_UI1 VT_BSTR	0 to 255 0 to 255 0 to 255 0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Byte Array	PIBx,v PEBx,v PIBYTEx,v PEBYTEx,v		VT_ARRAY| VT_UI1 VT_ARRAY| VT_UI1 VT_ARRAY| VT_UI1 VT_ARRAY| VT_UI1	0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element* 0 to 255 for each element*
USINT	D<B,I>d,USINTx	DT	VT_UI1 VT_BSTR	0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
USINT Array	D<B,I>d,USINTx,v		VT_ARRAY| VT_UI1	0 to 255 for each element*
Char	PICHARx PECHARx	DT	VT_I1 VT_I1 VT_BSTR	-128 to 127 -128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Char Array	PICHARx,v PECHARx,v		VT_ARRAY| VT_I1 VT_ARRAY| VT_I1	-128 to 127 for each element* -128 to 127 for each element*

SINT	D<B,I>d,SINTx	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
SINT Array	D<B,I>dSINTx,v		VT_ARRAY| VT_UI1	-128 to 127 for each element*
Word	PIWn PEWn PIWORDn PEWORDn	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 65535 0 to 65535 0 to 65535 0 to 9999 0.0 to 999.3 0ms to 2h46m30s0ms 0.0 to 9990.0 (s) 1990-1-1 to 2168-12-31
Word Array	PIWn,v PEWn,v PIWORDn,v PEWORDn,v		VT_ARRAY| VT_UI2 VT_ARRAY| VT_UI2 VT_ARRAY| VT_UI2 VT_ARRAY| VT_UI2	0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element*
UINT	D<B,I>d,UINTn	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 9999*** 0.0 to 999.3*** 0ms to 2h46m30s0ms*** 0.0 to 9990.0 (s)*** 1990-1-1 to 2168-12-31
D<B,I>d,UINTn,v	D<B,I>d,UINTn,v		VT_ARRAY| VT_UI2	0 to 65535 for each element*
Integer	PIINTn PEINTn	BCD D	VT_I2 VT_I2 VT_I2 VT_BSTR	-32768 to 32767 -32768 to 32767 -999 to 999 1990-1-1 to 2168-12-31

Integer Array	PIINTn,v PEINTn,v		VT_ARRAY| VT_I2 VT_ARRAY| VT_I2	-32768 to 32767 for each element* -32768 to 32767 for each element*
Double Word	PIDm PEDm PIDWORDm PEDWORDm	BCD TOD T	VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 99999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Word Array	PIDm,v PEDm,v PIDWORDm,v PEDWORDm,v		VT_ARRAY| VT_UI4 VT_ARRAY| VT_UI4 VT_ARRAY| VT_UI4 VT_ARRAY| VT_UI4	0 to 4294967295 for each element* 0 to 4294967295 for each element 0 to 4294967295 for each element 0 to 4294967295 for each element
UDINT	D<B,I>d,UDINTm	BCD TOD T	VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
UDINT Array	D<B,I>d,UDINTm,v		VT_ARRAY| VT_UI4	0 to 4294967295 for each element*
Double Integer	PIDINTm PEDINTm	BCD TOD T	VT_I4 VT_I4 VT_I4 VT_BSTR VT_BSTR	-2147483648 to 2147483647 -2147483648 to 2147483647 -9999999 to 9999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Integer Array	PIDINTm,v PEDINTm,v		VT_ARRAY| VT_I4	-2147483648 to 2147483647 for each element*

Real	PIREALm PEREALm		VT_R4 VT_R4	+/-1.2e-38 to +/-3.4e+38 +/-1.2e-38 to +/-3.4e+38
Real Array	PIREALm,v PEREALm,v		VT_ARRAY|VT_R4 VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element* +/-1.2e-38 to +/-3.4e+38 for each element

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

**: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

***: The SIDirect Communication Driver documentation, as not all processors have ladder logic internally Communication Driverto interpret those data formats the same way (for example, the S7-1200 PLC).

Where:

x	is the start address, with a range from 0 to 65535.
y	is the bit position, with a range from 0 to 7. 0 is the LSB (Least Significant Bit). 7 is the MSB (Most Significant Bit).
n	is the start address of 2-byte data/2-byte data arrays, with a range from 0 to 65534.
m	is the start address of 4-byte data/4-byte data arrays, with a range from 0 to 65532.
v	is the length of data in elements (an item in an array), with a range from 1 to (net PDU data size/type size - header information).

Note: All peripheral inputs are **Read-Only**. The longest string or array that can be read in a cyclic service is the length of the PDU size minus 32 bytes. The longest string the InTouch software can process is 131 bytes. The longest string that can be poked is 256 bytes or the PDU size minus 28 bytes, whichever is less. The Communication Driver does **not** process a write (**POKE**) to a Peripheral Input Byte. Some input modules are not readable.

Peripheral Output Bytes

The following table summarizes the data format, item or point, suffix, data type, and range for Peripheral Input Bytes. See Conversions and Suffixes of Items (Absolute Addressing) for suffix definitions.

Data Format	Item/Point	Suffix	Data Type	Range

Bit	POx.y		VT_BOOL	0 or 1
	PAx.y		VT_BOOL	0 or 1
	PQx.y		VT_BOOL	0 or 1
	POXx.y		VT_BOOL	0 or 1
	PAXx.y		VT_BOOL	0 or 1
	PQXx.y		VT_BOOL	0 or 1
String	POSx.v		VT_BSTR	String
	PASx.v		VT_BSTR	String
	PQSx.v		VT_BSTR	String
	POSTRINGx.v		VT_BSTR	String
	PASTRINGx.v		VT_BSTR	String
	PQSTRINGx.v		VT_BSTR	String
Byte	POBx	DT	VT_UI1	0 to 255
	PABx		VT_UI1	0 to 255
	PQBx		VT_UI1	0 to 255
	POBYTEx		VT_UI1	0 to 255
	PABYTEx		VT_UI1	0 to 255
	PQBYTEx		VT_UI1	0 to 255
			VT_BSTR	1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Byte Array	POBx.v		VT_ARRAY|VT_UI1	0 to 255 for each element*
	PABx.v		VT_ARRAY|VT_UI1	0 to 255 for each element*
	PQBx.v		VT_ARRAY|VT_UI1	0 to 255 for each element*
	POBYTEx.v		VT_ARRAY|VT_UI1	0 to 255 for each element*
	PABYTEx.v		VT_ARRAY|VT_UI1	0 to 255 for each element*
	PQBYTEx.v		VT_ARRAY|VT_UI1	0 to 255 for each element*
			VT_ARRAY|VT_UI1	
			VT_ARRAY|VT_UI1	
			VT_ARRAY|VT_UI1	
USINT	D<B,I>d,USINTx	DT	VT_UI1	0 to 255
			VT_BSTR	1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***

USINT Array	D<B,I>d,USINTx,v		VT_ARRAY| VT_UI1	0 to 255 for each element*
Char	PACHARx POCHARx PQCHARx	DT	VT_I1 VT_I1 VT_I1 VT_BSTR	-128 to 127 -128 to 127 -128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Char Array	POCHARx.v PACHARx.v PQCHARx.v		VT_ARRAY| VT_I1 VT_ARRAY| VT_I1 VT_ARRAY| VT_I1	0 to 255 0 to 255 0 to 255 0 to 255 0 to 255 0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
SINT	D<B,I>d,SINTx	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
SINT Array	D<B,I>dSINTx,v		VT_ARRAY| VT_UI1	-128 to 127 for each element*
Word	POWn PAWn PQWn POWORDn PAWORDn PQWORDn	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 65535 0 to 65535 0 to 65535 0 to 65535 0 to 65535 0 to 9999 0.0 to 999.3 0ms to 2h46m30s0ms 0.0 to 9990.0 (s) 1990-1-1 to 2168-12-31

Word Array	POWn,v PAWn,v PQWn,v POWORDn,v PAWORDn,v PQWORDn,v		T_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2	0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element* 0 to 65535 for each element*
UINT	D<B,I>d,UINTn	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 9999*** 0.0 to 999.3*** 0ms to 2h46m30s0ms*** 0.0 to 9990.0 (s)*** 1990-1-1 to 2168-12-31
UINT Array	D<B,I>d,UINTn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element*
Integer	POINTn PAINTn PQINTn	BCD D	VT_I2 VT_I2 VT_I2 VT_I2 VT_BSTR	-32768 to 32767 -32768 to 32767 -32768 to 32767 -999 to 999 1990-1-1 to 2168-12-31
Integer Array	POINTn,v PAINTn,v PQINTn,v		VT_ARRAY|VT_I2 VT_ARRAY|VT_I2 VT_ARRAY|VT_I2	-32768 to 32767 for each element* -32768 to 32767 for each element* -32768 to 32767 for each element*

Double Word	PODm PADm PQDm PODWORDm PADWORDm PQDWORDm	BCD TOD T	VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 4294967295** 0 to 99999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Word Array	PODm,v PADm,v PQDm,v		VT_ARRAY| VT_UI4 VT_ARRAY| VT_UI4 VT_ARRAY| VT_UI4	0 to 4294967295 for each element* 0 to 4294967295 for each element* 0 to 4294967295 for each element*
UDINT	D<B,I>d,UDINTm	BCD TOD T	VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
UDINT Array	D<B,I>d,UDINTm,v		VT_ARRAY| VT_UI4	0 to 4294967295 for each element*
Double Integer	PODINTm PADINTm PQDINTm	BCD TOD T	VT_I4 VT_I4 VT_I4 VT_I4 VT_BSTR VT_BSTR	-2147483648 to 2147483647 -2147483648 to 2147483647 -2147483648 to 2147483647 -9999999 to 9999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Integer Array	PODINTm,v PADINTm,v PQDINTm,v		VT_ARRAY| VT_I4 VT_ARRAY| VT_I4 VT_ARRAY| VT_I4	-2147483648 to 2147483647 for each element* -2147483648 to 2147483647 for each element* -2147483648 to 2147483647 for each element*

Real	POREALm PAREALm PQREALm		VT_R4 VT_R4 VT_R4	+/-1.2e-38 to +/-3.4e+38 +/-1.2e-38 to +/-3.4e+38 +/-1.2e-38 to +/-3.4e+38
Real Array	POREALm,v PAREALm,v PQREALm,v		VT_ARRAY| VT_R4 VT_ARRAY| VT_R4 VT_ARRAY| VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element* +/-1.2e-38 to +/-3.4e+38 for each element* +/-1.2e-38 to +/-3.4e+38 for each element*

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

**: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

Where:

x	is the start address, with a range from 0 to 65535.
y	is the bit position, with a range from 0 to 7. 0 is the LSB (Least Significant Bit). 7 is the MSB (Most Significant Bit).
n	is the start address of 2-byte data/2-byte data arrays, with a range from 0 to 65534.
m	is the start address of 4-byte data/4-byte data arrays, with a range from 0 to 65532.
v	is the length of data in elements (an item in an array), with a range from 1 to (net PDU data size/type size - header information).

***: The Communication Driver provides suffixes (S5T, TR, DT, KT, and BCD) that are used to interpret raw bytes as having a special data format. Review the PLC processor documentation, as not all processors have ladder logic internally to interpret those data formats the same way (for example, the S7-1200 PLC).

Note: All peripheral outputs are **Write-Only**. The longest string or array that can be read in a cyclic service is the length of the PDU size minus 32 bytes. The longest string the InTouch software can process is 131 bytes. The longest string that can be poked is 256 bytes or the PDU size minus 28 bytes, whichever is less. All output modules are not readable. Only POKES are allowed.

Counters

The following table summarizes data format, item or point, suffix, data type, and range for Counters.

Data Format	Item/Point	Suffix	Data Type	Range

Word	C*x*	None	VT_UI2	0…65535
	Z*x*	None	VT_UI2	0 to 65535
		BCD	VT_UI2	0 to 9999
		KT	VT_BSTR	0.0 to 999.3
		S5T	VT_BSTR	0ms to 2h46m30s

Where:

x is the start address, with a range from 0 to 65535.

When the suffix is used, the client is responsible for ensuring the correct value is specified or returned.

Note: S7-1200 and S7-1500 PLCs do not support "direct" counter\timer access by the client/server. That is, you cannot use the current S7-300/400 item name Cx or Tx and its associated suffixes to subscribe to counter\timer values from the S7-1200 and S7-1500 PLCs. However, you can use the database items to indirectly output the associated system counter\timer value (internal IEC counter\timer) from the PLC.

For example:

If the value of counter C0 is 42, the value read by just using the "C0" item syntax is 42. However, if the item syntax is "C0 BCD," the value returned by the Communication Driver is 66.

A similar principle applies to poking:

If the value for the counter is 42, using the "C0" pokes a value of 42 into the counter C0. However, with the "C0 BCD" syntax, the poke value is 66.

Note: All counters are **Read/Write**. The Communication Driver processes a write (**POKE**) to a counter. Although the Communication Driver allows poking any word value into counters, the S7 PLC can only process values in the range of 0…2457 or 0…999 (BCD).

Timers

The following table summarizes data format, item or point, suffix, data type, and range only for the S7-300, S7-400, and S7-1200 Timers.

Data Format	**Item/Point**	**Suffix**	**Data Type**	**Range**
Word	Tx	None	VT_UI2	0 to 14745
	TREALx	BCD	VT_UI2	0 to 9999
		KT	VT_BSTR	0.0 to 999.3
		S5T	VT_BSTR	0ms to 2h46m30s0ms
		None	VT_R4	0.0 to 9990.00

Where:

x is the start address, with a range from 0 to 65535.

When the suffix is used, the client is responsible for ensuring the correct value is specified or returned.

Note: S7-1200 and S7-1500 PLCs do not support "direct" counter\timer access by the client/server. That is, you cannot use the current S7-300/400 item name C*x* or T*x* and its associated suffixes to subscribe to counter\timer values from the S7-1200 and S7-1500 PLCs. However, you can use the database items to indirectly output the associated system counter\timer value (internal IEC counter\timer) from the PLC.

For example:

If the value of timer T0 is 42, the value read by just using the "T0" item syntax is 42. However, if the item syntax is "T0BCD," the value returned by the Communication Driver is 66.

A similar principle applies to poking:

If the value for the timer is 42, using the "T0" pokes a value of 42 into the timer T0. However, with the "T0BCD" syntax, the poke value is 66.

Note: All timers are **Read/Write**. The Communication Driver processes a write (**POKE**) to a timer. Although the Communication Driver allows poking any word value into timers, the S7 PLC can only process values that represent a valid time format.

Block Items

The SIDirect Communication Driver supports Block Items for the S7-300, S7-400, and S7-1500 PLCs. The server does not support Block Items for the S7-1200 PLCs.

The Block Items have two sets of items:

- Read-Only Block Items
- Write-Only Block Items

Read-Only Block Items

The following table summarizes the data format, item or point, suffix, data type, and range for Read-Only Block Items. See Conversions and Suffixes of Items (Absolute Addressing) for suffix definitions.

Data Format	Item/Point	Suffix	Data Type	Range
Bit	BLd,Xx.y		VT_BOOL	0 or 1
String	BLd,Sx,v BLd,STRINGx,v		VT_BSTR VT_BSTR	String String
Byte	BLd,Bx BLd,BYTEx	DT	VT_UI1 VT_UI1 VT_BSTR	0 to 255 0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Byte Array	BLd,Bx,v BLd,BYTEx,v		VT_ARRAY}VT_UI1 VT_ARRAY|VT_UI1	0 to 255 for each element* 0 to 255 for each element*
USINT	D<B,I>d,USINTx	DT	VT_UI1 VT_BSTR	0 to 255 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***

USINT Array	D<B,I>d,USINTx,v		VT_ARRAY|VT_UI1	0 to 255 for each element*
Char	BLd,CHARx	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999
Char Array	BLd,CHARx,v		VT_ARRAY|VT_I1	-128 to 127 for each element*
SIN T	D<B,I>d,SINTx	DT	VT_I1 VT_BSTR	-128 to 127 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999***
SINT Array	D<B,I>dSINTx,v		VT_ARRAY|VT_UI1	-128 to 127 for each element*
Word	BLd,Wn BLd,WORDn	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 65535 0 to 9999 0.0 to 999.3 0ms to 2h46m30s0ms 0.0 to 9990.0 (s) 1990-1-1 to 2168-12-31
Word Array	BLd,Wn,v BLd,WORDn,v		VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2	0 to 65535 for each element* 0 to 65535 for each element*
UINT	D<B,I>d,UINTn	BCD KT S5T TR D	VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_R4 VT_BSTR	0 to 65535 0 to 9999*** 0.0 to 999.3*** 0ms to 2h46m30s0ms*** 0.0 to 9990.0 (s)*** 1990-1-1 to 2168-12-31
UINT Array	D<B,I>d,UINTn,v		VT_ARRAY|VT_UI2	0 to 65535 for each element*
Integer	BLd,INTn	BCD D	VT_I2 VT_I2 VT_BSTR	-32768 to 32767 -999 to 999 1990-1-1 to 2168-12-31

Integer Array	BLd,INTn,v		VT_BSTR	-32768 to 32767 for each element*
Double Word	BLd,Dm BLd,DWORDm	BCD TOD T	VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 4294967295** 0 to 99999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Word Array	BLd,Dm,v BLd,DWORDm,v		VT_ARRAY|VT_UI4	0 to 4294967295 for each element* 0 to 4294967295 for each element**
UDINT	D<B,I>d,UDINTm	BCD TOD T	VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295** 0 to 99999999*** 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
UDINT Array	D<B,I>d,UDINTm,v		VT_ARRAY|VT_UI4	0 to 4294967295 for each element*
Double Integer	BLd,DINTm	BCD TOD T	VT_I4 VT_I4 VT_BSTR VT_BSTR	-2147483648 to 2147483647 -9999999 to 9999999 0:00:00.000 to 23:59:59.999 -24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS
Double Integer Array	BLd,DINTm,v		VT_ARRAY|VT_I4	-2147483648 to 2147483647 for each element
Real	BLd,REALm		VT_R4	+/-1.2e-38 to +/-3.4e+38
Real Array	BLd,REALm,v		VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element*

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

**: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

Where:

x	is the start address, with a range from 0 to 65535.
y	is the bit position, with a range from 0 to 7. 0 is the LSB (Least Significant Bit). 7 is the MSB (Most Significant Bit).
n	is the start address of 2-byte data/2-byte data arrays, with a range from 0 to 65534.
m	is the start address of 4-byte data/4-byte data arrays, with a range from 0 to 65532.
v	is the length of data in elements (an item in an array), with a range from 1 to (net PDU data size/type size - header information).

Note: All Block items are **Read-Only**. The longest string or array that can be read in a block service is the length of 65534 bytes. The longest string the InTouch software can process is 131 bytes.

Write-Only Block Items

The following table summarizes the data format, item or point, suffix, data type, and range for Write-Only Block Items. See Conversions and Suffixes of Items (Absolute Addressing) for suffix definitions.

Data Format	Item/Point	Suffix	Data Type	Range
Byte Array	BWd BWCd-q,x		VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1	0 to 255 for each element*,** **Note:** The Communication Driver does not cache the value of the item written to the PLC. 0 to 255 for each element*,*** **Note:** The Communication Driver caches the value of the item written to the PLC.
	BWCd-q.Send		VT_BOOL	TRUE (1)***,**** FALSE (0)***,**** The value of item (BWCd-q.x), cached by the Communication Driver, is sent to the PLC when this item is transitioned from FALSE to TRUE. The value of this item remains TRUE until the appropriate

				acknowledgment is received from the PLC. After the acknowledgment is received from the PLC, the value of this item is set to 0.

*: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

**: The starting address is always at 0 for Writes. The number of bytes written to the PLC block is determined from the length of the input byte stream. However, the length of the byte stream must be less than 65535.

***: The Siemens S7 block write protocol BSEND does not allow a starting address. All block writes to the PLC always start at address 0. To circumvent this limitation, the BWC item syntax allows for a starting address. The logical number of bytes written to the PLC block is determined from the difference in the start address and end address, (q-x+1), or the length of the input byte stream.

If the input byte stream is longer than the (q-x+1), only (q-x+1) bytes are written.

If the input byte stream is shorter than (q-x+1), the whole input byte stream is written.

Internally, the Communication Driver allocates cache buffers based on the PLC data block and the end address. For more information, see Block Write Caching.

****: The BWCd-q.Send item is readable to allow monitoring of the status of the block send function.

Where:

d	is the block ID, in decimal, with a range from 0 to 4294967295.
x	is the start address, with a range from 0 to 65535.
q	is the end address, with a range from 0 to 65535. It must be equal or greater than x.

Note: All BW and BWC block items are **Write-Only**. BWCxx.Send items are **Read/Write**. The longest string or array that can be written in a block service is the length of 65534 bytes. The longest string the InTouch software can process is 131 bytes.

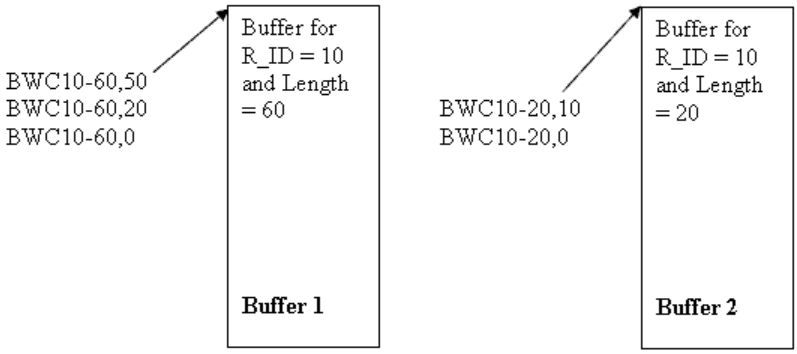
Block Write Caching

BWC is a special item that caches the data before sending it to the PLC. The item syntax for the BWC item is:

```
BWC<R_ID>-<length>,<start_address>
```

The R_ID is the ID that is configured for the Block Write in the PLC. A caching buffer is created, based on R_ID and length as an index, inside the server whenever you advise the BWC item. The buffer is not created if one already exists for the same R_ID and length.

For example, you advise the following items: BWC10-60,50, BWC10-60,20, BWC10-60,0, BWC10-20,10 and BWC10-20,0. Only two buffers are created inside the server for these items:



Consider following scenarios:

1. You poke 20 bytes of data in item BWC10-60,0. Buffer-1 is filled from byte 0 to 19.
2. You poke 15 bytes of data in item BWC10-60,50. Buffer-1 is filled from byte 50 to 59. The last 5 bytes are truncated because only 10 bytes are available in that offset.
3. You poke 30 bytes of data in item BWC10-60,20. Buffer-1 is filled from byte 20 to 49.
4. You poke 20 bytes of data in item BWS10-20,0. Buffer-2 is filled from byte 0 to 19.
5. You poke 1 to item BWC10-20.Send. Buffer-2 is flushed to the PLC with byte 20.
6. You poke 1 to item BWC10-60.Send. Buffer-1 is flushed to the PLC with byte 60.

Alarms and Events

Alarm and event information can be received from the S7-300 and S7-400 PLCs. The item syntax for Alarms and Events is as follows:

ALARM<*EV_ID*>.<Extension 1>[,<Extension 2>[<Suffix>]]

EVENT<*EV_ID*>.<Extension 1>[,<Extension 2>[<Suffix>]]

Note: The S7-1200 and S7-1500 PLCs do not provide alarms and events capability.

The following table shows valid values and valid value combinations for Extension 1, Extension 2, and Suffix.

Item:ALARM<EV_ID>

Extension 1	**Extension 2**	**Suffix**	**Data Type**	**Range**
EVENT_ STATE			VT_UI2	0 to 65535
STATE			VT_UI2	0 to 65535
ACK_STATE			VT_UI2	0 to 65535
TIME_ STAMP			VT_BSTR	String*, ****
NO_ADD_ VALUES			VT_UI2	0 to 10
ADD_VALUE*w*	DATA_TYPE		VT_BSTR	String
	LENGTH		VT_UI2	0 to 65535

	X*x.y*		VT_BOOL	0 or 1
	S*x,v* STRING*x,v*		VT_BSTR VT_BSTR	String String
	B*x* BYTE*x*	DT	VT_UI1 VT_UI1 VT_BSTR	0 to 255 0 to 255 String
	B*x,v* BYTE*x,v*		VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1	0 to 255 for each element** 0 to 255 for each element**
	CHAR*x*	DT	VT_I1 VT_BSTR	-128 to 127 String
	CHAR*x,v*		VT_ARRAY|VT_I1	-128 to 127 for each element**
	W*n* WORD*n*	BCD KT S5T D	VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_BSTR	0 to 65535 0 to 65535 0 to 9999 0.0 to 999.3 0ms to 2h46m30s0ms String
	W*n,v* WORD*n,v*		VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2	0 to 65535 for each element** 0 to 65535 for each element**
	INT*n*	BCD D	VT_I2 VT_I2 VT_BSTR	-32768 to 32767 0 to 9999 String
	INT*n,v*		VT_ARRAY|VT_I2	-32768 to 32767 for each element**

	D*m* DWORD*m*	BCD T TOD	VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295*** 0 to 4294967295*** 0 to 99999999 String String
	D*m,v* DWORD*m,v*		VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4	0 to 4294967295 for each element** 0 to 4294967295 for each element**
	DINT*m* DINT*m,v*	BCD T TOD	VT_I4 VT_I4 VT_BSTR VT_BSTR VT_ARRAY|VT_I4	-2147483648 to 2147483647 0 to 99999999 String String -2147483648 to 2147483647 for each element**
	REAL*m*		VT_R4	+/-1.2e-38 to +/-3.4e+38
	REAL*m,v*		VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element**

Item:ALARM<EV_ID>

Extension 1	**Extension 2**	**Suffix**	**Data Type**	**Range**
EVENT_ STATE			VT_UI2	0 to 65535
STATE			VT_UI2	0 to 65535
ACK_STATE			VT_UI2	0 to 65535
TIME_ STAMP			VT_BSTR	String*, ****
NO_ADD_ VALUES			VT_UI2	0 to 10
ADD_VALUE*w*	DATA_TYPE		VT_BSTR	String
	LENGTH		VT_UI2	0 to 65535
	X*x.y*		VT_BOOL	0 or 1

	S*x,v* STRING*x,v*		VT_BSTR VT_BSTR	String String
	B*x* BYTE*x*	DT	VT_UI1 VT_UI1 VT_BSTR	0 to 255 0 to 255 String
	B*x,v* BYTE*x,v*		VT_ARRAY|VT_UI1 VT_ARRAY|VT_UI1	0 to 255 for each element** 0 to 255 for each element**
	CHAR*x*	DT	VT_I1 VT_BSTR	-128 to 127 String
	CHAR*x,v*		VT_ARRAY|VT_I1	-128 to 127 for each element**
	W*n* WORD*n*	BCD KT S5T D	VT_UI2 VT_UI2 VT_UI2 VT_BSTR VT_BSTR VT_BSTR	0 to 65535 0 to 65535 0 to 9999 0.0 to 999.3 0ms to 2h46m30s0ms String
	W*n,v* WORD*n,v*		VT_ARRAY|VT_UI2 VT_ARRAY|VT_UI2	0 to 65535 for each element** 0 to 65535 for each element**
	INT*n*	BCD D	VT_I2 VT_I2 VT_BSTR	-32768 to 32767 0 to 9999 String
	INT*n,v*		VT_ARRAY|VT_I2	-32768 to 32767 for each element**
	D*m* DWORD*m*	BCD T TOD	VT_UI4 VT_UI4 VT_UI4 VT_BSTR VT_BSTR	0 to 4294967295*** 0 to 4294967295*** 0 to 99999999 String

				String
	D*m,v* DWORD*m,v*		VT_ARRAY|VT_UI4 VT_ARRAY|VT_UI4	0 to 4294967295 for each element** 0 to 4294967295 for each element**
	DINT*m* DINT*m,v*	BCD T TOD	VT_I4 VT_I4 VT_BSTR VT_BSTR VT_ARRAY|VT_I4	-2147483648 to 2147483647 0 to 99999999 String String -2147483648 to 2147483647 for each element**
	REAL*m*		VT_R4	+/-1.2e-38 to +/-3.4e+38
	REAL*m,v*		VT_ARRAY|VT_R4	+/-1.2e-38 to +/-3.4e+38 for each element**

*: Starting with version 1.1, this string value can be used to timestamp other alarm items.

**: For DDE/SuiteLink, the item value is the Hex ASCIi representation of the complete array. The result is one string containing all the elements of the array in the Hex ASCIi representation of the binary data in big-endian format when data is returned to the DDE/SuiteLink layer.

***: For DDE/SuiteLink, this value is restricted to the range of 0 to 2147483647. Values higher than that are clamped to the maximum value of 2147483647 in a SuiteLink or DDE client. In this case, the quality of the item shows "Clamp High."

****: For alarm blocks (such as SFB31, 34, 35) that provide multiple event states in one notification, the timestamp that comes with the notification reflects only the timestamp of the last changing state. This restriction is prescribed by the message that it receives from the PLC.

Where:

<EV_ID>	is the ID defined by Step7, in the integer format, filled with leading zeros up to six (6) characters.
x	is the start address, with a range from 0 to 65535.
y	is the bit position, with a range from 0 to 7. 0 is the LSB (Least Significant Bit). 7 is the MSB (Most Significant Bit).

n	is the start address of 2-byte data/2-byte data arrays, with a range from 0 to 65534.
m	is the start address of 4-byte data/4-byte data arrays, with a range from 0 to 65532.
v	is the length of data in elements (an item in an array), with a range from 1 to (net PDU data size/type size - header information).
w	is the length of the net S7 string-data in characters, with a range from 1 to 10.

Note: All alarms and events are **Read-Only**. The longest string or array that can be read in a cyclic service is the length of the PDU size minus 32 bytes. The longest string the InTouch software can process is 131 bytes. The SIDirect Communication Driver does **not** process writes (**POKES**) to Alarms and Events.

Examples:

ALARM000010.TIME_STAMP

EVENT001234.ADD_VALUE2,LENGTH

ALARM000555.ADD_VALUE10,REAL0

EVENT000001.ADD_VALUE3,D0TOD

Alarm and event information is delivered from the S7-300/400 PLCs in the form of data items with the syntax described above. Alarm provider and alarm acknowledgment functionality is **not** supported.

The configuration of the alarms is first performed in the appropriate function blocks in the S7-300/400 PLCs as follows:

Name	SFB/SFC	S7 CPU
ALARM_SQ	SFC 17	S7-300/400
ALARM_S	SFC 18	S7-300/400
ALARM_DQ	SFC 107	S7-300/400
ALARM_D	SFC 108	S7-300/400
NOTIFY_8P	SFB 31	S7-400
ALARM	SFB 33	S7-400
ALARM_8	SFB 34	S7-400
ALARM_8P	SFB 35	S7-400
NOTIFY	SFB 36	S7-400

Events must be configured in the Symbol Editor.

Alarms and Events Terms

The following table lists the terms available in Alarms and Events Terms and their descriptions.

Term	Description
EVENT_STATE	State of the Alarm/Event itself. If the Alarm/Event is TRUE, then EVENT_STATE is TRUE and vice versa. For more detailed information, see the Siemens Step7 documentation.
STATE	The state in general whether the Alarm/Event is available. Maybe a data block is deleted where a bit should be monitored.
ACK_STATE	The state of the acknowledgment of coming or going Alarms/Events. For more detailed information, see the Siemens Step7 documentation.
TIME_STAMP	The timestamp of the Alarm/Event provided by the PLC.
NO_ADD_VALUES	The number of additional values sent with this Alarm/Event message.
ADD_VALUEw,DATA_TYPE	The data type of a specific additional value of an Alarm/Event.
ADD_VALUEw,LENGTH	The length of a specific additional value of an Alarm/Event.
<EV_ID>	The event ID created automatically by the Step7 programming software. In case of Alarms (FB33 to FB36, SFC17/18/107/108), this is the EV_ID-parameter of the function block. The value of the parameter must be converted from hexadecimal to decimal, and then filled up with leading zeros to the length of 6 characters (for example: EV_ID: DW#16#4E25 => <EV_ID>: 020005). In case of Events (generated by the symbol editor) this is the "Message number." This number is in decimal format and must be filled with leading zeros up to 6 characters (for example: "Message number": 20000 => <EV_ID>: 020000).

Conversions and Suffixes of Items (Absolute Addressing)

This section describes what data-format items and suffixes are converted and what they are converted into. The items and suffixes described here apply to the SIDirect Legacy object (absolute addressing). For items and suffixes that apply to the SIDirect Symbolic object, see Conversions and Suffixes of Items (Symbolic Addressing).

Endian Conversion

In endian conversions, all items with the following data formats are copied in a reverse-byte order to convert the data from the big endian of the PLC to the little endian of the computer:

- Word
- Integer
- Double Word
- Double Integer
- Real

Suffix BCD

All items with the following data formats and suffix BCD are converted from the BCD format into the integer and back:

- Word
- Integer
- Double Word
- Double Integer

Suffix DT

All items with the following data formats and suffix DT (Date and Time) are converted from DT into a message and back to store a value in the range of 1990-1-1-0:00:00.000 to 2089-12-31-23:59:59.999."

- Byte
- Char

This is an 8-byte value (although declared as "byte") that contains both the date and time. In the client, you see a string such as: 1999-12-13-07:06:05.888. The construction is a BCD interpretation. This means that the value in the memory of the PLC (seen as a hex value) represents directly the single "parts" of the string above.

The example above looks like the following in the memory:

0x9912130706058880

The last character ("0" in this example) is not used in this string, but represents the day of the week. If a DT item is poked, the server writes the correct day of the week to the PLC.

Suffix KT

All items with the following data format and suffix KT are converted from KT to a message and back to store a value in the range of 0.0 to 999.3.

- Word

The item contains a time value in the same format as in the old Step-5 PLCs. In the client, you see a string such as: 999.3. The construction is like a BCD interpretation, but the digits are twisted.

The example above looks like the following in the memory of the PLC:

0x3999

Another example, 0x2345, in the memory of the PLC is 345.2 as the item value.

Suffix S5T

All items with the following data format and suffix S5T are converted from S5T to a message and back to store a value in the range of 0ms to 2h46m30s.

- Word

The memory in the PLC is exactly the same as for the KT items, but the presentation is different, although the meaning is the same. This means a memory content of 0x3999 (as in the example for KT) results in the string of 2h46m30s0ms.

The meaning of 999.3 (KT) is:

999	The first three characters are the time value in BCD.
3	The last digit is the multiplier. Possible values are: 0: 0.01s 1: 0.1s 2: 1s 3: 10s.

This means:

A value of 123.0 represents:	123*0.01s=1.23s (equals 1s230ms)
A value of 543.2 represents:	543*1s=543s (equals 9m3s0ms)
A value of 999.3 represents:	999*10s=9990s (equals 2h46m30s0ms)

Suffix TR

All items with the following data format and with suffix TR (Time as real value) are converted from TR into a real value or back to store a value in the range of 0.0 to 9990.0 (s).

- Word

The memory in the PLC is exactly the same as for the KT items, but the presentation is different, although the meaning is the same. The memory content of 0x3999 (as in the example for KT) results in the real value of 9990.0. The construction is the result of the multiplication as described in the examples for S5T, given to the client as a real value.

Suffix D

All items with the following data formats and with suffix D (Date) are converted from D into a message or back to store a value in the range of 1990-1-1 to 2168-12-31.

- Word
- Integer

The item contains the date. The construction is the number of days since 1/1/1990. The integer value 4010, for example, represents 2000-12-24.

Suffix T

All items with the following data formats and with suffix T (Time) are converted from T into a message or back to store a value in the range of
–24D_20H_31M_23S_648MS to 24D_20H_31M_23S_647MS.

- Double Word
- Double Integer

The item contains a time in the IEC format. The client shows a value such as: 3D_7H_32M_11S_153MS. This is the time in milliseconds, shown as a more readable string.
The range from 0 to 2147483647 (0x0 to 0x7FFFFFFF) is interpreted as a positive time value. The range from -2147483648 to -1 (0x80000000 to 0xFFFFFFFF) is interpreted as a negative time value.

Suffix TOD

All items with the following data formats and with suffix TOD (Time Of Day) are converted from TOD into a message or back to store a value in the range of 0:00:00.000 to 23:59:59.999.

- Double Word
- Double Integer

The item contains the time of a day. The client shows a value such as: 4:58:15.654. This is the time in milliseconds (as for T), shown as a more readable string. The highest value is 23:59:59.999. There are no negative values. All values greater than 86399999 (0x05265BFF) are shown with quality 0x0056 (Clamp Hi).

Note: If you use the Communication Driver in the English operating system, the following applies. The string is always represented in a 24-hour format, regardless of the time representation of the operating system used. This means you see the time 1:13:5 P.M. as 13:13:5.0.

LREAL Data Type and Syntax

S7-1500 PLCs contain a data type named LREAL. The LREAL data type is a ANSI/IEEE 754-1985 64-Bit Floating Point value with a range illustrated in the following table:

Data Type	Bit Size	Range
LREAL	64	-1.7976931348623158e+308 to -2.2250738585072014e-308, ±0, +2.2250738585072014e-308 to +1.7976931348623158e+308

S7-1500 PLCs may not support LREAL types on all memory regions.

The item name syntax for LREAL is as follows:

Memory Type	Syntax

Data Block	D<B,I>*d*,LREAL*m* D<B,I>*d*,LREAL*m*,*v*
Flag Bytes	FLREAL*m* MLREAL*m* FLREAL*m*,*v* MLREAL*m*,*v*
Input Bytes	ILREAL*m* ELREAL*m* ILREAL*m*,*v* ELREAL*m*,*v*
Output Bytes	OLREAL*m* ALREAL*m* QLREAL*m* OLREAL*m*,*v* ALREAL*m*,*v* QLREAL*m*,*v*
Peripheral Input Bytes	PILREAL*m* PELREAL*m* PILREAL*m*,*v* PELREAL*m*,*v*
Peripheral Output Bytes	POLREAL*m* PALREAL*m* PQLREAL*m* POLREAL*m*,*v* PALREAL*m*,*v* PQLREAL*m*,*v*

Where:

d is the data block number, with a range from 1 to 65535

m is the start offset of an LREAL tag

v is the length of an array in elements, with a range from 1 to 65535

DTL Data Type and Syntax

The S7-1500 PLCs contain a data type named DTL (Date Time Long). The DTL can be subscribed as a string (VT_BSTR), which provides the full range and resolution of 1 nanosecond, or as a date and time (VT_DATE), which provides the full range but with the highest resolution of 1 second. The DTL data type in the PLC is a 12-byte structure with the following format:

Byte	Field Name	Date Type	Value Range
0,1	Year	UINT	1970 to 2554
2	Month	USINT	1 to 12
3	Day	USINT	1 to 31
4	Weekday	USINT	1 (Sunday) to 7 (Saturday)
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Seconds	USINT	0 to 59
8,9,10,11	Nanoseconds	UDINT	0 to 999,999,999

The Communication Driver reads the 12-byte structure from the S7 PLC and converts it into one of the following canonical types based on the item name suffix specified:

DTL Suffix	Canonical Type	Range
<none>	VT_BSTR	1970-1-1-0:0:0.000000000 to 2554-12-31-23:59:59.999999999 (Highest resolution in units of 1 nanoseconds increments)
TREAL	VT_DATE	1970-1-1-0:0:0 to 2554-12-31-23:59:59 (Highest resolution is in units of 1 sec increments) The syntax for VT_DATE can be YYYY/MM/DD, MM/DD/YYYY, or YYYY-MM-DD.

The item name syntax for DTL is as follows:

Memory Type	**Syntax**	**Canonical Type**
Data Block	D<B,I>*d*,DTL*m*	VT_BSTR
	D<B,I>*d*,DTL*m*TREAL	VT_DATE
	D<B,I>*d*,DTL*m,v*	VT_ARRAY |VT_DATE
	D<B,I>*d*,DTL*m,v*TREAL	VT_ARRAY |VT_DATE
Flag Bytes	FDTL*m*	VT_BSTR
	MDTL*m*	VT_BSTR
	FDTL*m*TREAL	VT_DATE
	MDTL*m*TREAL	VT_DATE
	FDTL*m,v*	VT_ARRAY |VT_BSTR
	MDTL*m,v*	VT_ARRAY |VT_BSTR
	FDTLm,vTREAL	VT_ARRAY |VT_DATE
	MDTLm,vTREAL	VT_ARRAY |VT_DATE
Input Bytes	IDTL*m*	VT_BSTR
	EDTL*m*	VT_BSTR
	IDTL*m*TREAL	VT_DATE
	EDTL*m*TREAL	VT_DATE
	IDTL*m*,*v*	VT_ARRAY |VT_BSTR
	EDTL*m*,*v*	VT_ARRAY |VT_BSTR
	IDTL*m*,*v*TREAL	VT_ARRAY |VT_DATE
	EDTL*m*,*v*TREAL	VT_ARRAY |VT_DATE

Output Bytes	ODTL*m*	VT_BSTR
	ADTL*m*	VT_BSTR
	QDTL*m*	VT_BSTR
	ODTL*m*TREAL	VT_DATE
	ADTL*m*TREAL	VT_DATE
	QDTL*m*TREAL	VT_DATE
	ODTL*m*,*v*	VT_ARRAY |VT_BSTR
	ADTL*m*,*v*	VT_ARRAY |VT_BSTR
	QDTL*m*,*v*	VT_ARRAY |VT_BSTR
	ODTL*m*,*v*TREAL	VT_ARRAY |VT_DATE
	ADTL*m*,*v*TREAL	VT_ARRAY |VT_DATE
	QDTL*m*,*v*TREAL	VT_ARRAY |VT_DATE
Peripheral Input Bytes	PIDTL*m*	VT_BSTR
	PEDTL*m*	VT_BSTR
	PIDTL*m*TREAL	VT_DATE
	PEDTL*m*TREAL	VT_DATE
	PIDTL*m*,*v*	VT_ARRAY |VT_BSTR
	PEDTL*m*,*v*	VT_ARRAY |VT_BSTR
	PIDTL*m*,*v*TREAL	VT_ARRAY |VT_DATE
	PEDTL*m*,*v*TREAL	VT_ARRAY |VT_DATE

Peripheral Output Bytes	PODTL*m*	VT_BSTR
	PADTL*m*	VT_BSTR
	PQDTL*m*	VT_BSTR
	PODTL*m*TREAL	VT_DATE
	PADTL*m*TREAL	VT_DATE
	PQDTL*m*TREAL	VT_DATE
	PODTL*m*,*v*	VT_ARRAY |VT_BSTR
	PADTL*m*,*v*	VT_ARRAY |VT_BSTR
	PQDTL*m*,*v*	VT_ARRAY |VT_BSTR
	PODTL*m*,*v*TREAL	VT_ARRAY |VT_DATE
	PADTL*m*,*v*TREAL	VT_ARRAY |VT_DATE
	PQDTL*m*,*v*TREAL	VT_ARRAY |VT_DATE

Where:

d	is the data block number, with a range from 1 to 65535
m	is the start offset of an DTL tag
v	is the length of an array in elements (an item in an array), with a range from 1 to 65535

DTL Clamping

Any DTL value being read from or poked to the PLC that has a year field less than the minimum or greater than the maximum year results in the date and time being clamped as follows:

For Canonical VT_BSTR:

Year < 1970 = 1970-01-01-00:00:00.000000000

Year > 2554 = 2554-12-31-23:59:59.999999999

For Canonical VT_DATE:

Year < 1970 = 1970-01-01-00:00:00

Year > 2554 = 2554-12-31-23:59:59

If any other field, month, day, hour, min, second, or nanosecond is less than the minimum or greater than the maximum, it results in a rejected poke or read. Also, an uncertain quality results for subscribed tags.

This applies to the syntax of all DTL names.

Using DTL without a Suffix

When poking or subscribing a DTL without a suffix where the RequestedType=VT_BSTR and the CanonicalType=VT_BSTR and you do not enter the entire 9 digits of the nanosecond field, the field is padded to the right with zeros.

The reason for doing this is to maintain the meaning of the decimal place after the seconds field. For example, 2011-10-31-1:1:1.5 results in 500000000 being poked to the nanoseconds field of the DTL, which really means 1/2 a second. To enter 5 nanoseconds, you must poke 2011-10-31-1:1:1.000000005.

All other fields of the date and time are considered to be padded to the left.

Using DTL with a TREAL Suffix

When poking or subscribing a DTL using the TREAL suffix where the RequestedType=VT_BSTR and the CanonicalType=VT_DATE, the string passed between the SIDirect Communication Driver and the client is formatted using the "short date" and "long time" format that you configure in the region and language settings of the computer.

High-Speed Counters

S7-1500 PLCs support High-Speed Counters. After the High-Speed Counters are enabled in the PLC program, they can be accessed through the Input Bytes in DWord format (for example, ID1000, depending on which counter is to be addressed, the configuration of the counter, and the CPU model of the S7-1500 PLC).

For more information about configuring and using the High-Speed Counters, see the S7-1500 PLC documentation.

Generic OPC Syntax

An Communication Driver is a container for OPC Groups, providing 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 devices in the field device. All access to OPC items is maintained through the OPC Group.

The fully qualified name for an OPC item is the Item ID, equivalent to Item Name. The syntax for specifying a unique Item ID is Communication Driver-dependent. In OPC data acquisition Communication Drivers, the syntax can be as follows:

```
TCPIP.PLC1.DB1,B20
```

Where each component (delimited by a period) represents a branch or leaf of the field device's hierarchy.

In this example:

- PLC1 is the name of the target PLC.
- DB1,B20 is the specific data point or 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 data point.
- OPC provides another parameter, called Access Path, that defines optional specifications for obtaining that data.

In Communication Drivers, Access Paths are equivalent to Device Groups. This parameter defines the update interval between the Communication Driver and the field device for accessing the values of data points in the PLC.

VT_Array Syntax in Application Server

VT_Array items from the SIDirect Communication Driver can fail to return good VTQ data in Object Viewer without the addition of an index to the Item/Point syntax. While the syntax provided in this section is correct, VT_Arrays require a [-1] offset. Without this offset, the VT_Array will not count the 0 byte in the range.

Following is an example of the syntax for a VT_Array item to receive proper validation under a LMX client such as Object Viewer in Application Server:

```
S7C_S7_001.SG1.attribute(DB111,WORD10,4)[-1]
```

In this example, the syntax has the following structure:

```
<SIDirect_DiDevice_Instance>.<ScanGroup>.attribute(D<B>d,WORDn,v)[-1]
```

where

- d is the data block number,
- n is the start address of a 2-byte data array, and
- v is the length of data in elements (the size of the array). Each item in an array is called an element.

S7-1500 Item Syntax

The S7-1500 PLCs are a newer series of Siemens S7 PLCs. Its item syntax follows a similar convention as the S7 300/400 series, with some differences. The S7-1500 PLCs support the additional datatypes of LREAL and DTL (Date Time Long).

SIDirect Symbolic Object Reference (Symbolic Addressing)

This section provides reference information specific to the SIDirect Symbolic object. Use the Symbolic object when symbolic addressing is used to communicate with the S7-1500 PLC.

SIDirect Symbolic Naming Convention

The Symbolic interface included with the SIDirect Communication Driver supports symbolic addressing, whether or not Optimized Block Access has been enabled in the Siemens S7 1200 or S7-1500 PLC. The tagnames used in this hierarchy correspond to the symbolic names that you configure in the PLC.

Note: Symbolic addressing is NOT supported through the SIDirect Legacy interface.

The naming conventions used in the following sections about data types are:

[square brackets]	Alphanumeric name is required
< angle brackets >	Alphanumeric name is optional
{ curley brackets }	Naming fields within the brackets can be recursive

Basic Data Types

Basic data types for symbolic addressing include the following items:

- Bit data block items
- Analog data block items

- String data block items
  `[DataBlockName].[SymbolicTagName]`

Example:
```
DB5_INT.TAG_INT
BatchProcess_Block.EmergencyStartStop
```

Complex Data Types

Complex data types for symbolic addressing include the following items:

- Array type items
- Structure data block items
  `[DataBlockName].[SymbolicTagName]<[ArrayIndex]>.<MemberName>`

Array index ranges between -n to n including zero.

Example:
```
DB9_STRING.TAG_STRING
Kettle2.PigmentTube[4]
Data_block_Items.IEC_Timer1.CV
```

User Defined Data Types

User defined data types for symbolic addressing include the following item:

- User Defined Type (UDT) items
  `[DataBlockName].{[StructTagName]<[ArrayIndex]>}.[MemberName]<[ArrayIndex]>`

Example:
```
ProductionRoom.Reactor1.Concentrate_Pump.Motor_OL
FoodPlant.CookDeck[1].IngredientTanks[2,5].InletValve
```

If the Data Block name (in the database) is same as the user defined structure tag name, the item syntax is:
`[DataBlockName].THIS[Array Index].<Member Name>`

Example:

Data Block Name: PUMP

User defined structure tag name: PUMP

Item Syntax is:
```
PUMP.THIS[0].Speed
```

System Data Types

You should observe the following item syntax constraints when using symbolic addressing:

- Input, Output, Memory area items
- Counter and Timer items
  `[PLCTagName]`

Example:

`PLCTag_Byte`	PLC Tag with reference to Memory Area item “%MB1”
`PLCTag_Timer`	PLC Tag with reference to Timer Area item “%T0”

General SIDirect Item Syntax Constraints

You should observe the following item syntax constraints when using symbolic addressing:

- Maximum length of a symbolic name:128 characters in the Data Block Table
- Maximum dimension of an array:6
- Maximum number of elements per array:1600
- Maximum nesting depth of a structure item:8 levels
- Maximum of number of components in a structure:252

Data Types in the TIA Portal

The TIA Portal for S7 1200 or S7-1500 PLCs defines a large number of data types and data formats. These pertain to how the S7 1200 or S7-1500 PLC handles data transfer internally. Data types, data formats, structural elements, suffixes, and value ranges for items using symbolic addressing in S7 1200 or S7-1500 PLCs are described in the following sections.

Data types can be divided into three categories:

- General Data Types: all basic data types defined in the controller for general purpose usage.
- Counter/Timer Data Types: all structures and parameters provided by the system for counters and timers.
- Special Data Types: used for special purposes through specific instructions built into the controller.

General Data Types

General data types include all basic data types defined in the controller for general purpose usage. See Conversions and Suffixes of Items (Symbolic Addressing) for suffix definitions.

Note: Structural element names are predefined by the system and cannot be changed.

Data Format	Structural Element	Suffix	Data Type	Access	Value Range / Notes
Bool			VT_BOOL	Read/Write	FALSE or TRUE
Byte			VT_UI1	Read/Write	0 to 255
Char			VT_UI1	Read/Write	ASCII character set
DInt			VT_I4	Read/Write	-2147483648 to +2147483647
DWord			VT_UI4	Read/Write	0 to 4294967295
Int			VT_I2	Read/Write	-32768 to 32767
LInt			VT_I8	Read/Write	-9223372036854775808 to +9223372036854775807
LReal			VT_R8	Read/Write	-1.7976931348623158e+308 to -2.2250738585072014e-308

					±0,0 +2.2250738585072014e-308 to +1.7976931348623158e+308
LWord			VT_UI8	Read/Write	0 to 18446744073709551615
Real			VT_R4	Read/Write	-3.402823e+38 to -1.175495e-38 ±0,0 +1.175495e-38 to +3.402823e+38
SInt			VT_I1	Read/Write	-128 to 127
String			VT_BSTR	Read/Write	0 to 254 characters
UDInt			VT_UI4	Read/Write	0 to 4294967295
UInt			VT_UI2	Read/Write	0 to 65535
ULInt			VT_UI8	Read/Write	0 to 18446744073709551615
USInt			VT_UI1	Read/Write	0 to 255
WChar			VT_UI2	Read/Write	$0000 - $D7FF
WString			VT_BSTR	Read/Write	0 to 254 characters x. possible values: 0 to 16382
Word			VT_UI2	Read/Write	0 to 65535
Date			VT_UI2	Read/Write	0 to 65535 (days)
		D	VT_BSTR	Read only	1990-01-01 to 2168-12-31
Date_And_Time			VT_Date	Read/Write	1990-01-01-00:00:00.000 to 2089-12-31-23:59:59.999
		DT	VT_BSTR	Read/Write	1990-01-01-00:00:00.000 to 2089-12-31-23:59:59.999 **Note:** DT suffix is required for SuiteLink.
LDT			VT_UI8	Read/Write	0 to 18446744073709551615 (nanoseconds)
		LDT	VT_BSTR	Read/Write	1970-01-01-0:0:0.000000000 to 2263-04-11-23:47:16.854775808

LTime			VT_I8	Read/Write	-9223372036854775808 to +9223372036854775807 (nanoseconds)
		LT	VT_BSTR	Read only	-106751d23h47m16s854ms775us808ns to 106751d23h47m16s854ms775us807ns
LTime_Of_Day			VT_UI8	Read/Write	0 to 18446744073709551615
S5Time			VT_I4	Read/Write	0 to 9990000 (milliseconds)
		S5T	VT_BSTR	Read only	10MS to 9S_990MS (Resolution: 0.01s)
					100MS to 1MIN_39S_900MS (Resolution: 0.1s)
					1S to 16MIN_ 39S (Resolution: 1s)
					0MS to 2H_46M_30S_0MS (Resolution: 10s <Default>)
Time			VT_I4	Read/Write	-2147483648 to +2147483647 (milliseconds)
		T	VT_BSTR	Read only	-24d20h31m23s648ms to +24d20h31m23s647ms
Time_Of_Day			VT_UI4	Read/Write	0 to 4294967295 (milliseconds)
		TOD	VT_BSTR	Read only	00:00:00.000 to 23:59:59.999
DTL *<Structure>*					
UInt	.YEAR		VT_UI2	Read only	1970 to 2262
USInt	.MONTH		VT_UI1	Read only	1 to 12
USInt	.DAY		VT_UI1	Read only	1 to 31
USInt	.WEEKDAY		VT_UI1	Read only	1(Sunday) to 7(Saturday)
USInt	.HOUR		VT_UI1	Read only	0 to 23

USInt	.MINUTE		VT_UI1	Read only	0 to 59
USInt	.SECOND		VT_UI1	Read only	0 to 59
UDInt	.NANOSECOND		VT_UI4	Read only	0 to 999999999

Counter/Timer Data Types

Counter/Timer includes all structures and parameters for counters and timers provided by the system.

Note: Structural element names are predefined by the system and cannot be changed.

Data Format	Structural Element	Suffix	Data Type	Access	Value Range / Notes
Counter			VT_UI2	Read/Write	0 to 999
Timer			VT_I4	Read/Write	0 to 999000
IEC_COUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)
Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)
Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)
Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
Int	.PV		VT_I2	Read/Write	-32768 to 32767 (Preset Counter Value)
Int	.CV		VT_I2	Read/Write	-32768 to 32767 (Current Counter Value)
IEC_DCOUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)

Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)
Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)
Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
DInt	.PV		VT_I4	Read/Write	-2147483648 to +2147483647 (Preset Counter Value)
DInt	.CV		VT_I4	Read/Write	-2147483648 to +2147483647 (Current Counter Value)
IEC_LCOUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)
Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)
Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)
Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
LInt	.PV		VT_I8	Read/Write	-9223372036854775808 to +9223372036854775807 (Preset Counter Value)
LInt	.CV		VT_I8	Read/Write	-9223372036854775808 to +9223372036854775807 (Current Counter Value)
IEC_SCOUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)

Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)
Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)
Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
SInt	.PV		VT_I1	Read/Write	-128 to 127 (Preset Counter Value)
SInt	.CV		VT_I1	Read/Write	-128 to 127 (Current Counter Value)
IEC_UCOUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)
Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)
Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)
Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
UInt	.PV		VT_UI2	Read/Write	0 to 65535 (Preset Counter Value)
UInt	.CV		VT_UI2	Read/Write	0 to 65535 (Current Counter Value)
IEC_UDCOUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)
Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)

Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)
Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
UDInt	.PV		VT_UI4	Read/Write	0 to 4294967295 (Preset Counter Value)
UDInt	.CV		VT_UI4	Read/Write	0 to 4294967295 (Current Counter Value)
IEC_ULCOUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)
Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)
Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)
Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
ULInt	.PV		VT_UI8	Read/Write	0 to 18446744073709551615 (Preset Counter Value)
ULInt	.CV		VT_UI8	Read/Write	0 to 18446744073709551615 (Current Counter Value)
IEC_USCOUNTER			*<Structure>*	N/A	
Bool	.CU		VT_BOOL	Read/Write	FALSE or TRUE (Count Up)
Bool	.CD		VT_BOOL	Read/Write	FALSE or TRUE (Count Down)
Bool	.R		VT_BOOL	Read/Write	FALSE or TRUE (Reset)
Bool	.LD		VT_BOOL	Read/Write	FALSE or TRUE (Load)

Bool	.QU		VT_BOOL	Read/Write	FALSE or TRUE (Status of Up Count)
Bool	.QD		VT_BOOL	Read/Write	FALSE or TRUE (Status of Down Count)
USInt	.PV		VT_UI1	Read/Write	0 to 255 (Preset Counter Value)
USInt	.CV		VT_UI1	Read/Write	0 to 255 (Current Counter Value)
IEC_TIMER			*<Structure>*	N/A	
Time	.ST		VT_I4	Read/Write	-2147483648 to +2147483647 (Start Time)
Time	.PT		VT_I4	Read/Write	-2147483648 to +2147483647 (Present Time)
Time	.ET		VT_I4	Read/Write	-2147483648 to +2147483647 (Elapsed Time)
Bool	.RU		VT_BOOL	Read/Write	FALSE or TRUE (Running Status)
Bool	.IN		VT_BOOL	Read/Write	FALSE or TRUE (Start Input)
Bool	.Q		VT_BOOL	Read/Write	FALSE or TRUE (Pulse Output)
IEC_LTIMER			*<Structure>*	N/A	
LTime	.ST		VT_I8	Read/Write	-9223372036854775808 to +9223372036854775807 (Start Time)
LTime	.PT		VT_I8	Read/Write	-9223372036854775808 to +9223372036854775807 (Present Time)
LTime	.ET		VT_I8	Read/Write	-9223372036854775808 to +9223372036854775807 (Elapsed Time)
Bool	.RU		VT_BOOL	Read/Write	FALSE or TRUE (Running Status)
Bool	.IN		VT_BOOL	Read/Write	FALSE or TRUE (Start Input)
Bool	.Q		VT_BOOL	Read/Write	FALSE or TRUE (Pulse Output)

Special Data Types

Special data types include:

- System data type
- Hardware data type

These are used for special purposes through specific instructions in the controller. Their availability depends on the model of the controller. Some frequently used special data types are listed in the following table.

Note: Structure element names are predefined by the system and cannot be changed.

Data Format	Structural Element	Suffix	Data Type	Access	Value Range / Notes
AOM_IDENT			VT_UI4	Read/Write	0 to 4294967295
CONN_ANY			VT_UI2	Read/Write	0 to 65535
CONN_OUC			VT_UI2	Read/Write	0 to 65535
CONN_PRG			VT_UI2	Read/Write	0 to 65535
CONN_R_ID			VT_UI4	Read/Write	0 to 4294967295
CREF	*<Structure>*		N/A		
Byte	.BLOCK_TYPE		VT_UI1	Read/Write	0 to 255
UInt	.CB_NUMBER		VT_UI2	Read/Write	0 to 65535
UDInt	.OFFSET		VT_UI4	Read/Write	0 to 4294967295
DB_ANY			VT_UI2	Read/Write	0 to 65535
DB_WWW			VT_UI2	Read/Write	0 to 65535
EVENT_ANY			VT_UI4	Read/Write	0 to 4294967295
EVENT_ATT			VT_UI4	Read/Write	0 to 4294967295
EVENT_HWINT			VT_UI4	Read/Write	0 to 4294967295
ErrorStruct	*<Structure>*		N/A		
Word	.ERROR_ID		VT_UI1	Read/Write	0 to 255
Byte	.FLAGS		VT_UI1	Read/Write	0 to 255
Byte	.REACTION		VT_UI1	Read/Write	0 to 255
Byte	.MODE		VT_UI2	Read/Write	0 to 255
UInt	.OPERAND_NUMBER		VT_UI2	Read/Write	1970 to 2262
UInt	.POINTER_NUMBER_LOCATION		VT_UI1	Read/Write	1970 to 2262

UInt	.SLOT_NUMBER_SCOPE		VT_UI2	Read/Write	1970 to 2262
CREF	*<Structure>*		N/A		
Byte	.CREF.BLOCK_TYPE		VT_UI4	Read/Write	0 to 4294967295
UInt	.CREF.CB_NUMBER		VT_UI1	Read/Write	1970 to 2262
UDInt	.CREF.OFFSET		VT_UI2	Read/Write	0 to 999999999
NREF	*<Structure>*		N/A		
Byte	.NREF.AREA		VT_UI1	Read/Write	0 to 255
UInt	.NREF.DB_NUMBER		VT_UI2	Read/Write	1970 to 2262
UDInt	.NREF.OFFSET		VT_UI4	Read/Write	0 to 4294967295
HW_ANY			VT_UI2	Read/Write	0 to 65535
HW_DEVICE			VT_UI2	Read/Write	0 to 65535
HW_DPMASTER			VT_UI2	Read/Write	0 to 65535
HW_DPSLAVE			VT_UI2	Read/Write	0 to 65535
HW_HSC			VT_UI2	Read/Write	0 to 65535
HW_IEPORT			VT_UI2	Read/Write	0 to 65535
HW_INTERFACE			VT_UI2	Read/Write	0 to 65535
HW_IO			VT_UI2	Read/Write	0 to 65535
HW_IOSYSTEM			VT_UI2	Read/Write	0 to 65535
HW_MODULE			VT_UI2	Read/Write	0 to 65535
HW_PTO			VT_UI2	Read/Write	0 to 65535
HW_PWM			VT_UI2	Read/Write	0 to 65535
HW_SUBMODULE			VT_UI2	Read/Write	0 to 65535
NREF	*<Structure>*		N/A		
Byte	.NREF.AREA		VT_UI1	Read/Write	0 to 255
UInt	.NREF.DB_NUMBER		VT_UI2	Read/Write	0 to 65535
UDInt	.NREF.OFFSET		VT_UI4	Read/Write	0 to 4294967295
OB_ANY			VT_I2	Read/Write	-32768 to 32767
OB_ATT			VT_I2	Read/Write	-32768 to 32767
OB_CYCLIC			VT_I2	Read/Write	-32768 to 32767
OB_DELAY			VT_I2	Read/Write	-32768 to 32767

OB_DIAG			VT_I2	Read/Write	-32768 to 32767
OB_HWINT			VT_I2	Read/Write	-32768 to 32767
OB_PCYCLE			VT_I2	Read/Write	-32768 to 32767
OB_STARTUP			VT_I2	Read/Write	-32768 to 32767
OB_TIMEERROR			VT_I2	Read/Write	-32768 to 32767
OB_TOD			VT_I2	Read/Write	-32768 to 32767
PIP			VT_UI2	Read/Write	0 to 65535
PORT			VT_UI2	Read/Write	0 to 65535
RTM			VT_UI2	Read/Write	0 to 65535
STRUCT	*<Structure>*		N/A		
The STRUCT data type does not have a pre-defined name or member names. It represents a data structure composed of different data types. Any data type can be included in STRUCT. See the Siemens programming guidelines for additional information.					

Tag References from Application Server and InTouch

The symbolic tag names supports space and period characters in item syntax. For the names of the data block, tag, or member with a space or a period, write it within the double quotes as shown in the below examples.

Data Block	**Tag**	**Member**	**Application Server Attribute I/O reference**
DB	Tag	Member	In User Defined Object (UDO): DB.Tag.Member In Object Viewer: SIDir.T01.DB.Tag.Member In InTouch: DB.Tag.Member
DB	Tag	Mem D	In UDO: attribute(DB.Tag."Mem D") In Object Viewer: SIDir.T01.attribute(DB.Tag."Mem D") In InTouch: DB.Tag."Mem D"
DB D2	Tag T2	Mem M2	In UDO: attribute("DB D2"."Tag T2"."Mem M2") In Object Viewer: SIDir.T01.attribute("DB D2"."Tag T2"."Mem M2") In InTouch: "DB D2"."Tag T2"."Mem M2"
DB.D2	Tag.T2	Mem.M2	In UDO: attribute("DB.D2"."Tag.T2"."Mem.M2") In Object Viewer: SIDir.T01.attribute("DB.D2"."Tag.T2"."Mem.M2") In InTouch: "DB.D2"."Tag.T2"."Mem.M2"
DB	T2[0]	Mem	In UDO: attribute(DB.T2[0].Mem)

			In Object Viewer:SIDir.T01.attribute(DB.T2[0].Mem) In InTouch: DB.T2[0].Mem
DB	T1 T2[2]	Mem	In UDO: attribute(DB."T1 T2"[2].Mem) In Object Viewer: SIDir.T01.attribute(DB."T1 T2"[2].Mem) In InTouch: DB."T1 T2"[2].Mem

Where:

- SIDir - represents a DDE/SL object configured to connect SIDirect Communication Driver
- T01 - is the topic name in the DDE/SL object

Conversions and Suffixes of Items (Symbolic Addressing)

- This section describes what data-format items and suffixes are converted and what they are converted into. The items and suffixes described here apply to the SIDirect Symbolic object (symbolic addressing). For items and suffixes that apply to the SIDirect Legacy object, see Conversions and Suffixes of Items (Absolute Addressing)

Suffix D (Symbolic)

The suffix D can be used with the DATE data type. When used without the suffix D, DATE saves the date as an unsigned integer, containing the year, month, and day. See General Data Types for additional information about DATE.

The suffix D (Date) converts DATE into a string to store the year, month, and day.

Usage: **DATE D**

For example: `MyDate D`

- The suffix D converts DATE to an 8-byte string (VT_BSTR) that stores the date. Range is 1990-01-01 to 2168-12-31. The construction is the number of days since 1/1/1990.
- When used without the suffix D, the SIDirect Communication Driver reads DATE from the PLC as an unsigned integer (VT_UI2). Range is 0 to 65535 (days).

Suffix LDT (Symbolic)

The suffix LDT can be used with the LDT data type. The LDT data type, when used without the suffix LDT, saves the date as an unsigned integer. See General Data Types for additional information about LDT.

The suffix LDT converts the LDT data type into a string.

Usage: **LDT LDT**

For example: `MyLDT LDT`

- The suffix LDT converts the LDT data type into a 12-byte string (VT_BSTR) that stores the date and time in string format. Range is 1970-01-01-0:0:0.000000000 to 2263-04-11-23:47:16.854775808. The construction is the number of nanoseconds since 1/1/1970 0:0.
- When used without the suffix LDT, the SIDirect Communication Driver reads data type LDT from the PLC as an unsigned integer (VT_UI8). Range is 0 to 18446744073709551615 (nanoseconds)

Suffix LT (Symbolic)

The suffix LT can be used with the LTime data type. When used without the suffix, LTime saves time as an integer, containing number of days (d), hours (h), minutes (m), seconds (s), milliseconds (ms), microseconds, (us), and nanoseconds (ns). See General Data Types for additional information about LTime.

The suffix LT converts the LTime data type into a 64-bit string defining the time.

Usage: **LTime LT**

For example: `MyLTime LT`

- The suffix LT converts LTime into a 64-bit string (VT_BSTR) that stores the time in string format. Range is -106751d23h47m16s854ms775us808ns to 106751d23h47m16s854ms775us807ns
- When used without the suffix LT, the SIDirect Communication Driver reads LTime from the PLC as an integer (VT_I8). Range is -9223372036854775808 to +9223372036854775807

Suffix S5T (Symbolic)

The suffix S5T can be used with the S5Time data type. When used without the suffix S5T, the SIDirect Communication Driver reads the 2-byte time from the S7-1500 PLC. See General Data Types for additional information about S5Time.

The suffix S5T converts STime into a string.

Usage: **S5Time S5T**

For example: `MyS5Time S5T`

- The suffix S5T converts S5Time into string (VT_BSTR) that stores the time as a value in the range of 0ms to 2h46m30s.
- When used without the suffix S5T, the SIDirect Communication Driver reads S5Time from the PLC as an unsigned integer (VT_UI2) with a range of 0 to 65535 (milliseconds).

Suffix T (Symbolic)

The suffix T can be used with the Time data type. When used without the suffix S5T, the SIDirect Communication Driver reads the 4-byte time from the S7-1500 PLC. See General Data Types for additional information about Time.

The suffix T converts Time into a string.

Usage: **Time T**

For example: `MyTime T`

- The suffix T converts Time into a string (VT_BSTR) that stores the time as a value in the range of -24d20h31m23s648ms to +24d20h31m23s647ms.
- When used without the suffix T, the SIDirect Communication Driver reads Time from the PLC as an integer (VT_I4) with a range of -2147483648 to +2147483647 (milliseconds).

Suffix TOD (Symbolic)

The suffix TOD can be used with the Time_Of_Day data type. When used without the suffix TOD, the SIDirect Communication Driver reads the 4-byte time from the S7-1500 PLC. See General Data Types for additional information about Time_Of_Day.

The suffix TOD converts Time_Of_Day to a string.

Usage: **Time_Of_Day TOD**

For example: `MyTime_Of_Day TOD`

- The suffix TOD converts Time_Of_Day to a string (VT_BSTR) that stores the time as a value in the range of 0:00:00.000 to 23:59:59.999.
- When used without the suffix TOD, the SIDirect Communication Driver reads Time_Of_Day from the PLC as an unsigned 4-byte integer (VT_UI2) with a range of 0 to 4294967295 (milliseconds).

Data Conversion

The following table describes how the SIDirect Communication Driver handles values that cannot be converted or do not meet the limit specifications.

Conversion	Description
NONSPECIFIC	If a value cannot be converted, the quality of the item goes to NONSPECIFIC.
Uncertain-HIGHLIMITED	If a value is greater than the upper limit, the quality of the item goes to uncertain-HIGHLIMITED.
Uncertain-LOWLIMITED	If a value is minor than the lower limit, the quality of the item goes to uncertain-LOWLIMITED.

Quality Settings

The SIDirect Communication Driver uses the general OPC-defined quality settings. An item can have six basic data quality states.

Quality Code	Quality State	Description
00C0	Data quality good	Data communications is good and data is good.
		The register is read or written to without any problems converting the data.
0055	Clamp low	Data communications is good but the data is uncertain.
		The data is clamped at a low limit.

		The register is correctly read or written to, but it is necessary to clamp its value to a limit.
		The value is smaller than the minimum allowed.
0056	Clamp high	Data communications is good but the data is uncertain.
		The data is clamped at a high limit.
		The register is correctly read or written to, but it is necessary to clamp its value to a limit.
		The value is larger than the maximum allowed.
		A string is truncated.
		For example, a floating point value is clamped to FLT_MAX.
0040	Quality uncertain/No convert	Data communications is good but the data is uncertain.
		The data cannot be converted.
		The server may return either a constant in place of the data or return quality information alone.
		The data is usable. However, it is not known whether the value is too large or too small.
		Incorrect data type.
		Floating point is not a number.
		For example, 0x000a in a PLC BCD register.
0004	Bad configure/No access	This is a configuration error.
		Data communications is good but the data cannot be sent and/or received. The data is bad and cannot be used.
		Item cannot be accessed.
		The item does not exist or is not available.
		The server can communicate with the PLC but cannot access the register.
		The server determined the point is not valid.
		The PLC responds that the register does not exist, cannot be read, or cannot be written to.
		The server cannot access a fenced, write-protected, or read-only item.

		The PLC is in a mode that does not permit access to this item.
		The number of data bytes is incorrect but the message is otherwise good.
		The command or op code is invalid but the message is otherwise good.
		The PLC is busy. The server has given up retrying.
0018	No communications	Data communications is down.
		Cannot access the PLC due to a communications error.
		Data is bad and cannot be used.
		The device group is in a slow poll or equivalent mode.
		The PLC does not exist and/or is not responding.
		There is no link validating the message.
		There is a lack of resources in the server. A TSR or driver cannot allocate memory.
		There is a lack of resources in the communications link.
		The communications link is off-line.
		All communications channels are in use.
		The network cannot route the message to the PLC.

Item Validation

When items are added, they are not initially validated. Since SIDirect symbolic addressing is in text format, the SIDirect Communication Driver can only determine whether an item is valid by querying the PLC to check whether the item is defined, and then returning the data type. Validation begins after items are added.

If you are adding many items, performance would be negatively impacted if the query was performed immediately. This delayed validation improves performance.

The SIDirect Communication Driver contains a tag database in its address space that closely mirrors the tag database in the S7 1200 or S7-1500 PLC. If the tag database in the PLC is not available, adding an item always succeeds and the data type VT_EMPTY is sent to the PLC. Once the SIDirect Communication Driver successfully queries the PLC, the data type is updated to the correct type.

Chapter 6

Troubleshooting the SIDirect Communication Driver

Troubleshooting Tools

The OI Server Manager provides access to diagnostics and other statistical data. The Log Viewer provides access to event messages logged during the operation of the SIDirect Communication Driver. Your client, for example, the InTouch software, can also monitor connectivity with the PLC through the SYSStatus item. Use these tools together with the information in this chapter to troubleshoot your SIDirect Communication Driver.

Finding the SIDirect Communication Driver Version Number

This section describes how to find the version number of your Communication Driver.

To find the version number

1. From the **Start** menu, point to **Settings**, and click on the **Control Panel** option.
2. Click **Programs and Features.**
3. Find SIDirect Communication Driver in the programs list and select it. The version is displayed. In some versions of Windows, you may have to click a hyperlink, **Click here for support information**. The release version of the Communication Driver appears in the **Support Info** dialog box.

OR

- Click on the **OI.SIDIR.1** Server node in the hierarchy-tree view. In the **Details** pane on the right you see the build version numbers of the respective SIDirect Communication Driver components.

OR

1. Search for **SIDirect.dll**.
2. Right-click on the **File Name** and select **Properties** on the menu.
3. Click the **Details** tab on the **Properties** dialog box. The version of your Communication Driver is listed under **File Version**.

Monitoring Connectivity Status with the PLC

You can use the built-in discrete item, SYSStatus, to monitor the status of communications with the PLC. This item is set to:

- 0 (zero) when communications with the PLC fails.
- 1 (one) when communications is successful.

For DDE/SuiteLink clients, SYSStatus always comes from the leaf level of a Communication Driver hierarchy branch, which is the destination PLC node. For OPC clients, SYSStatus can be accessed at all hierarchy levels. SYSStatus at the root level of the whole hierarchy tree is always good, as it represents the quality status of the local computer itself. For practical application, OPC clients should reference SYSStatus at any hierarchy levels other than the root.

Enter the following DDE reference formula in the appropriate place in your client:

```
=OISIDirect|S7PLC!$SYS$Status
```

where,

- **OISIDirect** is the name of the Communication Driver application.
- **S7PLC** is the exact device group defined in the DAServer for the PLC.
- **SYSStatus** is the discrete item that monitors the status of connectivity with the PLC.

Enter the following OPC item reference syntax when adding the item in your OPC client:

```
YourOPCAccessPath.$SYS$Status
```

where,

- **YourOPCAccessPath** is the assembly of hierarchy node names leading to a specific device (controller).
- **SYSStatus** is the discrete item used to monitor the status of connectivity with the device (controller).

Monitoring the status of Communication Driver Conversations

The **InTouch WindowsViewer** supports built-in topic names, **DDEStatus** and **IOStatus**, that can monitor the status of specific Communication Driver conversations.

For example, assume that **WindowViewer (VIEW)** is communicating with the SIDirect Communication Driver to a PLC. The PLC is defined in the InTouch application with the access name **S7PLC**. The discrete items, **DDEStatus** and **IOStatus**, are set to 0 when this Communication Driver conversation failed, and to 1 when this Communication Driver conversation is successful.

Using DDEStatus and IOStatus in Excel

The status of communications between the PLC and InTouch software can be read into Excel by entering the following DDE reference formula in a cell on a spreadsheet:

```
=view|DDEStatus!S7PLC
```

or

```
=view|IOStatus!S7PLC
```

where,

- **view** is the name of the InTouch application.
- **[DDE][IO] Status** is the built-in topic name that monitors the status of communications between the Communication Driver and the InTouch software.
- **S7PLC** is the exact access name defined in the InTouch application for the PLC.

Reading Values from the Communication Driver into Excel

Values can be read directly into Excel spreadsheets from the Communication Driver by entering a DDE formula into a cell using the following format:

```
=applicationname|<devicegroup>!itemname
```

Example formula:

```
=SIDirect|S7PLC!'DB1,B20'
```

where,

- **SIDirect** is the name of the Communication Driver application.
- **S7PLC** is the exact device group name defined in the Communication Driver for the PLC.
- **DB1,B20** is the actual location in the PLC that contains the data value. This is the item name.
- In this example, each time the value of **<DB1,B20>** changes in the PLC, the Communication Driver automatically sends the new value to the cell containing the formula in Excel.

Note: See the Microsoft Excel documentation for complete details on entering Remote Reference formulas for cells.

Writing Values to the Communication Driver from Excel

You can write values from Microsoft Excel to the Communication Driver by creating an Excel macro that uses the POKE command.

The proper command is entered in Excel as follows:

```
channel=INITIATE("applicationname","topicname")
=POKE(channel,"itemname", Data_Reference)
=TERMINATE (channel)
=RETURN()
```

The following describes each of the above POKE macro statements:

channel=INITIATE("applicationname","topicname")

- Opens a channel to a specific topic name that is defined in the Communication Driver in a particular application name (the executable name without the .exe).
- Assigns the number of that opened channel to channel.

Note: When using the channel=INITIATE statement, the word channel must be used in the =POKE statement instead of the actual cell reference. The "application name" and "topic name" portions of the formula must be enclosed in quotation marks.

=POKE(channel,"itemname", Data_Reference)

- Pokes the value contained in the Data_Reference to the specified item name or the actual location in the PLC, via the channel number that is returned by the previously executed INITIATE function.
- Data_Reference is the row/column ID of the cell containing the data value.

=TERMINATE(channel)

- Closes the channel at the end of the macro.
- Channel is the channel number returned by the previously executed INITIATE function.
- Some applications have a limited number of channels, therefore they should be closed when finished

=RETURN()

Marks the end of the macro.

Note: See the **.xlm** sample Excel poke macro provided on the install CD. See the Microsoft Excel documentation for complete details on entering Remote Reference formulas for cells.

Debugging Communications Between the SIDirect Communication Driver and the PLC

The OI Server Manager allows you use on-line diagnostics of the SIDirect Communication Driver components at run-time, locally and remotely.

To perform on-line diagnostics

- Select any active SIDirect Communication Driver on any node in the OI Server Manager.

  The Diagnostics branch is visible only if the Communication Driver is active as indicated by the green icon on the server branch. It contains the following sub-branches:

  - Client Groups
  - Structure
  - Transactions
  - Statistics
  - Messages
  - Device Groups

    Each of these sub-branches contains live information from the Communication Driver. They allow detailed diagnostics of objects within the SIDirect Communication Driver.

Note: If you have pokes that are folded, the diagnostics shows ALL items in the transaction. However, because they are folded, only items that have actually been sent have both the message ID and value. All other items that have not been sent, because of the folding, are listed in this transaction with the same timestamp but without the message and value.

For more information on Diagnostics, see "Using the Diagnostics Node" section in the Communication Drivers Pack help.

Diagnostics and Error Tracing

The SIDirect Communication Driver uses the standard diagnostic information provided by the Toolkit. Access to other internal diagnostic registers of the PLC is performed through reads and writes via the syntax used in Item Naming.

SIDirect Driver Diagnostic Info Items

The SIDirect Communication Driver provides Diagnostic System items specific to the Symbolic object. Diagnostic Info Items are the group of items that provide statistics information to reflect the performance of the driver.

Diagnostic Info Item names start with “SYS” to differentiate them from PLC tags.

Item Name	Data Type	Item Description	Access	Range/ Format
SYSItemUpdateRate	VT_UI4	Returns the instantaneous updates per second for subscribed items. To reset the value, set system item SYSResetStatistics to 1.	Read only	Range: 0 to 4294967295
SYSLoad	VT_R4	Returns the load of the connection to the PLC. The load is the rate of item updates over the last 15 minutes, and is expressed as a multiplier of the Load Factor. To reset the value, set system item SYSResetStatistics to 1. A load factor of 1 indicates that the rate of item updates is 1000 values/second; a load factor of 2 indicates that the rate of item updates is 2000 values/second.	Read only	

SYSTagsResolved	VT_UI4	Returns the number of unique tags that have been sent successfully to the PLC for update.	Read only	Range: 0 to 4294967295
SYSReadCount	VT_UI4	Returns the cumulative number of values that have been received successfully from the PLC. To reset the value, set system item SYSResetStatistics to 1.	Read only	Range: 0 to 4294967295
SYSWriteCount	VT_UI4	Returns the cumulative number of values that have been written successfully to the PLC. To reset the value, set system item SYSResetStatistics to 1.	Read only	Range: 0 to 4294967295
SYSReadCountFailed	VT_UI4	Returns the cumulative number of failed read requests sent from the server to the PLC. To reset the value, set system item SYSResetStatistics to 1.	Read only	Range: 0 to 4294967295
SYSWriteCountFailed	VT_UI4	Returns the cumulative number of failed write requests sent from the server to the PLC. To reset the value, set system item SYSResetStatistics to 1.	Read only	Range: 0 to 4294967295

SYSLastReadDuration	VT_UI4	Returns the elapsed time in milliseconds for the most recent read request to the PLC. Quality is set to 0x08 if a read has not occurred since the server started up.	Read only	Range: 0 to 4294967295
SYSLastWriteDuration	VT_UI4	Returns the elapsed time in milliseconds for the most recent write request to the PLC. Quality is set to 0x08 if a write has not occurred since the server started up.	Read only	Range: 0 to 4294967295
SYSLastReadCount	VT_UI4	Returns the total number of tags that were involved in the most recent read request to the PLC.	Read only	Range: 0 to 4294967295
SYSLastWriteCount	VT_UI4	Returns the total number of tags that were involved in the most recent write request to the PLC.	Read only	Range: 0 to 4294967295
SYSHostName	VT_BSTR	Returns the configured host name or IP address of PLC.	Read only	
SYSLastReadTime	VT_DATE	Returns the system time of the most recent read request to the PLC. Quality is set to 0x08 if a read has not occurred since the server started up.	Read only	System time is returned as UTC

SYSLastWriteTime	VT_DATE	Returns the system time of the most recent write request to the PLC. Quality is set to 0x08 if a write has not occurred since the server started up.	Read only	System time is returned as UTC
SYSStartTime	VT_DATE	Returns the startup time of the server.	Read only	System time is returned as UTC
SYSConnStatus	VT_UI4	Returns the current known connection status to the PLC.	Read only	0 = Disconnected 1 = Connecting 2 = Connected
SYSLastConnTime	VT_DATE	Returns the system time of the most recent connection attempt to the PLC.	Read only	System time is returned as UTC
SYSLastDisconnTime	VT_DATE	Returns the system time of the most recent disconnection attempt to the PLC. Quality is set to 0x08 if a disconnection has not occurred since the server started up.	Read only	System time is returned as UTC
SYSPLCType	VT_BSTR	Returns the PLC model number.	Read only	
SYSPLCFirmware	VT_BSTR	Returns the PLC firmware version.	Read only	
SYSResetStatistics	VT_BOOL	To reset system item statistics, set this item to 1. The server returns the value of this item to 0 when the reset operation finishes.	Read/Write	1 = True (reset statistics) 0 = False

SYSSyncDb	VT_BOOL	To force an upload of the tag database from the PLC to server, set this item to 1. The tag database will be uploaded, even if it is up to date. The server automatically returns the value of this item to 0 when the upload operation finishes.	Read/Write	1 = True (force upload) 0 = False The server automatically syncs up the tag database from the PLC. This system item can be used to ensure that the tag database in the server is synchronized to that in the PLC.
SYSPLCAccessLevel	VT_UI4	Returns the CPU Protection Level configured in the PLC. Refer to the Siemens TIA documentation for descriptions of each access level.	Read only	0 = Full Access including Fail Safe Access 1 = Full Access 2 = Read Access 3 = HMI Access 4 = No Access
SYSDiagnosticLogging	VT_UI4	Enable logger flags at protocol level. Example: All flags can be enabled by writing 15(0x0F) to this system item. The logger output folder is the same aacfg file. The file name is the same name as the PLC hierarchy node name. For more information see Using the OI SIDirect Diagnostic Log	Read/Write	0x01 = Info 0x02 = Warning 0x04 = Error 0x08 = Trace

Using the SIDirect Communication Driver Diagnostic Log

The SIDitect Communication Driver provides additional logging capability. This capability can be turned on using a system tag named sysdiagnosticlogging.

Diagnostic logging records the previous three hours. The log lines are output to a file on disk in the SIDirect Communication Driver data folder:

C:\ProgramData\Wonderware\OI-Server\$Operations Integration Supervisory Servers$\OI.SIDirect\LogFiles

Note: The SIDirect Communication Driver provides Diagnostic logs specific to the Symbolic objects only.

Diagnostics Facility

The SIDirect Communication Driver provides diagnostics in the following areas:

- Communications Processor Diagnostics
- S7 Communications Processor Diagnostics
- Items Diagnostics
- Messages Diagnostics
- Device Group Diagnostics

Communications Processor Diagnostics

The diagnostics window of the CP (Communications Processor) shows all Communication Driver-provided diagnostic columns plus a column for any error code on the CP level. The diagnostic information shows Communication Driver information text plus the error text description.

S7 Communications Processor Diagnostics

The diagnostics window of the CP shows all Communication Driver-provided diagnostic columns plus a column for any error code on the PLC CP level. The diagnostic information shows the Communication Driver information text plus the error code description.

Items Diagnostics

To the IO Server-provided diagnostics of items, the following two columns are added:

- The message ID
  The message ID consists of two parts:
  - A letter indicating the type of message.
    For example, P-poll, C-cyclic, B-block, and so on.
  - A message number.
    For example, order ID, block ID, event ID, and so on.
- The S7 error code
  The diagnostic information shows the message ID. For example, order ID, block ID, or event ID, and the error code description.

The following table shows different error codes and the corresponding description:

Letter Code	Letter Description	Error Code	Error Description
R	Result Error	255	OK
		1	Data hardware error
		3	Object access denied
		5	Data address is invalid
		6	Data type not supported

		7	Data type is not consistent
		10	Data does not exist in the PLC
L	Lib Error	120	Connection aborted
		123	Connection cannot be established
		23	Connection name not found
		25	No ldb or xdb file
		90	Firmware error
		140	An error occurred during the installation of the SINEC driver
		141	Internal error
		121	Request in this connection state not executable
		20	Invalid cref
		130	Invalid cyclic read state
		80	Data size too small
		30	No order for this order ID
		7	Invalid parameter in structure
		122	Maximum of available requests exceeded
		100	Type for Mini-DB call not allowed
		101	Value for Mini-DB call not allowed
		143	No license found
		3	Resources exhausted
		142	SINEC windows message server cannot be started
		53	Object access denied
		51	OD or attributes of object are inconsistent
		50	Object undefined
		31	Order ID already used
		81	Message received, but corresponding function not called
		11	Service not supported
		112	VFD already used by another application

		113	Connection already used by another application
		150	Incorrect symbolic address
		151	Size of symbolic address and size of buffer inconsistent
		8	PDU size violation
		9	Too many dll users
		2	Incorrect CP descriptor
		10	Wrong function for received message called
		40	Invalid r_id on this cref
		41	r_id already used
		42	No r_id found
		82	Data range or data type wrong
		83	Invalid segment
		160	Error in remote block receive
		161	Error in remote block send
		162	Data transfer cancelled by remote BSEND block
		163	Remote block database too small
		170	No receive block
B	Block Error	1	Timeout for initial values
		2	Timeout for update values
		3	Item range exceeds block range
E	Event Error	3	Item range exceeds add_value range
G	General Error	1	PO/PA item not readable
F	Fatal Error	2	Item doesn't fit in a single message and won't be advised
T	Timeout Error	3	Timeout for initial values
N	Connection Error	4	Received S7_ABORT_IND from the SAPI

Messages Diagnostics

Four types of messages with different diagnostics appear.

Variable Service Message with:

- P – poll message
- W – poke message
- C – cyclic message
- B – block message

In addition to the standard Communication Driver diagnostic messages, the following information is added:

- Message ID
- Two PDU sizes indicating block size if applicable and request block size
- The message or S7 error code

The diagnostic information shows the Communication Driver information text plus the message ID, PDU size, and error text description.

Device Group Diagnostics

The device group diagnostics window shows the Communication Driver-provided standard diagnostic columns plus the following four additional columns:

- Number of poll messages
- Number of cyclic services
- Number of block messages
- Number of alarm & event messages

The diagnostic information shows the corresponding additional values.

Communication Driver Diagnostic Messages

The SIDirect Communication Driver generates messages that you can use for diagnostic purposes. These DASTrace diagnostic messages do not necessarily indicate that error conditions exist.

The following table is not a complete list of messages that are shown in the Logger when DASTrace is enabled. Other diagnostic messages are shown, as well.

Note: The logger messages use the following codes: %s to represent strings, %d to represent numbers (integer), %x to represent the address of the object, and %ums to represent time in milliseconds.

Logger Message	Explanation	Probable Cause	Solution

Message is in SlowPollMode (%s msg=0x%08X) for topic %s	The message state is set to Slow poll interval.	The response for this message has not been received by the server. Either the response time is very small or the PLC is very busy.	Adjust the message reply timeout in the server in accordance with the performance of the PLC to achieve the optimal behavior.
Message leaves the SlowPollMode (%s msg=0x%08X) for topic %s	The message state is coming out of the Slow poll interval.	The server recovered from the error situation.	N/A
UpdateInterval for Device Group: %s changed to %d	The update interval for the device group has been changed. The messages are now to be polled at a new update interval for this device group.	N/A	N/A
Setting all items to bad in hierarchy: %s	The server updates the status of all the items as Bad on this connection.	The connection with the PLC is broken.	Check the connection with the PLC.
Connection with '%s' dropped, restarting connection	The socket connection with the PLC fails due to a given error code: "The PLC initiated termination of the Socket Connection."	The PLC is not responding to the Connect request from the Communication Driver.	The SIDirect Communication Driver will attempt to automatically reconnect. No user action is required.
Connection with '%s' dropped while pending, going to slow poll	The socket connection with the PLC fails due to a given error code: "The PLC initiated termination of the Socket Connection."	The PLC has been disconnected and/or has faulted.	Check the PLC connection and clear any outstanding PLC faults.
Block receive size: doesn't match msg size	The server receives an invalid block packet from the PLC.	This message from the PLC is garbled.	Check the PLC health status and the network condition.

Error Tracing with the Logger

The SIDirect Communication Driver supports error messages, board-specific error messages, trace logger messages, and error codes. Use the Log Flag data to customize the type of messages logged to the Log Viewer.

Note: See the Log Viewer online documentation for more information about using log flags.

SIDirect Communication Driver Logger Flags

The following logger flags are specific to the SIDirect Communication Driver.

- Errors
  General errors from the server have the prefix "ERROR." All related errors, such as for poll messages, have the prefix "POLL_ERROR." They include CONNECTION_ERROR, POLL_ERROR, POKE_ERROR, CYCLIC_ERROR, BLOCK_ERROR, and ALARMS_AND_EVENTS_ERROR.
- Trace
  General traces from the server have the prefix "TRACE." All related traces, such as for poll messages, have the prefix "POLL_TRACE." They include CONNECTION_TRACE, POLL_TRACE, POKE_TRACE, CYCLIC_TRACE, BLOCK_TRACE, and ALARMS_AND_EVENTS_TRACE.

The following table lists all the available logger flags and their meanings.

Type	Name	Description
Server Flag	ERROR	Shows general server errors.
	TRACE	Shows general server traces.
Transaction Flag	CONNECTION_ ERROR	Shows connection errors.
	POLL_ERROR	Shows errors of poll messages.
	POKE_ERROR	Shows errors of poke messages.
	CYCLIC_ERROR	Shows errors of cyclic messages.
	BLOCK_ERROR	Shows errors of block messages.
	ALARMS_AND_ EVENTS_ERROR	Shows errors of alarm and scan messages.
	CONNECTION_ TRACE	Shows connection traces.
	POLL_TRACE	Shows traces of poll messages.
	POKE_TRACE	Shows traces of poke messages.
	CYCLIC_TRACE	Shows traces of cyclic messages.
	BLOCK_TRACE	Shows traces of block messages.
	ALARMS_AND_ EVENTS_TRACE	Shows traces of alarm and scan messages.

Error Messages, Trace Messages, Error Codes, and Warnings

In addition to the SIDirect Communication Driver error and warning messages, S7 Trace messages and generic Communication Driver error codes are supported. Use these messages together with the OI Server Manager Diagnostic root data to troubleshoot SIDirect Communication Driver problems.

Note: The logger messages use the following codes: %s to represent strings, %d to represent numbers (integer), %x to represent the address of the object, and %ums to represent time in milliseconds.

You can also use the Log Flag data to customize the type of messages logged to the Log Viewer. For more information about using log flags, see the Log Viewer online Help.

Communication Driver Error Messages

The following list shows error messages produced by the Communication Driver that are logged to the Log Viewer with the DASProtFail log flags.

Logger Message	Explanation	Probable Cause	Solution
Internal Error: CIoVariant::Update()	The internal type conversion encounters an invalid or unknown type.	This is an internal program error.	Turn on POLL_TRACE in the Logger to obtain additional trace information. Report the error to Wonderware Technical Support.
Send fail because of wrong order id	The message cannot be sent to the PLC because of the order ID is incorrect.	This is an internal program error.	Turn on POLL_TRACE, POKE_TRACE, and CYCLIC_TRACE in the Logger to obtain additional trace information. If the problem persists, report it to Technical Support.
(%d):s7_multiple_read_req (orderid=%d) [(%d) %s]	The poll message cannot be sent to the PLC.	This is an internal program error.	Turn on POLL_TRACE in the Logger to obtain additional trace information. If the problem persists, report it to Technical Support.
Internal state error: multiple read response (S7Type 0x%X)	The server receives a duplicate response for the poll message.	This is an unknown error.	Turn on POLL_TRACE in the Logger to obtain additional trace information. If the problem persists, check with the PLC vendor.

(%d):Invalid item name: %s (%s)	The requested item name has a bad syntax.	The item syntax is wrong.	Correct the item syntax as defined in this user's guide.
(%d):Invalid item name suffix: %s (%s)	The requested item suffix has a bad syntax.	The item suffix is wrong.	Correct the item suffix as defined in this user's guide.
Alarm event header key not: FF09: %02X%02X	The alarm received from the PLC has a bad header.	This is a PLC issue.	Turn on ALARMS_AND_EVENTS_TRACE in the Logger to obtain additional trace information. Report the error to Wonderware Technical Support.
Send: (MSG_FAIL): message (%s msg=0x%08X) [msg_state=%d,con_state=%d]	The server fails to send the message to the PLC.	This is an internal error.	Turn on POLL_TRACE, POKE_TRACE, and CYCLIC_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
(%d): s7_brcv_init (r_id=%d)[(%d) %s]	There is an error in s7_brcv_init (Block services).	A configuration or communications error occurred.	Turn on BLOCK_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
Timeout for initial values of block with r_id=%d	The block message has timed out for the initial updates.	The PLC does not send the block update.	Check the PLC program and see the B_SEND is configured correctly.
Timeout updating values of block with r_id=%d	Timeout occurs while updating the subsequent values for the block message.	The connection with the PLC may be dropped.	Check the PLC program and see the B_SEND is configured correctly.
Release blockid:%d for message (%s msg=0x%08X,con=%s) [number of blockids=%d] was not successful	Releasing the block ID from the block message operation fails.	This is an internal error.	Turn on BLOCK_TRACE in the Logger to obtain additional trace information. If the error persists, report it to Technical Support.

S7BlockMessage:: HandleResponse: premature block end received at size %d (expected: %d)	The message received from the PLC for the block service has no data.	This is an unknown error.	Turn on BLOCK_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
S7BlockMessage:: HandleResponse: unsegmented block messages did not have correct size: %d (len: %d)	The unsegmented message received from the PLC for the block service has incorrect data size.	This is an unknown error.	Turn on BLOCK_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
S7BlockMessage:: HandleResponse: segmented messages did not add up correctly in size: %d to indicated response data (len: %d)	The segmented message received from the PLC for the block service has incorrect data size.	This is an unknown error.	Turn on BLOCK_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
S7BlockMessage:: HandleResponse: cannot allocate memory for response data (len: %d)	The server cannot allocate the memory for the response received from the PLC for the block message.	The server runs out of memory.	Turn on BLOCK_TRACE in the Logger to obtain additional trace information. Restart the server or the computer.
S7BlockMessage:: HandleResponse: too much block data received: (received: %d + new: %d more than expected: %d)	The server receives too much data for the block message.	This is a PLC problem.	Turn on BLOCK_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
Internal Error: Unknown type: %d updating event item: %s	The server receives the updates for the unknown type of alarm and event message.	This is an unknown error.	Turn on ALARMS_AND_EVENTS_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
Could not generate data for item %s	The server cannot read the poke value for the item.	This is an internal error.	Turn on POKE_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.

Can't send request for msg=0x%08X (no orderid available)	The server cannot send the message to the PLC because it runs out of order IDs.	This is an unknown error.	Turn on POLL_TRACE, POKE_TRACE, and CYCLIC_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
Send fail because of wrong order id	The server cannot send the message to the PLC because the order ID is incorrect.	This is an unknown error.	Turn on POLL_TRACE, POKE_TRACE, and CYCLIC_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
(%d): s7_multiple_write_req (orderid=%d) [(%d) %s]	The write request to the PLC fails.	The data poke may be too long or this is an internal error.	Turn on POKE_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
Internal state error: multiple write response (S7Type 0x%X)	The server receives a duplicate response for the poke message.	This is an unknown error.	Turn on POKE_TRACE in the Logger to obtain additional trace information. If the problem persists, report the error to Technical Support.
(%d): s7_cycl_read_init_req (orderid=%d) [(%d) %s]	An error occurs when sending the S7 cyclic read initiation request message.	A configuration or communications error occurred.	Check the PLC configuration/connection and resource limitation, particularly the ones related to the cyclic servce.
(%d): s7_cycl_read_start_req (orderid=%d): [(%d) %s]	An error occurs when sending the S7 cyclic read start request message.	A configuration or communications error occurred.	Check the PLC configuration or connection.
(%d): s7_cycl_read_delete_req (orderid=%d): [(%d) %s]	An error occurs when sending the S7 cyclic read delete message.	A configuration or communications error occurred.	Check the PLC configuration or connection.

ERROR in S7_CYLC_READ_INIT_CNF - PLC responded with error code	The PLC returns an error code in response to the cyclic read initiation request.	It is a PLC, Communication Driver configuration, or resource error.	Turn on CYCLIC_TRACE in the Logger to obtain additional trace information. Check the PLC configuration/connection and resource limitation, particularly the ones related to the cyclic service.
ERROR in S7_CYCL_READ_START_CNF - PLC responded with error code	The PLC returns an error code in response to the cyclic read start request.	It is a PLC or Communication Driver configuration error.	Turn on CYCLIC_TRACE in the Logger to obtain additional trace information. If the problem persists, contact Technical Support.
ERROR in S7_CYCL_READ_DELETE_CNF - PLC responded with error code	The PLC returns an error code in response to the cyclic read delete request.	It is a PLC or Communication Driver configuration error.	Turn on CYCLIC_TRACE in the Logger to obtain additional trace information. If the problem persists, contact Technical Support.
Internal state error: cyclic read response (S7Type %d -> 0x%X)	The server receives a duplicate response for the cyclic message.	This is an unknown error.	Turn on CYCLIC_TRACE in the Logger to obtain additional trace information. If the problem persists, contact Technical Support.
Scan event header key not: 120A: %02X%02X	When parsing the packet returned by the Scan event packet from the PLC, the header key is incorrect.	This is a programming error.	Turn on ALARMS_AND_EVENTS_TRACE in the Logger to obtain additional trace information. Repeat the test and/or restart the Communication Driver/PLC. If the error persists, contact the PLC vendor.
Item doesn't fit in a single message and won't be advised	The server cannot fit this item into a message, therefore it cannot advise this item.	The item byte range is larger than the PLC PDU size.	Split the item into smaller items so that they can be fitted into the available PDU size (480 bytes).
Can't create poke message for item %s, data size is too large	The server cannot fit this item into a poke message,	The item byte range is larger than the PLC PDU size.	Split the item into smaller items so that they can be fitted into the available PDU size (480 bytes).

	therefore it cannot poke this item.		
Can't create poke message for item %s, not able to generate data	The poke data for creating a poke message cannot be generated.	The poke data value cannot be converted into the PLC datatype.	Check the value being poked and create the correct format.
,%s, Leaving Slow Poll Mode	This is only a piece of information about the server leaving the Slow Poll mode.	The connectivity to the PLC recovered from a failure. Normal communications is resumed.	If this message shows up consistently, verify the network connectivity to the PLC.
,%s, Entering Slow Poll Mode	This is only a piece of information about the server entering the Slow Poll mode.	The connectivity to the PLC failed. The Communication Driver tries to reconnect at the Slow Poll interval.	Verify the network connectivity to the PLC. Turn on CONNECTION_TRACE, DASSend, and DASReceive in the Logger to obtain additional diagnostic information.
TIMEOUT for pending initiate request	A timeout occurs while waiting for an initiate confirmation.	A configuration or communications error occurred.	Check the communications or configuration. If the problem persists, turn on CONNECTION_TRACE, DASSend, and DASReceive in the Logger to obtain additional diagnostic information.
(%d):s7_initiate_req [(%d) %s]	There is an error in initiating a request (establishing a connection).	A configuration or communications error occurred.	Check the connection and the PLC configuration/ program. If the problem persists, turn on CONNECTION_TRACE, DASSend, and DASReceive in the Logger to obtain additional diagnostic information.

Can't connect	There is an error establishing a connection.	A configuration or communications error occurred.	Check the connection and the PLC configuration/ program. If the problem persists, turn on CONNECTION_TRACE, DASSend, and DASReceive in the Logger to obtain additional diagnostic information.
Received order id on a deleted non read/write (type: 0x%04X) message: 0x%x	The server receives a response from the PLC for a message that has already been deleted from the server.	Unexpected order ID is received from the PLC. The Communication Driver discards the message associated with it.	If the problem persists, contact the PLC vendor.
Generate data failed for item %s because at least one element of the array is not filled	The server cannot poke the array item as the items are not filled correctly.	You poke the array items but some of the element are left unfilled. All elements in the array are rejected for poking.	Fill all elements in the array before poking the array.
TIMEOUT for connection (while %s), m_state=%d	A timeout occurs while waiting for a response message to <command>.	A communications/ configuration error occurred.	Verify the network connectivity to the PLC. Turn on CONNECTION_TRACE, DASSend, and DASReceive in the Logger to obtain additional diagnostic information.
(%d): s7_msg_initiate_req (orderid=%d) for %s [(%d) %s]	There is an error in the message initiate request (initiating alarms and events).	A configuration or communications error occurred.	Verify the network connectivity to the PLC. Turn on CONNECTION_TRACE, DASSend, and DASReceive in the Logger to obtain additional diagnostic information.

ERROR: order ID %d exists for cyclic ID: %d in cyclic reference map	The server tries to add the order ID for the cyclic service that has already been occupied by some other message.	This is an internal error.	If the problem persists, turn on CYCLIC_TRACE in the Logger to obtain additional diagnostic information.
ERROR: Retrieving order ID: %d from cyclic ID: %d in cyclic reference map	The server cannot find the order ID in the order ID map for this message.	This is an internal error.	If the problem persists, turn on CYCLIC_TRACE in the Logger to obtain additional diagnostic information.
ERROR: UNKNOWN cyclic ID (%d) in cyclic read indication	The server receives an unknown cyclic message form the PLC.	The PLC reports a cyclic response that is not requested by the Communication Driver.	Check with the PLC vendor.
,%s, Connection aborted	The connection to the PLC is closed.	Either the PLC closed the connection or the server closed the connection.	If this is caused by the normal shutdown or items removal, no actions are required. If not, verify the PLC configuration for the Keep-Alive parameter.

S7 Trace Messages

The SIDirect Communication Driver provides five types of trace messages as follows:

- Connection Trace
- Poll Trace
- Cyclic Trace
- Block Trace
- Alarms and Events Trace

The following table lists the trace messages produced by the Communication Driver. For more information about trace messages, see SIDirect Communication Driver Logger Flags.

Logger Message	**Explanation**	**Probable Cause**	**Solution**
CONNECTION_TRACE Messages			

(%d):s7_initiate_req [0]	There is an error establishing a connection. The server cannot send the Connect Request to the PLC. The first parameter is the error code of the function call (-1 = Message Blocked, -2 = Message Failed) from the PLC. The parameters inside the square bracket is 0.	The credit is not available to send this request.	If after some time the problem still exists, restart the server.
Can't connect	There is an error establishing a connection. The server cannot send the Connect Request to the PLC.	The credit is not available to send this request.	If after some time the problem still exists, restart the server.
Close connection (con=%s): (cpd=%d,cref=%d)	INFO: The Communication Driver closes the connection with the PLC. The parameters associated with the PLC connection are listed.	N/A	N/A
Connection (con=%s) was not successful	The server connection attempts to the PLC fails.	The PLC is in a faulty condition or the connection is broken.	Check the PLC or check the cable connection.
Connection (con=%s) was successful	INFO: The server connects with the PLC successfully.	N/A	N/A
Internal Error: %s Set state of connection to %d	The connection with the PLC cannot be established.	This is an unknown error.	Check the PLC or cable connection, or restart the server.
Open connection (con=%s)	INFO: The socket connect call to the PLC is successful but the connection negotiation is still not completed	N/A	N/A
s7_abort = OK	INFO: The SIDirect Communication Driver closes the connection with the PLC successfully.	The connection is closed by either the server or the PLC	N/A
s7_get_initiate_cnf cnf amq called: %d	INFO: PLC agrees on this parameter with the Communication Driver. (cnf_amq_called)	N/A	N/A

s7_get_initiate_cnf cnf amq calling: %d	INFO: PLC agrees on this parameter with the Communication Driver. (cnf_amq_calling)	N/A	N/A
s7_get_initiate_cnf cnf pdu size: %d	INFO: PLC agrees on this parameter with the Communication Driver. (cnf_pdu_size). The PDU size is returned.	N/A	N/A
s7_get_initiate_cnf ind amq called: %d	INFO: This is the Communication Driver side negotiation parameter with the PLC. (ind_amq_called)	N/A	N/A
s7_get_initiate_cnf ind amq calling: %d	INFO: This is the Communication Driver side negotiation parameter with the PLC (ind_amq_calling).	N/A	N/A
s7_get_initiate_cnf ind pdu size: %d	INFO: This is the Communication Driver side negotiation parameter with the PLC. (ind_pdu_size) The PDU size of the PLC is requested.	N/A	N/A
s7_initiate_req = OK	The server sends the Connect Request to the PLC successfully.	N/A	N/A
%s Set state of connection to CLOSED	The SIDirect Communication Driver closes the connection to the PLC.	The connection is closed by the server or by the PLC.	If the cable is not disconnected, check the PLC configuration and see if you have specified the Keep-Alive parameter. The Keep-Alive parameter causes the closing of connection if there are no activities for some specified amount of time.
%s Set state of connection to ERROR	The connection with the PLC cannot be established.	The server cannot connect to the PLC at all.	Check the PLC or cable connection, or restart the server.

%s Set state of connection to OPEN	INFO: The server opens a connection with the PLC.	N/A	N/A
%s Set state of connection to PENDING	INFO: The server is awaiting the response from the PLC for the Connection Request.	N/A	N/A
%s Set state of connection to SLOW POLL MODE	The server is going into the Slow Poll mode.	The connection with the PLC is bad.	Check the PLC or check the cable connection.
POLL_TRACE Messages			
%s dumping s7 objects (containing %d items)	INFO: Dumping the information about the S7Info in the logger.	N/A	N/A
(1): s7_multiple_read_req (orderid=%d) [0]	INFO: The multiple read request message cannot be sent to the PLC because the server runs out of credit.	N/A	N/A
(2): s7_multiple_read_req (orderid=%d) [0]	The multiple read request message cannot be sent to the PLC because of unknown reasons.	N/A	If the problem persists, restart the server.
add pItem (%s) to S7Info (pS7Info=0x%08X, addr: %d range: %d)	INFO: The server adds item to the S7Info while building the Poll message.	N/A	N/A
Build message (%s msg=0x%08X) for topic %s	INFO: The server builds a POLL message for the topic.	N/A	N/A
Delete message (%s msg=0x%08X) for topic %s	INFO: Destructor for the POLL message gets called.	N/A	N/A
dumping read values	INFO: Dumping the read values received from the PLC in the logger.	N/A	N/A
Got orderid:%d for message (%s msg=0x%08X,con=%s) [number of orderids=%d]	INFO: The server assigns the order ID for the POLL message.	N/A	N/A

Release orderid:%d for message (%s msg=0x%08X,con=%s) [number of orderids=%d]	INFO: Release the order ID from the message.	N/A	N/A
remove pItem (%s) from S7Info (pInfo=0x%08X)	INFO: The server removes the item from the S7Info.	N/A	N/A
s7_multiple_read_req = OK (orderid=%d)	INFO: The multiple read request message is sent to the PLC successfully.	N/A	N/A
S7Info=%s:result=%d var_length=%d value (in HEX ASCII):	INFO: The server logs the information about the read value received from the PLC in the logger.	N/A	N/A
POKE_TRACE Messages			
%s dumping s7 objects (containing %d items)	INFO: Dumping the information about the S7Infos in the logger.	N/A	N/A
%s: result=%d var_length=%d value (in HEX ASCII)	INFO: Logging the Result received from the PLC for the write request.	N/A	N/A
(1): s7_multiple_write_req (orderid=%d) [(%d) %s]	INFO: The multiple write request message cannot be sent to the PLC because running out of credit.	N/A	N/A
(2): s7_multiple_write_req (orderid=%d) [(%d) %s]	The multiple write request message cannot be sent to the PLC because of unknown reasons.	N/A	If problem persists, restart the Server or check the PLC.
add pItem (%s) to S7Info (pS7Info=0x%08X, addr: %d range: %d)	INFO: Server adding item to the S7Info while building the Poke Message.	N/A	N/A
Build message (%s msg=0x%08X) for topic %s	INFO: Server building a Poke message for the topic.	N/A	N/A

Delete message (%s msg=0x%08X) for topic %s	INFO: Destructor for the Poke message gets called.	N/A	N/A
Got orderid:%d for message (%s msg=0x%08X,con=%s) [number of orderids=%d]	INFO: Server assigns the order id for the Poke Message.	N/A	N/A
Release orderid:%d for message (%s msg=0x%08X,con=%s) [number of orderids=%d]	INFO: Release the order ID from the poke message.	N/A	N/A
remove pItem (%s) from S7Info (pInfo=0x%08X)	INFO: Server removing the item from the S7Info.	N/A	N/A
s7_multiple_write_req = OK (orderid=%d)	INFO: The multiple write request message sent to the PLC successfully.	N/A	N/A
S7PokeMessage::AddPokeItem (Item=%s)	INFO: Server adding item to the Poke Message.	N/A	N/A
TREAL item %s high clamped to 9990s	INFO: The value of TREAL item is clamped at high value.	N/A	N/A
TREAL item %s low clamped to 0ms	INFO: The value of TREAL item is clamped at low value.	N/A	N/A
CYCLIC_TRACE Messages			
%s dumping s7 objects (containing %d items)	INFO: Dumping the information about the S7Infos in the logger.	N/A	N/A
(-1):s7_cycl_read (orderid=%d) [(%d) %s]	The server could not send the cyclic read request to the PLC. Temporary error.	N/A	N/A
(1): s7_cycl_read_delete_req (orderid=%d): [0]	The server could not send the cyclic read delete request to the PLC because of running out of credit.	N/A	N/A

(1): s7_cycl_read_start_req (orderid=%d): [0]	The server could not send the cyclic read start request to the PLC because of running out of credit.	N/A	N/A
(2): s7_cycl_read_delete_req (orderid=%d): [0]	The server could not send the cyclic read delete request to the PLC because of Unknown reason.	N/A	N/A
add pItem (%s) to S7Info (pS7Info=0x%08X, addr: %d range: %d)	INFO: Server adding item to the S7Info while building the cyclic Message.	N/A	N/A
Build message (%s msg=0x%08X) for topic %s	INFO: Server building a cyclic message for the topic.	N/A	N/A
Delete message (%s msg=0x%08X) for topic %s	INFO: Destructor for the cyclic message gets called.	N/A	N/A
Got orderid:%d for message (%s msg=0x%08X,con=%s) [number of orderids=%d]	INFO: Server assigns the order id for the cyclic Message.	N/A	N/A
Release orderid:%d for message (%s msg=0x%08X,con=%s) [number of orderids=%d]	INFO: Release the order ID from the cyclic message.	N/A	N/A
remove pItem (%s) from S7Info (pInfo=0x%08X)	INFO: Server removing the item from the S7Info.	N/A	N/A
s7_cycl_read = OK (orderid=%d)	INFO: The server successfully sends the cyclic read request to the PLC.	N/A	N/A
s7_cycl_read_delete_req = OK (orderid=%d)	INFO: The server successfully sends the cyclic read delete request to the PLC.	N/A	N/A

s7_cycl_read_start_req = OK (orderid=%d)	INFO: The server successfully sends the cyclic read start request to the PLC.	N/A	N/A
s7_get_cycl_read_init_cnf (cpd=%d,m_cref=%d,orderid=%d) Build Poll Messages: [(%d) %s]	INFO: The server gets the confirmation from the PLC for the Cyclic read request.	N/A	N/A
S7Info=%s:result=%d var_length=%d value (in HEX ASCII)	INFO: Logging the value received from the PLC for the cyclic message	N/A	N/A
BLOCK_TRACE Messages			
(0): s7_brcv_init (r_id=%d) [0]	INFO: The block receive initiation request is successful.	N/A	N/A
add item (%s) to S7BlockMessage (%s msg=0x%08X)	INFO: Adding items to the block message.	N/A	N/A
Build message (%s msg=0x%08X) for topic %s	INFO: Server building a Block message for the topic.	N/A	N/A
Delete message (%s msg=0x%08X) for topic %s	INFO: Destructor for the block message gets called.	N/A	N/A
Release blockid:%d for message (%s msg=0x%08X,con=%s) [number of blockids=%d]	INFO: Release block id for the block message.	N/A	N/A
remove item (%s) from S7BlockMessage (%s msg=0x%08X)	INFO: Remove item from the block message.	N/A	N/A
s7_brcv_init = OK (r_id=%d)	INFO: The block receive initiation request is successful.	N/A	N/A
s7_brcv_stop = OK (r_id=%d)	INFO: The block receive stop request is successful.	N/A	N/A
ALARMS_AND_EVENTS_TRACE Messages			

%s: ack_state=%d	INFO: Update the alarm or event with the acknowledgement state information.	N/A	N/A
%s: event_state=%d	INFO: Update the alarm or event with the event state information.	N/A	N/A
%s: no_add_value=%d	INFO: Update the alarm or event with the number of additional value information.	N/A	N/A
%s: state=%d	INFO: Update the alarm or event with the state information.	N/A	N/A
%s: time_stamp=%s	INFO: Update the alarm or event with the time stamp information.	N/A	N/A
(%d): s7_msg_abort_req (orderid=%d) for %s [0]	The alarm abort request to the PLC failed.	N/A	N/A
add item (%s) to S7Event (msg=0x%08X)	INFO: Add item to the event message.	N/A	N/A
Build alarm object (0x%08X) for connection %s	INFO: Server builds the alarm object that are going to receive the alarms and events.	N/A	N/A
Build scan object (0x%08X) for connection %s	INFO: Server builds the scan object.	N/A	N/A
Delete alarm object (0x%08X) for connection %s	INFO: The destructor of the alarm object gets called.	N/A	N/A
Release eventid:%d (con=%s) [number of eventids=%d]	INFO: Release order id from the event message.	N/A	N/A
remove item (%s) from S7Event (0x%08X)	INFO: Remove item from the event message.	N/A	N/A

s7_msg_abort_req = OK (orderid=%d) for %s	INFO: The server sends the alarm abort request to the PLC successfully.	N/A	N/A
s7_msg_initiate_req = OK (orderid=%d) for %s	INFO: The server sends the alarm registration request to the PLC successfully.	N/A	N/A

Communication Driver Error Codes

The following table lists the Communication Driver error codes and the error messages that appear with the codes, and their descriptions.

Code	Error Message	Description
C004D000L	Invalid item name	The requested item name has bad syntax.
C004D001L	Item name not exist	The requested item name has good syntax, but it does not exist.
C004D002L	Device not connect	The device is not connected, so data cannot be acquired.
C004D100L	Device off scan	The device is communicating, but it cannot accept queries for data items.
C004D101L	Timeout	A message transaction with the device timed out.

Communication Driver Protocol Warnings

The following table lists protocol warnings generated by the Communication Driver. The log flag for these messages is DASProtWarn.

Logger Message	Explanation	Probable Cause	Solution
Send: exit (MSG_OK): Attempt to send POLL message while pending (%s msg=0x%08X) [msg_state=%d,con_state=%d]	The server tries to send the Poll message while it waits for the response from the PLC for the same message.	The response from the PLC is slow.	This is a flow control issue. The server is too fast to send the message but the PLC is slow in responding to those messages. Try to reduce the load from the PLC by disconnecting other clients from the PLC or reducing the scan rate of the message. If the problem exists call the PLC vendor.

Update item (%s, quality=0x%04X) on %s	The server updates the item with Bad quality. This message shows up only when the item quality is Bad.	N/A	Check the OPC quality for the appropriate error message.
S7Info (%s pS7Info=0x%08X/ pMsg=0x%08X) returned error: %s	Logging the error code returned by the PLC for the read request.	There is an item access error in the PLC.	Check the PLC configuration and see whether the memory area you try to access exists in the PLC with proper access right.
Could not generate data for item %s	The server cannot read the poke value.	This is an internal error.	Turn on POKE_TRACE in the Logger to obtain additional trace information. Report the error to Technical Support.
Send: exit (MSG_OK): Attempt to send POKE message while pending (%s msg=0x%08X) [msg_state=%d,con_state=%d]	The server tries to send the poke message while it waits for the response from the PLC for the same message.	The response from the PLC is slow.	This is a flow control issue. The server is too fast to send the message but the PLC is slow in responding to those messages. Try to reduce the load from the PLC by disconnecting other clients from the PLC or reducing the scan rate of the message. If the problem exists call the PLC vendor.
S7 Topic's <%s> property <%s> was changed to <%s>	The server is not hot-configurable for the given property.	The server does not use the changed value.	Re-start the server to see this change in effect.
Invalid value, clamp at high limit for poking item: %s on %s	Poke data is clamped into a valid range before it is sent to the PLC.	Poke value exceeds the S7 data type range.	See the Communication Driver user's guide for the correct range of values.
Clamping S5T poke data for %s on %s (client poke %ums clamped to 9990000)	S5T poke value is clamped to 9990000 before it is sent to the PLC.	Poke value exceeds the valid range.	See the Communication Driver user's guide for the correct range of values.
Loosing precision on converting S5T poke data for %s on %s (client poke %ums converted to 0ms)	Non-zero S5T poke value is converted to 0.	Poke value is below the S7 S5T type resolution.	See the Communication Driver user's guide for the correct range of values.

Loosing precision on converting S5T poke data for %s on %s (client poke %ums converted to %ums)	Precision is lost on value.	The resolution of the S7 data type does not match the poked value.	See the Communication Driver user's guide for the correct range of values.
Invalid poke value, clamp at low limit for item: %s on %s	Poke data is clamped into a valid range before it is sent to the PLC.	Poke value exceeds the S7 data type range.	See the Communication Driver user's guide for the correct range of values.
Invalid poke value, cannot convert value for item: <item name> on <device group>	Poke data is set to constant.	Poke value exceeds the S7 data type range.	See the Communication Driver user's guide for the correct range of values.
ERROR: Invalid value, clamp at high limit for poking item: %s on %s	Poke data is clamped into a valid range.	Poke value exceeds the S7 data type range.	See the Communication Driver user's guide for the correct range of values.
ERROR: Invalid value, clamp at low limit for poking item: %s on %s	Poke data is clamped into a valid range.	Poke value exceeds the S7 data type range.	See the Communication Driver user's guide for the correct range of values.
ERROR: Invalid value, cannot convert for poking item: %s on %s	Poke data is set to constant.	Poke value exceeds the S7 data type range.	See the Communication Driver user's guide for the correct range of values.
Write complete fails - item: %s on %s	The server cannot write the value of the item to the PLC.	Connection to the PLC is bad or Item access is denied by the PLC.	Check the PLC connection or configuration.
S7Cp's <%s> property <%s> was changed to <%s>	The server is not hot-configurable for the given property.	The server does not use the changed value.	Restart the server to see this change in effect.
(%d):s7_msg_initiate_req (orderid=%d) for %s [(0)]	There is an error in the message initiate request (initiating alarms and events).	It is a communications error or a PLC configuration error.	Check the connection and the PLC configuration/program.

Data Conversion

The following table describes how the SIDirect Communication Driver handles values that cannot be converted or do not meet the limit specifications.

Conversion	**Description**

NONSPECIFIC	If a value cannot be converted, the quality of the item goes to NONSPECIFIC.
Uncertain-HIGHLIMITED	If a value is greater than the upper limit, the quality of the item goes to uncertain-HIGHLIMITED.
Uncertain-LOWLIMITED	If a value is minor than the lower limit, the quality of the item goes to uncertain-LOWLIMITED.

Quality Settings

The SIDirect Communication Driver uses the general OPC-defined quality settings. An item can have six basic data quality states.

Quality Code	**Quality State**	**Description**
00C0	Data quality good	Data communications is good and data is good.
		The register is read or written to without any problems converting the data.
0055	Clamp low	Data communications is good but the data is uncertain.
		The data is clamped at a low limit.
		The register is correctly read or written to, but it is necessary to clamp its value to a limit.
		The value is smaller than the minimum allowed.
0056	Clamp high	Data communications is good but the data is uncertain.
		The data is clamped at a high limit.
		The register is correctly read or written to, but it is necessary to clamp its value to a limit.
		The value is larger than the maximum allowed.
		A string is truncated.
		For example, a floating point value is clamped to FLT_MAX.
0040	Quality uncertain/No convert	Data communications is good but the data is uncertain.
		The data cannot be converted.
		The server may return either a constant in place of the data or return quality information alone.

Quality Code	Quality State	Description
		The data is usable. However, it is not known whether the value is too large or too small.
		Incorrect data type.
		Floating point is not a number.
		For example, 0x000a in a PLC BCD register.
0004	Bad configure/No access	This is a configuration error.
		Data communications is good but the data cannot be sent and/or received. The data is bad and cannot be used.
		Item cannot be accessed.
		The item does not exist or is not available.
		The server can communicate with the PLC but cannot access the register.
		The server determined the point is not valid.
		The PLC responds that the register does not exist, cannot be read, or cannot be written to.
		The server cannot access a fenced, write-protected, or read-only item.
		The PLC is in a mode that does not permit access to this item.
		The number of data bytes is incorrect but the message is otherwise good.
		The command or op code is invalid but the message is otherwise good.
		The PLC is busy. The server has given up retrying.
0018	No communications	Data communications is down.
		Cannot access the PLC due to a communications error.
		Data is bad and cannot be used.
		The device group is in a slow poll or equivalent mode.

Quality Code	Quality State	Description
		The PLC does not exist and/or is not responding.
		There is no link validating the message.
		There is a lack of resources in the server. A TSR or driver cannot allocate memory.
		There is a lack of resources in the communications link.
		The communications link is off-line.
		All communications channels are in use.
		The network cannot route the message to the PLC.

Tested Communication Driver Hardware and Firmware

You are encouraged to use hardware/firmware that has been tested; however, testing of every possible combination is not feasible. Often the PLC manufacturer uses different model numbers when there are functionally minor differences, such as increased memory or faster CPU. Though we may not have tested these specific models, models within the same PLC series are supported. For example, if we have tested the S7-1211C, then the S7-1200 family is supported. For Siemens hardware/firmware, here are the general guidelines for support:

- Tested hardware is supported.
- Other hardware with the same series number as the test hardware is supported.
- Supported hardware series with latest firmware version from Siemens is supported, unless noted otherwise.

For a list of supported controllers, firmware, and operating systems for Communication Drivers, refer to the Compatibility Matrix, available at the Global Customer Support website: . Enter “SIDirect” in the search entry and select the appropriate version to see the support details.