Wonderware®

ModbusSerial DAServer User's Guide

Version 2.5

Last Revision: 12/18/06

Wonderware

Invensys Systems, Inc.
26561 Rancho Parkway South
Lake Forest, CA 92630 U.S.A.
(949) 727-3200
http://www.wonderware.com

Contents

Before You Begin

About This Book

This book describes how the Wonderware® ModbusSerial DAServer™ is configured and used, after it has been installed. The book is organized in the following fashion:

- **Contents**
- **Introduction**: contains overview information about the ModbusSerial DAServer and the environment in which it works.
- **Configuration**: contains a detailed description of the user-interface elements of this DAServer in addition to its functionality.
- **Item Names**: describes the item-naming conventions for targeted devices.
- **Troubleshooting**: provides information about error messages and codes displayed by this ModbusSerial DAServer.
- **Reference**: describes the DAServer architecture in general.
- **Index**

You can view this document online or you can print it, in part or whole, by using the Adobe Acrobat Reader's print facility. To view this document properly, you must use version 4.0 of the Acrobat Reader.

C H A P T E R 1

Introduction

This chapter provides you with an overview of the Wonderware® ModbusSerial DAServer™, including the application- and bus-level communications protocols, item naming conventions, and DAServer features.

Contents

Overview

The Wonderware ModbusSerial DAServer (referred to as the DAServer throughout the remainder of this user's guide) is a Microsoft® Windows® application program that acts as a communications protocol server. It allows other Windows application programs access to data in the Schneider's Modicon-family of controllers and other Modbus devices, including the TSX Quantum, TSX Momentum, and the Modicon Micro, that are connected to the DAServer through either the computers' serial ports, data modem, or radio modem using the Modbus protocol.

The ModbusSerial DAServer also supports other Generic 4-digit, 5-digit, and 6-digit Modbus controllers supporting the Modbus protocol, provided the messaging used by these controllers conforms to the specifications described in the MODBUS Application Protocol Specification V1.1a.

While the DAServer is primarily intended for use with Wonderware InTouch® (Version 7.11 Patch 02 and later), it may be used by any Microsoft Windows program capable of acting as a DDE, SuiteLink™, or OPC client that can also coexist with FactorySuite™ 2000 and greater.

Generic Modbus Controllers

Starting in version 2.0 of the ModbusSerial DAServer, additional Modbus devices that are not listed in the supported hardware will be supported. These Modbus devices, referred to as Generic Modbus devices/controllers in this document and in the implementation of the DAServer, must be capable of supporting the Modbus protocol function codes, exception codes, and data types as described in Appendix B, The Modbus Protocol.

Compared to the Schneider PLCs listed in Appendix A, Supported DASModbusSerial Hardware and Firmware, the Generic Modbus devices/controllers offer the following additional capabilities:

- Support of Modbus devices that cannot handle multiple-coil write in one message.
- Support of Modbus devices that cannot handle multiple-holding-register write in one message.
- Configurable 4-digit, 5-digit, or 6-digit addressing.
- The maximum addressable register range will be verified by the Modbus devices and does not need to be configured into the DAServer.

Communications Protocols

The ModbusSerial DAServer (Data Access Server) communicates with clients and PLCs using the following different communications protocols:

- Application communications protocols such as OPC, DDE, and SuiteLink to communicate with clients located on either local or remote nodes.
- Modbus protocol supporting all function codes to communicate with the Modicon controllers. This all-function-codes-supporting Modbus protocol is communicated as follows:
  - Over the computers' serial ports and cables. A Serial 232/485 connection is a prerequisite.
  - Through a data modem and line. A Hayes-compatible serial data (phone) modem is required.
  - Over a radio modem and station. The Hayes-compatible serial radio modem is supported; it must have the capabilities to accept commands issued to the phone modems.

Note The ModbusSerial DAServer supports both RS232 and RS485 data modem and radio modem.

Note A serial port or data modem can have more than one controller connected to it. For the best performance and reliability, the use of data or radio modems that support the Modbus protocol is highly recommended.

For more information about the DAServer architecture, see the Reference section.

Application Communications Protocols

The DAServer utilizes the following application communications protocols to communicate with the clients.

OPC

OPC (OLE for Process Control) is a non-proprietary set of standard interfaces based upon Microsoft’s OLE/COM technology. This standard makes possible interoperability between automation/control applications, field systems/ devices, and business/office applications.

Avoiding the traditional requirement of software/application developers to write custom drivers to exchange data with field devices, OPC defines a common, high-performance interface that permits this work to be done once, and then easily reused by HMI, SCADA, control and custom applications.

Over the network, OPC uses DCOM (Distributed COM) for remote communications.

SuiteLink

SuiteLink uses a TCP/IP-based protocol and is designed specifically to meet industrial needs such as data integrity, high throughput, and easier diagnostics. This TCP/IP standard is supported on Windows NT and Windows NT-technology-based operating systems (for example, Windows 2003, Windows 2000, Windows XP, and Windows XP Embedded).

SuiteLink is not a replacement for DDE, FastDDE, or NetDDE. The protocol used between a client and a server depends on your network connections and configurations. SuiteLink provides the following features:

- Value Time Quality (VTQ) places a timestamp and quality indicator on all data values delivered to VTQ-aware clients.
- Extensive diagnostics of the data throughput, server loading, computer resource consumption, and network transport are made accessible through the operating system’s performance monitor. This feature is critical for the operation and maintenance of distributed industrial networks.
- Consistent high data volumes can be maintained between applications regardless if the applications are on a single node or distributed over a large node count.
- The network transport protocol is TCP/IP using Microsoft’s standard WinSock interface.

FastDDE

FastDDE provides a means of packing many proprietary Wonderware Dynamic Data Exchange messages into a single Microsoft DDE message. This packing improves efficiency and performance by reducing the total number of DDE transactions required between a client and a server.

Although Wonderware's FastDDE has extended the usefulness of DDE for our industry, this extension is being pushed to its performance constraints in distributed environments.

DDE

DDE is a communications protocol developed by Microsoft to allow applications in the Windows environment to send/receive data and instructions to/from each other. It implements a Client/Server relationship between two concurrently running applications.

The server application provides the data and accepts requests from any other application interested in its data. Requesting applications are called clients. Some applications such as InTouch and Microsoft Excel can simultaneously be both a client and a server.

NetDDE

NetDDE is a communications protocol that extends the standard DDE functionality to include communications over local area networks and through serial ports. Network extensions are available to allow DDE links between applications running on different computers connected via networks or modems.

For example, NetDDE supports DDE between applications running on IBM-compatible computers connected via LAN or modem, and DDE-aware applications running on non-IBM-compatible computers under operating environments such as VMS and UNIX.

Bus Communications Protocols

The following bus-level protocol is supported in the ModbusSerial DAServer:

- Modbus protocol that supports all function codes

Note The Modbus protocol must be installed on your computer and configured for the PLC with which you wish to communicate.

Modbus Protocol

The Modbus protocol is an open-data-communications network that sends messages over a variety of communications links/connection types. Two modes of communications within the Modbus protocol (pertaining to Modbus networks only) are utilized.

- **10-bit ASCII**: In the ASCII mode, Longitudinal Redundancy Checking (LRC) is used.
  - LRC checks the contents of the message, exclusive of the beginning : (colon) and ending CRLF pair. It is applied regardless of any parity-checking method used.
  - The main advantage of the ASCII transmission mode is its capability to allow time intervals of up to one (1) second to occur between characters without causing an error.
- **11-bit RTU**: In the RTU mode, Cyclic Redundancy Checking (CRC) is used.
  - The CRC field checks the contents of the entire message. It is applied regardless of any parity-checking method used.

- The RTU transmission mode's main advantage is its greater character density that allows better data throughput than the ASCII mode, for the same baud rate.

The transmission mode ASCII or RTU defines the bit contents of the message boxes transmitted serially on those Modbus networks. It determines how information will be packed into the message boxes and decoded.

Controllers can be set up to communicate using either one of the two transmission modes.

Note More information on the Modbus Protocol is provided in Appendix B, The Modbus Protocol. However, to better understand how to read and write data to the different Modicon controllers please refer to the Modicon Modbus Protocol Reference Guide (PI-MBUS-300) from Schneider Electric.

This version of the ModbusSerial DAServer supports the following connections from a COM port of the computer to the controllers:

- Serial 232 and 485 multi-drop connections
- Modem connection (Data Modem and Radio Modem)

Serial 232 or 485 Multi-Drop Connection

The Modbus protocol sends messages over an RS232 or RS485 asynchronous serial communications port that inputs and outputs data alternately. The Modbus protocol can be used in either a simple point-to-point manner from your computer to a device, or to a network of up to 247 slave devices. This communications links your computer directly to a device without using a modem.

For controllers directly connected to the serial port, the DAServer will not provide retry attempts or other special provisions to open the serial port. Once the serial port is opened, it is assumed that the serial connection is kept alive until the DAServer is terminated.

Modem Connection

The ModbusSerial DAServer allows two types of connectivity, permanent connection and on-demand connection, for the controllers connected to either the data modem or radio modem.

Permanent Connection

Once a connection to the controllers is made, the connection will be kept alive even when no data is being acquired from the controllers. If the connection fails for whatever reasons, the quality of associated data items will be changed to BAD or UNCERTAIN.

On-Demand Connection

In this mode, the DAServer will open a connection when data needs to be sent or received from the controllers. Once the data is downloaded from the controllers, the connection will be terminated. If the connection terminates because there is no outstanding requests to the controllers, the quality of the associated data items will not change; usually it is GOOD. If the connection terminates for any other reasons, the quality of the associated data items will be changed to BAD or UNCERTAIN.

Two types of modems are supported by the ModbusSerial DAServer:

- Hayes-Compatible Serial Data (Phone) Modem
- Hayes-Compatible Serial Radio Modem

Hayes-Compatible Serial Data (Phone) Modem

The Modbus protocol can send messages over an RS232 or RS485 asynchronous serial communications port that has a Hayes-compatible data modem connected to it. Using the Hayes-compatible data modem the Modbus protocol can be used in either a simple point-to-point manner from your computer to a device with its own data modem, or in a network of up to 247 slave devices controlled by one data modem.

This modem uses the AT Command Set language developed by Hayes Microcomputer Products. Any modem that recognizes Hayes modem commands is considered to be Hayes-compatible. This is very useful because most communications programs use Hayes modem commands. Most of the modems manufactured today are Hayes-compatible.

Note For higher reliability, the Modbus ASCII mode should be used when communicating through a standard voice modem. The use of a phone modem that supports the Modbus protocol is highly recommended.

Hayes-Compatible Serial Radio Modem

The Modbus protocol can send messages over an RS232 or RS485 asynchronous serial communications port that has a Hayes-compatible radio modem connected to it. Using the Hayes compatible radio modem the Modbus protocol can be used in either a simple point-to-point manner from your computer to a device with its own radio modem, or in a network of up to 247 slave devices controlled by one radio modem.

This modem also uses the AT Command Set language.

Note For the best performance, the use of a radio modem that supports the Modbus protocol is highly recommended.

Accessing Items via the DAServer

The method for accessing items through the DAServer depends on the communications protocol being used.

OPC

In the case of OPC communications, the protocol addresses an element of data in a conversation with six characteristics: node name, program name, group name, device group, link name, and item name.

- The node name (required for remote access) and device group are optional.
- A fully qualified OPC Item name (ItemID) is composed of the link name and item name.
- All other characteristics are specified through separate DAServer means.

To access an OPC item, the OPC client needs to connect to the DAServer (either in-process or out-of-process) and create an OPC group defining the data-acquisition properties for the collection of items to be added. OPC groups can be either public or private. Public OPC groups are shared across multiple clients, whereas private OPC groups are local to a single client. Optionally, a device group, which indicates the access path to the items for read/write, can be specified from the DAServer.

The following briefly describes each characteristic of the OPC protocol:

- **node name**: Computer (host) name identifying a specific node on the network (for Remote Access ONLY).
- **program name**: The registered OPC server name uniquely identifying a specific server (ProgID). For this DAServer, the program name is **ArchestrA.DASMBSerial.2**.
- **group name**: The OPC group created from the client for organizing a collection of items logically with the same data acquisition properties between the client and the server, such as update rate.
- **device group**: Meaningful names configured in the DAServer under a specific controller for the common custom attributes between the DAServer and the device, such as update interval. If not specified from the client, the default device group using the global configuration attribute values from the DAServer is assumed. Functionally, a device group is equivalent to an access path (optional).
- **link name**: The set of hierarchy node names, representing the specific devices on a communications path link from the hierarchy root to a specific controller as configured for this DAServer under the DAServer Manager, separated by delimiters.
- **item name**: A specific data element, the leaf of the hierarchy tree of this DAServer, within the specified group. For example, when using this DAServer, an item can be a relay, timer, counter, register, and so on, in the controller.

DDE/SuiteLink

In the case of DDE/SuiteLink communications, the protocol addresses an element of data in a conversation that uses a four-part naming convention that includes the node name, application name, topic name, and item name. The fully qualified DDE/SuiteLink naming convention includes all four parts, although the node name part (required for remote access only) is optional. The following briefly describes each portion of this naming convention:

- **node name**: Computer (host) name identifying a specific node on the network (for Remote Access ONLY).
- **application name**: The name of the Windows program (this DAServer) that will be accessing the data element. In the case of data coming from or going to the Modicon devices via the DDE/SuiteLink PlugIn of this DAServer, the application name portion of the address is **DASMBSerial**.
- **topic name**: Meaningful names are configured in the DAServer to identify specific devices. These names are then used as the topic names in all conversations with that device. For example, **Line 1 Wrapper**. Topic name maps to a device group defined in the DAServer.

Note You can define multiple device-group (topic) names for the same device (PLC) to poll different points at different rates.

- **item name**: A specific data element within the specified topic. For example, when using this DAServer, an item can be a relay, timer, counter, register, and so on, in the PLC.

Note The term "point" is used interchangeably with the term "item" in this user's guide.

For more information on item/point names, see the Item Names section.

Features

The Wonderware ModbusSerial DAServer provides the following features:

- The ability to communicate over multiple application-level protocols at the same time.
- The ability to add new application-level protocols on the fly.
- The ability to be configured remotely.
- New, robust diagnostic abilities.
- Additional server-specific diagnostics.
- XML storage.
  For example, the storage of the .aacfg file that has the details of all the device groups and device items that can be stored in XML.
- Full existing item-name space.
- Log of errors, warnings, traces, and Modbus messages, individually adjustable for reading and writing.
- OPC browsing.

For more in-depth information on the DAServer architecture, see the Reference section.

Demo Mode

You can install a fully functioning version of this DAServer for demonstration purposes without a license. Demo Mode allows you to test the functionality of the DAServer for 120 minutes. After that time, you must install a license to continue using the DAServer.

When you first start this DAServer, it checks for a license. If the DAServer cannot find a valid license installed on the local computer, it logs a warning message indicating a valid license cannot be retrieved, and enters Demo mode. Thereafter, the DAServer repeats its request for the license every 30 seconds. If no license is found, the DAServer again logs a warning message on the issue.

This process is repeated for 120 minutes, after which the server stops updating read/write on all device items (read from cache is allowed, but all non-system data would receive Bad quality status) and reject the addition of any new items. The DAServer continues to request for a license. Clients continue to function normally (for instance, you can still add or remove an item, but its quality is set to Bad until a license is obtained).

Note Use the $SYS$Licensed system item, a read-only Boolean item, to check the status of your license: True for Licensed or in Demo Mode and False for Not Licensed.

If you subsequently add a license to the License Manager, the DAServer logs a message acknowledging the license, switches out of Demo mode, and runs normally.

Note Once a DAServer obtains a valid license, it no longer checks for a license. However, if your license expires, your DAServer will no longer function when you restart the DAServer.

C H A P T E R 2

Configuration

Once the Wonderware ModbusSerial DAServer has been installed, a small amount of configuration is required. This configuration is performed using the DAServer Manager hosted in the System Management Console after it is started through the **Programs** menu of the Windows **Start** button.

Before the DAServer is activated, the device hierarchy, simulating the physical hardware layout, must first be built to establish communications to each of the controllers. Once the ModbusSerial hierarchy has been built, the respective devices for communications can be configured. Finally, the desired Device Groups for each controller may be created.

Note To run the ModbusSerial DAServer as a service, use the shortcut menu on the **DAServer name** and select **Configure As Service**. You can configure it as an auto service or manual service. For more information about configuring your DAServer as a service see the Activation/Deactivation/Service Component of the DAServer Manager documentation.

Note The ModbusSerial DAServer must be run as a service if the SuiteLink plug-in is installed.

Contents

Getting Started Quickly with the DAServer

This section briefly describes the procedures required to prepare the ModbusSerial DAServer for use. Detailed descriptions of each step can be found in subsequent sections of this documentation. This section is intended for people who are familiar with DAServers.

Note If you are not familiar with DAServer functionality, please proceed to the more-detailed procedures following this section.

The following procedures assume that you have:

- Configured the PLC with which you wish to communicate.

To prepare the ModbusSerial DAServer

1. Install the Wonderware ModbusSerial DAServer on Windows by running the **Setup.exe** program.

   Note The DAServer installation instructions are included in a separate Help file (.chm extension).

   - Accept all the default settings during installation.

   Important! Since there are no default values for security settings, you must take note of the User Name and password selected during the install.

2. Start the Wonderware DAServer Manager by selecting the **Programs** menu from the **Start** button on the taskbar.
3. Navigate to the **Wonderware** folder that contains the System Management Console, then click **System Management Console**.
4. From the **System Management Console**, find the ModbusSerial DAServer in the **DAServer Manager** tree, the location in which it is installed.
   - Under the Local branch node, the name of the DAServer is **ArchestrA.DASMBSerial.2**.

   Note See the DAServer Manager documentation for general information about working in this snap-in environment.

5. The new ModbusSerial DAServer must now be configured.
   - Before proceeding, determine the hierarchical structure of the network/PLC environment to which you plan to connect.

Important! The ModbusSerial DAServer can be configured to support a maximum of 32 COM_PORT objects, 32 DataMODEM objects, and 32 RadioMODEM objects. Each DataMODEM object supports a maximum of 100 LINE objects. Each RadioMODEM object supports a maximum of 200 STATION objects. On each COM_PORT, DataMODEM, or RadioMODEM, a total of 247 PLCs can be multi-dropped if RS485 connection is used.

6. Right-click the **Configuration** object that already exists in the tree, and select one of the following from the shortcut menu:
   - **Add COM_PORT Object**
     - A new COM_PORT object is created as a node in the hierarchy tree and is named **New_COM_PORT_000** by default.
     - The **New_COM_PORT_000 Parameters** configuration view is displayed at the right pane.

Important! Disconnecting all clients from the DAServer does not release the used COM_PORT. To release the COM_PORT, the DAServer needs to be deactivated and then reactivated.

- **Add DataMODEM Object**
  - A new DataMODEM obejct is created as a node in the hierarchy tree and is named **New_DataMODEM_000** by default.
  - The **New_DataMODEM_000 Parameters** configuration view is displayed at the right pane.
- **Add RadioMODEM Object**
  - A new RadioMODEM obejct is created as a node in the hierarchy tree and is named **New_RadioMODEM_000** by default.
  - The **New_RadioMODEM_000 Parameters** configuration view is displayed at the right pane.

7. **COM_PORT_000** Object

   Right-click on the **New_COM_PORT_000** object, and from the shortcut menu, select one of the following:

- **Add QuantumPLC Object** for the Quantum controllers.
  - A new QuantumPLC obejct is created as a node in the hierarchy tree and is named as **New_QuantumPLC_000** by default.
  - The **New_QuantumPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add TSXMomentumPLC Object** for the TSX Momentum controllers.
  - A new TSXMomentumPLC obejct is created as a node in the hierarchy tree and is named **New_TSXMomentumPLC_000** by default.
  - The **New_TSXMomentumPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add ModiconMicroPLC Object** for the ModiconMicro controllers.
  - A new ModiconMicroPLC object is created as a node in the hierarchy tree and is named **New_ModiconMicroPLC_000** by default.
  - The **New_ModiconMicroPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add ModbusPLC Object** for the Generic Modbus 4-Digit, 5-Digit, or 6-Digit controllers.
  - A new ModbusPLC object is created as a node in the hierarchy tree and is named **New_ModbusPLC_000** by default.

- The **New_ModbusPLC_000 Parameters** configuration view is displayed at the right pane.

DataMODEM_000 Object

Right-click on the **New_DataMODEM_000** object, and from the shortcut menu, select **Add LINE Object**.

- A new LINE object is created as a node in the hierarchy tree and is named **New_LINE_000** by default.
- The **New_LINE_000 Parameters** configuration view is displayed at the right pane.

Right-click on the **New_LINE_000** object, and from the shortcut menu, select one of the following:

- **Add QuantumPLC Object** for the Quantum controllers.
  - A new QuantumPLC obejct is created as a node in the hierarchy tree and is named **New_QuantumPLC_000** by default.
  - The **New_QuantumPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add TSXMomentumPLC Object** for the TSX Momentum controllers.
  - A new TSXMomentumPLC obejct is created as a node in the hierarchy tree and is named **New_TSXMomentumPLC_000** by default.
  - The **New_TSXMomentumPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add ModiconMicroPLC Object** for the ModiconMicro controllers.
  - A new ModiconMicroPLC object is created as a node in the hierarchy tree and is named **New_ModiconMicroPLC_000** by default.
  - The **New_ModiconMicroPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add ModbusPLC Object** for the Generic Modbus 4-Digit, 5-Digit, or 6-Digit controllers.
  - A new ModbusPLC object is created as a node in the hierarchy tree and is named **New_ModbusPLC_000** by default.
  - The **New_ModbusPLC_000 Parameters** configuration view is displayed at the right pane.

RadioMODEM_000 Object

Right-click on the **New_RadioMODEM_000** object, and from the shortcut menu, select **Add STATION Object**.

- A new STATION object is created as a node in the hierarchy tree and is named **New_STATION_000** by default.

- The **New_STATION_000 Parameters** configuration view is displayed at the right pane.

Right-click on the **New_STATION_000** object, and from the shortcut menu, select one of the following:

- **Add QuantumPLC Object** for the Quantum controllers.
  - A new QuantumPLC obejct is created as a node in the hierarchy tree and is named as **New_QuantumPLC_000** by default.
  - The **New_QuantumPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add TSXMomentumPLC Object** for the TSX Momentum controllers.
  - A new TSXMomentumPLC obejct is created as a node in the hierarchy tree and is named **New_TSXMomentumPLC_000** by default.
  - The **New_TSXMomentumPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add ModiconMicroPLC Object** for the ModiconMicro controllers.
  - A new ModiconMicroPLC object is created as a node in the hierarchy tree and is named **New_ModiconMicroPLC_000** by default.
  - The **New_ModiconMicroPLC_000 Parameters** configuration view is displayed at the right pane.
- **Add ModbusPLC Object** for the Generic Modbus 4-Digit, 5-Digit, or 6-Digit controllers.
  - A new ModbusPLC object is created as a node in the hierarchy tree and is named **New_ModbusPLC_000** by default.
  - The **New_ModbusPLC_000 Parameters** configuration view is displayed at the right pane.

8. Configure the respective device objects, created in the preceding steps, with the appropriate parameter values, if applicable.
   - Optionally, the desired device groups can be created under the **Device Groups** tabbed page with each of the PLC objects.
   - Desired device items can also be optionally created under the **Device Items** tabbed page with each of the PLC objects.

Note The hierarchy entry is added in the "edit mode," providing a convenient place for you to appropriately describe components of your specific hardware environment. Both hierarchy node name and device group name are numerically sequenced by default. They can be renamed at any time.

The DAServer is now ready for use. In order to use the DAServer, you must activate it from the **DAServer Manager** using either the shortcut menu's **Activate Server** command from the **ArchestrA.DASMBSerial.2** node, or an OPC Client.

Note To run the ModbusSerial DAServer as a service, right-click on the **DAServer name** (**ArchestrA.DASMBSerial.2**) and select **Configure As Service** from the shortcut menu. You can configure it as an auto service or manual service. For more information about configuring your DAServer as a service, see the Activation/Deactivation/Service Component of the DAServer Manager documentation.

Configuring the DAServer

Note This DAServer is hosted by the DAServer Manager, a Microsoft Management Console (MMC) snap-in, which is a part of the ArchestrA System Management Console (SMC) suite of utilities. Many high-level functions and user-interface elements of the DAServer Manager are universal to all DAServers, and **only** the documentation for the DAServer Manager contains descriptions of those universal functions/UI elements. Therefore, reading the documentation for both the MMC and the DAServer Manager is critical to understanding this user's guide. To read the documentation about the MMC and DAServer Manager, right-click the **DAServer Manager** icon and select the **Help** menu. Both the MMC Help and the DAServer Manager Help are displayed. An Adobe Acrobat version of the DAServer Manager documentation (DAServerManager.pdf) is also available in the CD-ROM directory\User Docs\English.

Note The shortcut menu items described in this document typically represent only a subset of any actual shortcut menu. Most items in each shortcut menu are standard Windows commands. For more information about those commands, please see **Help**, by right-clicking the **System Management Console** icon.

Note For more information on the Modbus protocol and to better understand how to read and write data to the different Modicon controllers, please refer to the Modicon Modbus Protocol Reference Guide (PI-MBUS-300) from Schneider Electric.

To prepare the ModbusSerial DAServer

1. Install the Wonderware ModbusSerial DAServer on Windows by running the **Setup.exe** program.

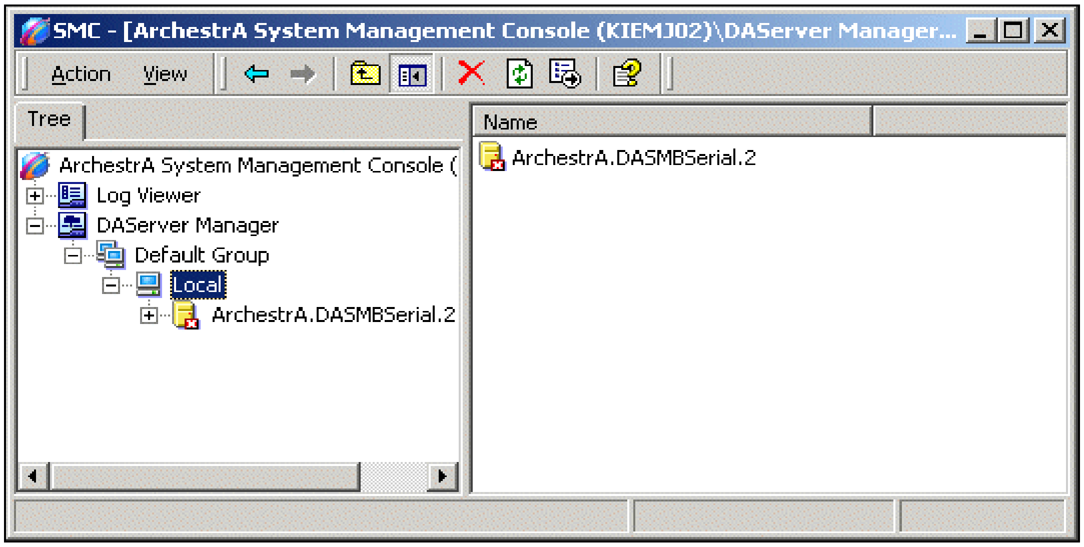
   Note DAServer installation instructions are included in a separate Help file (.chm extension).

2. Accept all the default settings during the installation.

   Important! Since there are no default values for security settings, you must take note of the User Name and password selected during the install.

3. After the DAServer has been installed, start the System Manager Console by clicking the **Start** button on the Windows taskbar, and pointing to **Programs**.

4. Point to the **Wonderware** folder that contains the System Management Console, then click **System Management Console**.
5. From the System Management Console tree, click on **DAServer Manager**.
6. Click on **Default Group**, then the **Local** node.
   - Under the Local node, the DAServer name is **ArchestrA.DASMBSerial.2**.


SMC - [ArchestrA System Management Console (KIEMJ02)\DAServer Manager...
Action
View
Tree
Name
ArchestrA System Management Console (
Log Viewer
DAServer Manager
Default Group
Local
ArchestrA.DASMBSerial.2
ArchestrA.DASMBSerial.2

Note See the DAServer Manager documentation for general information about working in this snap-in environment.

7. Before the DAServer is started, you must first build the device hierarchy to establish communications to each of the controllers.

Important! For step-by-step procedures on how to build the device hierarchy, see "ModbusSerial Hierarchy in the DAServer Manager."

Note Selecting the **Configuration** object of the hierarchy tree displays the **Global Parameters** configuration view for this DAServer. The default Poke Mode settings for the DAServer is Optimization mode. Configure all other global parameters as required for this DAServer. For more information about the **Global Parameters** dialog box, including descriptions of the different Poke Modes, see the DAServer Manager documentation. You can access that documentation by clicking the **DAServer Manager** icon and selecting the Help topics on the **Help** menu, and then navigating through the **DAServer Manager** book.

Important! Any Global Parameters that appear dimmed are either not supported or cannot be configured for this DAServer. Simulation Mode is not supported.

8. When the ModbusSerial hierarchy build has been completed, you can start configuring the respective devices for communications.

9. You may create the desired Device Groups for each controller by:
   - Navigating to the object of interest in the **DAServer Manager** tree view.
   - Clicking on the **Device Groups** tab.
   - Right-clicking in the **Device Groups** dialog box and selecting the **Add** command from the shortcut menu.

   Important! For step-by-step procedures on configuring Device Groups, please see "Device Group Definitions."

10. Finally, you may create the desired Device Items for each controller by:
    - Navigating to the object of interest in the **DAServer Manager** tree view.
    - Clicking on the **Device Items** tab.
    - Right-clicking in the **Device Items** dialog box and selecting the **Add** command from the shortcut menu.

    Important! For step-by-step procedures on configuring Device Items, please see "Device Item Definitions."

The DAServer is now ready for use. In order to use the DAServer, you must activate it.

Note When the DAServer or any of its configuration views is selected and you open multiple instances of the DAServer Manager, the DAServer Manager will place the configuration views from the subsequent instances of the same DAServer into read-only mode. Access to the second instance of the DAServer will resume after the first one has been deselected or closed. Likewise, access to the DAServer configuration will be unlocked for the next instance in this order.

- If you are using an OPC Client, it can auto-start the DAServer.
- If you are using DDE/SuiteLink, you must start the DAServer either as a manual or automatic service.
- To activate the DAServer, right-click on **ArchestrA.DASMBSerial.2** and select **Activate Server** from the shortcut menu.

Note To run the ModbusSerial DAServer as a service, right-click on the **DAServer name** (**ArchestrA.DASMBSerial.2**) and select **Configure As Service** from the shortcut menu. You can configure it as an auto service or manual service. For more information about configuring your DAServer as a service, see the Activation/Deactivation/ Service Component of the DAServer Manager documentation.

ModbusSerial Hierarchy in the DAServer Manager

The ModbusSerial DAServer can be configured to support the following objects:

- 32 COM_PORT objects, 32 DataMODEM objects, and 32 RadioMODEM objects, making a total of 96 objects in combination of the three.
- 100 LINE objects per DataMODEM object.
- 200 STATION objects per RadioMODEM object.
- A total of 247 Quantum, TSX Momentum, Modicon Micro, or Generic Modbus (4-Digit, 5-Digit, and 6-Digit) PLCs, or a combination of the four for each COM_PORT, each Line, and each Station.

Note Before attempting to configure your DAServer, you should determine the hierarchical structure of your network/PLC environment.

The server-specific configuration portion of the ModbusSerial DAServer hierarchy tree under the DAServer Manager starts at either one of the following objects:

- COM_PORT Object
- DataMODEM Object
- RadioMODEM Object

COM_PORT Object

This section describes how the COM_PORT object is configured after the DAServer has been installed.

The COM_PORT object is configured from the **Configuration** branch of the DAServer Manager hierarchy. Under this **COM_PORT** branch, the following PLC objects can be configured:

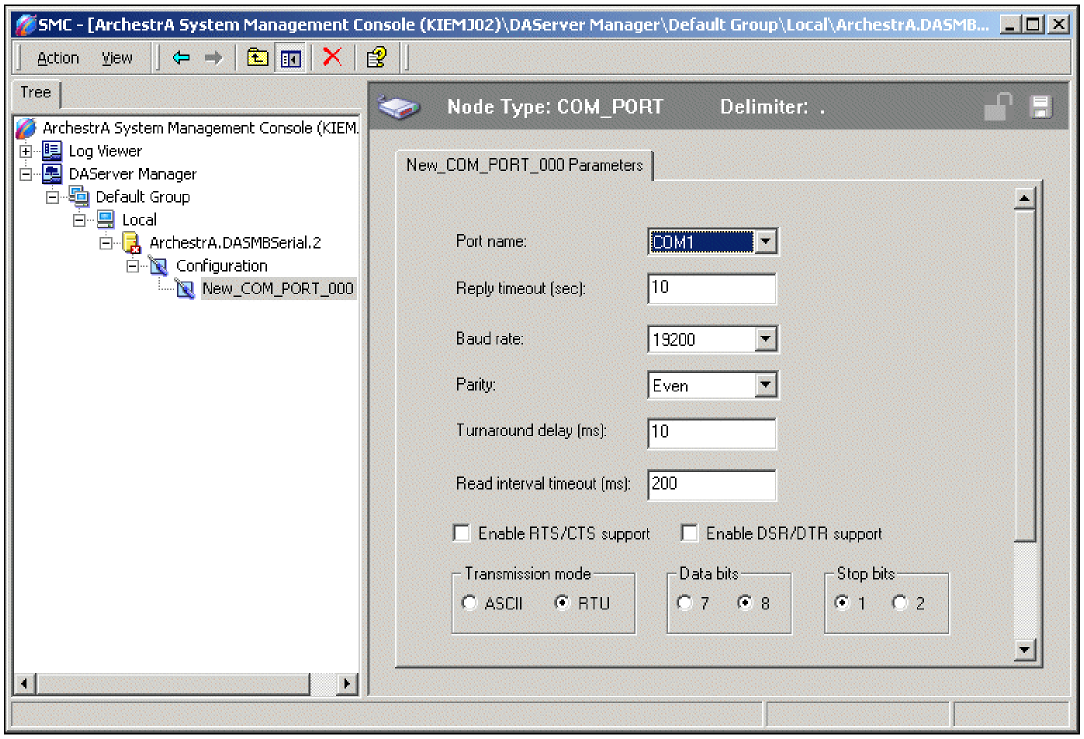
- QuantumPLC Object
- TSXMomentumPLC Object
- ModiconMicroPLC Object
- ModbusPLC Object

Important! If you subsequently clear your configuration hierarchy, you must create this COM_PORT object from the **Configuration** branch of the hierarchy. From this point, all of the following instructions apply.

Important! Disconnecting all clients from the DAServer does not release the used COM_PORT. To release the COM_PORT, the DAServer needs to be deactivated and then reactivated.

To create COM_PORT objects from the Configuration branch

1. Right-click on **Configuration**.
2. Select **Add COM_PORT Object** from the shortcut menu.
   - A new COM_PORT object is created as a node in the hierarchy tree, and it is named **New_COM_PORT_000** by default.
   - 32 of these COM_PORT objects can be created for each ModbusSerial DAServer.
3. Rename the newly created object as appropriate.
   - The **New_COM_PORT_000 Parameters** configuration view is displayed at the right pane.


SMC - [ArchestrA System Management Console (KIEMJ02)\DAServer Manager\Default Group\Local\ArchestrA.DASMB...
Action View
Tree
ArchestrA System Management Console (KIEM.
Log Viewer
DAServer Manager
Default Group
Local
ArchestrA.DASMBSerial.2
Configuration
New_COM_PORT_000
Node Type: COM_PORT
Delimiter: .
New_COM_PORT_000 Parameters

Port name: COM1
Reply timeout (sec): 10
Baud rate: 19200
Parity: Even
Turnaround delay (ms): 10
Read interval timeout (ms): 200
☐ Enable RTS/CTS support ☐ Enable DSR/DTR support
Transmission mode: ASCII [✓ RTU]
Data bits: 7 [✓ 8]
Stop bits: [✓ 1] 2

This configuration view has 11 configurable elements:

- **Port name**: Select a communications port used by the DAServer to communicate with the PLC.
  - A Maximum of 96 COM ports (COM1 through COM96) can be created.
  - The default port is COM1.
- **Reply timeout (sec)**: Enter the amount of time (in seconds) that the DAServer will wait for an acknowledgment.
  - Allowable values are 1 (one) to 300 seconds.
  - The default value is 10 seconds.

- **Baud rate**: Select the number of Serial bit rates setting, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 115200, 128000 matching the device.
  - The minimum value is 75.
  - The maximum value is 128000.
  - The default value is 19200.
- **Parity**: Select one of five settings that matches the configuration of the Modicon device.
  - None
  - Odd
  - Even
  - Mark
  - Space
  - The default value is Even.
- **Turnaround Delay (ms)**: Specify the amount of time, in milliseconds, between the last send/receive message and the next receive/send message to/from the PLC that the DAServer will wait before it sends an error message to the PLC.
  - The minimum value is 0 (zero).
  - The maximum value is 60000.
  - The default value is 10 milliseconds.

Note If the value is set to 0 (zero), there will not be any turnaround delay.

- **Read interval timeout (ms)**: Specify the maximum amount of time, in milliseconds, for the subsequent character to follow the former (Inter-character delay between two bytes of receiving data) in a message before the message times out.
  - The minimum value is 1 (one).
  - The maximum value is 60000.
  - The default value is 200 milliseconds.
- **Enable RTS/CTS support**: Selection for RTS/CTS support.
  - Checked – to enable RTS/CTS support.
  - Unchecked – to disable RTS/CTS support.
  - The default value is Unchecked.
- **Enable DSR/DTR support**: Selection for DSR/DTR support.
  - Checked – to enable DSR/DTR support.
  - Unchecked – to disable DSR/DTR support.
  - The default value is Unchecked.
- **Transmission mode**: Select the protocol configured for the equipment attached to the communications port.

    - ASCII
    - RTU
    - The default is RTU.

  Transmission mode pertains to the Modbus networks. It defines the bit contents of message boxes transmitted serially on those networks. It also determines how information will be packed into the message boxes and decoded.

  There are two transmission modes supported, ASCII and RTU.

    - **ASCII**: Allows time intervals of up to 1 (one) second to occur between characters without causing an error.
    - **RTU**: Has greater character density that allows better data throughput than the ASCII mode for the same baud rate.

- **Data bits**: The number of data bits is 7 (seven) or 8 (eight). Select the one that matches the configuration of the device.
    - The minimum value is 7 (seven).
    - The maximum value is 8 (eight).
    - The default value is 8 (eight).

Note When the Transmission mode is set to RTU, the Data bits are automatically set to 8 (eight). When the Transmission mode is set to ASCII, the Data bits are automatically set to 7 (seven). The data bits can also be manually set to 7 (seven) or 8 (eight).

- **Stop bits**: The number of stop bits is 1 (one) or 2 (two). Select the number matching the configuration of the device.
    - The default value is 1.

Note If the baud rate is greater than 300, the Stop bits must be set to 1 (one).

DataMODEM Object

This section describes how the DataMODEM object is configured from the **Configuration** branch of the DAServer Manager hierarchy. Directly under this **DataMODEM** branch, the following object can be configured:

- LINE Object

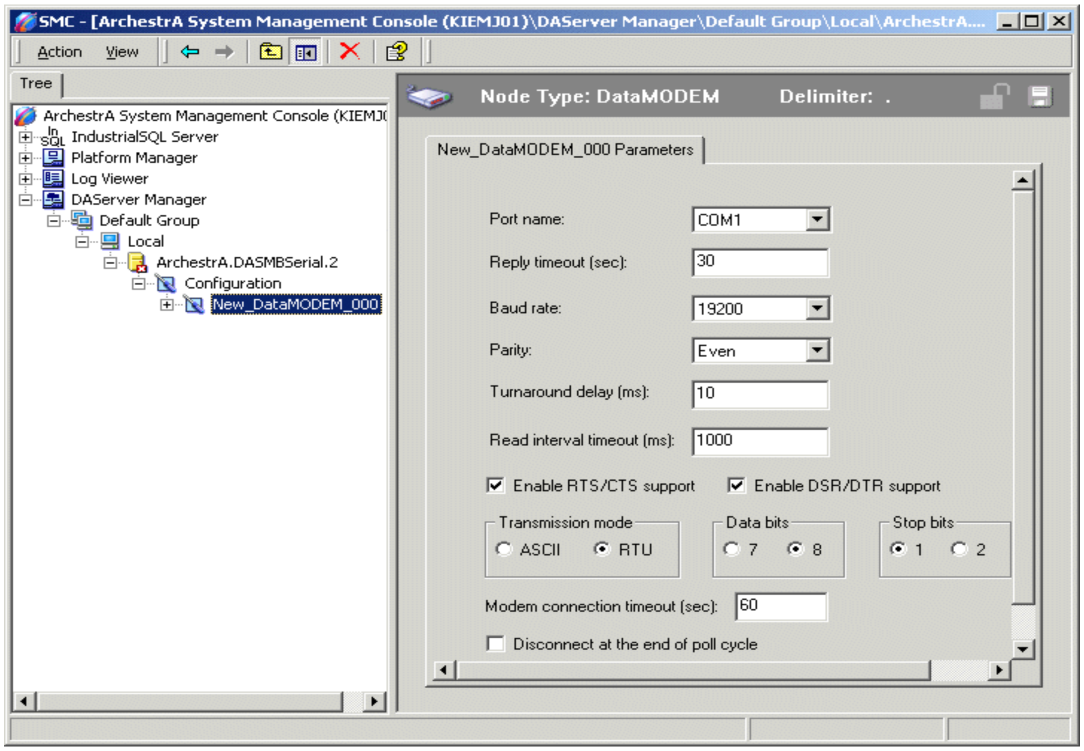
Important! If you subsequently clear your configuration hierarchy, you must create this DataMODEM object from the **Configuration** branch of the hierarchy. From this point, all of the following instructions apply.

To create DataMODEM objects from the Configuration branch

1. Right-click on **Configuration**.
2. Select **Add DataMODEM Object** from the shortcut menu.

- A new DataMODEM object is created as a node in the hierarchy tree, and it is named **New_DataMODEM_000** by default.
- 32 of these DataMODEM objects can be created for each ModbusSerial DAServer.

3. Rename the newly created object as appropriate.
   - The **New_DataMODEM_000 Parameters** configuration view is displayed at the right pane.



Port name: COM1
Reply timeout (sec): 30
Baud rate: 19200
Parity: Even
Turnaround delay (ms): 10
Read interval timeout (ms): 1000
[✓] Enable RTS/CTS support [✓] Enable DSR/DTR support
Transmission mode: ASCII [✓ RTU]
Data bits: 7 [✓ 8]
Stop bits: [✓ 1] 2
Modem connection timeout (sec): 60
[ ] Disconnect at the end of poll cycle

This configuration view has 13 configurable elements:

- **Port name**: Select a communications port used by the DAServer to communicate with the PLC.
  - A Maximum of 96 communications ports (COM1 through COM96) can be created.
  - The default port is COM1.
- **Reply timeout (sec)**: Enter the amount of time (in seconds) that the DAServer will wait for an acknowledgment.
  - Allowable values are 1 (one) to 300 seconds.
  - The default value is 30 seconds.
- **Baud rate**: Select the number of Serial bit rates setting, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 115200, 128000 matching the device.
  - The minimum value is 75.
  - The maximum value is 128000.

  - The default value is 19200.
- **Parity**: Select one of five settings that matches the configuration of the Modicon device.
  - None
  - Odd
  - Even
  - Mark
  - Space
  - The default value is Even.
- **Turnaround Delay (ms)**: Specify the amount of time, in milliseconds, between the last send/receive message and the next receive/send message to/from the PLC that the DAServer will wait before it sends an error message to the PLC.
  - The minimum value is 0 (zero).
  - The maximum value is 60000.
  - The default value is 10 milliseconds.

Note If the value is set to 0 (zero), there will not be any turnaround delay.

- **Read interval timeout (ms)**: Specify the maximum amount of time, in milliseconds, for the subsequent character to follow the former (Inter-character delay between two bytes of receiving data) in a message before the message times out.
  - The minimum value is 1 (one).
  - The maximum value is 60000.
  - The default value is 1000 milliseconds.
- **Enable RTS/CTS support**: Selection for RTS/CTS support.
  - Checked – to enable RTS/CTS support.
  - Unchecked – to disable RTS/CTS support.
  - The default value is Checked.
- **Enable DSR/DTR support**: Selection for DSR/DTR support.
  - Checked – to enable DSR/DTR support.
  - Unchecked – to disable DSR/DTR support.
  - The default value is Checked.
- **Transmission mode**: Select the protocol configured for the equipment attached to the communications port.
  - ASCII
  - RTU

    - The default is RTU.

  Transmission mode pertains to the Modbus networks. It defines the bit contents of message boxes transmitted serially on those networks. It also determines how information will be packed into the message boxes and decoded.

  There are two transmission modes supported, ASCII and RTU.

    - **ASCII**: Allows time intervals of up to 1 (one) second to occur between characters without causing an error.
    - **RTU**: Has greater character density that allows better data throughput than the ASCII mode for the same baud rate.

- **Data bits**: The number of data bits is 7 (seven) or 8 (eight). Select the one that matches the configuration of the device.
    - The minimum value is 7 (seven).
    - The maximum value is 8 (eight).
    - The default value is 8 (eight).

Note When the Transmission mode is set to RTU, the Data bits are automatically set to 8 (eight). When the Transmission mode is set to ASCII, the Data bits are automatically set to 7 (seven). The data bits can also be manually set to 7 (seven) or 8 (eight).

- **Stop bits**: The number of stop bits is 1 (one) or 2 (two). Select the number matching the configuration of the device.
    - The default value is 1.

Note If the baud rate is greater than 300, the Stop bits must be set to 1 (one).

- **Modem connection timeout (sec)**: Specify the amount of time, in seconds, to connect to the data modem. If this period is exceeded, the connection to the modem will be signaled failed.
    - The minimum value is 1 (one).
    - The maximum value is 300.
    - The default value is 60 seconds.
- **Disconnect at the end of poll cycle**: Selection for a disconnect at the end of a poll cycle.
    - Checked – to disconnect at the end of a poll cycle.
    - Unchecked – not to disconnect at the end of a poll cycle.
    - The default value is Unchecked.

Note If the **Disconnect at the end of poll cycle** check box is checked, it implies that the connection mode is OnDemand Connection; if it is not checked, it indicates that the connection mode is a Permanent Connection. For more information on connection modes, see Modem Connection.

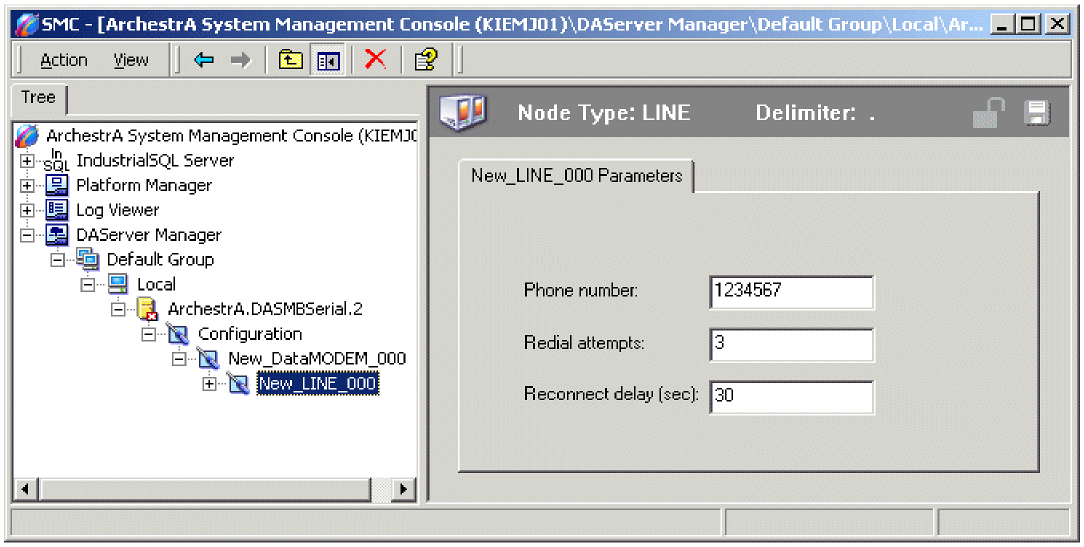
LINE Object

The LINE object is configured from the **DataMODEM** branch of the DAServer Manager hierarchy. Under this **LINE** branch, the following PLC objects can be configured:

- QuantumPLC Object
- TSXMomentumPLC Object
- ModiconMicroPLC Object
- ModbusPLC Object

To create LINE objects from the DataMODEM branch

1. Right-click on **DataMODEM**.
2. Select **Add LINE Object** from the shortcut menu.
    - A new LINE object is created as a node in the hierarchy tree, and it is named **New_LINE_000** by default.
    - 100 of these objects can be created for each DataMODEM object.
3. Rename the newly created object as appropriate.
    - The **New_LINE_000 Parameters** configuration view is displayed at the right pane.



Phone number: 1234567
Redial attempts: 3
Reconnect delay (sec): 30

This configuration view has three configurable elements:

- **Phone number**: Enter the phone number of the remote phone line.
    - The minimum number of digits is 1 (one).
    - The maximum number of digits is 255.

Note The maximum number of Line nodes supported is 100.

- **Redial attempts**: Specify the maximum number of dialing attempts to be made to connect to the remote modem.

- The minimum value is 0 (zero).
- The maximum value is 100.
- The default value is 3 (three).

- **Reconnect delay (sec)**: Specify the minimum amount of waiting time, in seconds, before a line is to be reconnected (after a failure of line is detected).
    - The minimum value is 1 (one).
    - The maximum value is 3600.
    - The default value is 30 seconds.

RadioMODEM Object

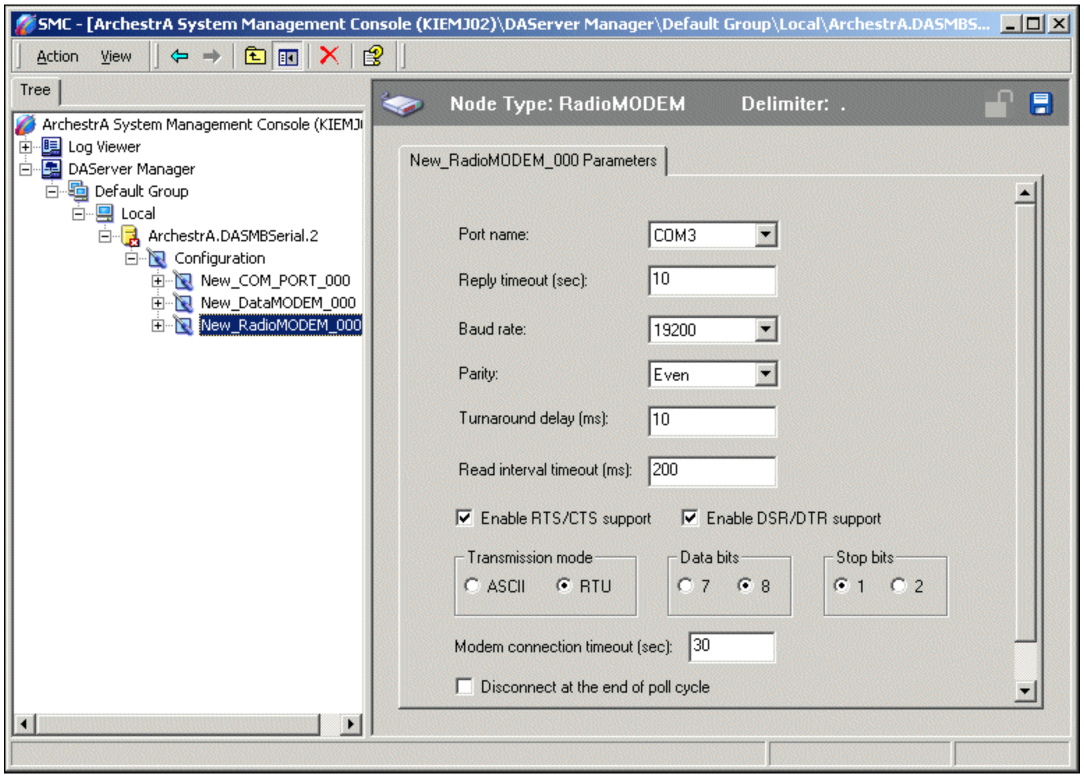
This section describes how the RadioMODEM object is configured from the **Configuration** branch of the DAServer Manager hierarchy. Directly under this **RadioMODEM** branch, the following object can be configured:

- STATION Object

Important! If you subsequently clear your configuration hierarchy, you must create this RadioMODEM object from the **Configuration** branch of the hierarchy. From this point, all of the following instructions apply.

To create RadioMODEM objects from the Configuration branch

1. Right-click on **Configuration**.
2. Select **Add RadioMODEM Object** from the shortcut menu.
    - A new RadioMODEM object is created as a node in the hierarchy tree, and it is named **New_RadioMODEM_000** by default.
    - 32 of these objects can be created for each ModbusSerial DAServer.
3. Rename the newly created object as appropriate.
    - The **New_RadioMODEM_000 Parameters** configuration view is displayed at the right pane.


SMC - [ArchestrA System Management Console (KIEMJ02)\DAServer Manager\Default Group\Local\ArchestrA.DASMBS...
Action
View
Tree
ArchestrA System Management Console (KIEMJ
Log Viewer
DAServer Manager
Default Group
Local
ArchestrA.DASMBSerial.2
Configuration
New_COM_PORT_000
New_DataMODEM_000
New_RadioMODEM_000
Node Type: RadioMODEM
Delimiter: .
New_RadioMODEM_000 Parameters

Port name: COM3

Reply timeout (sec): 10

Baud rate: 19200

Parity: Even

Turnaround delay (ms): 10

Read interval timeout (ms): 200

[✓] Enable RTS/CTS support [✓] Enable DSR/DTR support

Transmission mode: ASCII [✓ RTU]

Data bits: 7 [✓ 8]

Stop bits: [✓ 1] 2

Modem connection timeout (sec): 30

[ ] Disconnect at the end of poll cycle

This configuration view has 13 configurable elements:

- **Port name**: Select a communications port used by the DAServer to communicate with the PLC.
  - A Maximum of 96 communications ports (COM1 through COM96) can be created.
  - The default port is COM1.
- **Reply timeout (sec)**: Enter the amount of time (in seconds) that the DAServer will wait for an acknowledgment.
  - Allowable values are 1 (one) to 300 seconds.
  - The default value is 10 seconds.
- **Baud rate**: Select the number of Serial bit rates setting, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 115200, 128000 matching the device.
  - The minimum value is 75.
  - The maximum value is 128000.
  - The default value is 19200.
- **Parity**: Select one of five settings that matches the configuration of the Modicon device.
  - None
  - Odd
  - Even

    - Mark
    - Space
    - The default value is Even.
- **Turnaround Delay (ms)**: Specify the amount of time, in milliseconds, between the last send/receive message and the next receive/send message to/from the PLC that the DAServer will wait before it sends an error message to the PLC.
    - The minimum value is 0 (zero).
    - The maximum value is 60000.
    - The default value is 10 milliseconds.

Note If the value is set to 0 (zero), there will not be any turnaround delay.

- **Read interval timeout (ms)**: Specify the maximum amount of time, in milliseconds, for the subsequent character to follow the former (Inter-character delay between two bytes of receiving data) in a message before the message times out.
    - The minimum value is 1 (one).
    - The maximum value is 60000.
    - The default value is 200 milliseconds.
- **Enable RTS/CTS support**: Selection for RTS/CTS support.
    - Checked – to enable RTS/CTS support.
    - Unchecked – to disable RTS/CTS support.
    - The default value is Checked.
- **Enable DSR/DTR support**: Selection for DSR/DTR support.
    - Checked – to enable DSR/DTR support.
    - Unchecked – to disable DSR/DTR support.
    - The default value is Checked.
- **Transmission mode**: Select the protocol configured for the equipment attached to the communications port.
    - ASCII
    - RTU
    - The default is RTU.

  Transmission mode pertains to the Modbus networks. It defines the bit contents of message boxes transmitted serially on those networks. It also determines how information will be packed into the message boxes and decoded.

  There are two transmission modes supported, ASCII and RTU.

    - **ASCII**: Allows time intervals of up to 1 (one) second to occur between characters without causing an error.

  - **RTU**: Has greater character density that allows better data throughput than the ASCII mode for the same baud rate.
- **Data bits**: The number of data bits is 7 (seven) or 8 (eight). Select the one that matches the configuration of the device.
  - The minimum value is 7 (seven).
  - The maximum value is 8 (eight).
  - The default value is 8 (eight).

Note When the Transmission mode is set to RTU, the Data bits are automatically set to 8 (eight). When the Transmission mode is set to ASCII, the Data bits are automatically set to 7 (seven). The data bits can also be manually set to 7 (seven) or 8 (eight).

- **Stop bits**: The number of stop bits is 1 (one) or 2 (two). Select the number matching the configuration of the device.
  - The default value is 1.

Note If the baud rate is greater than 300, the Stop bits must be set to 1 (one).

- **Modem connection timeout (sec)**: Specify the amount of time, in seconds, to connect to the data modem. If this period is exceeded, the connection to the modem will be signaled failed.
  - The minimum value is 1 (one).
  - The maximum value is 300.
  - The default value is 30 seconds.
- **Disconnect at the end of poll cycle**: Selection for a disconnect at the end of poll cycle.
  - Checked – to disconnect at the end of a poll cycle.
  - Unchecked – not to disconnect at the end of a poll cycle.
  - The default value is Unchecked.

Note If the **Disconnect at the end of poll cycle** check box is checked, it implies that the connection mode is OnDemand Connection; if it is not checked, it indicates that the connection mode is a Permanent Connection. For more information on connection modes, see Modem Connection.

STATION Object

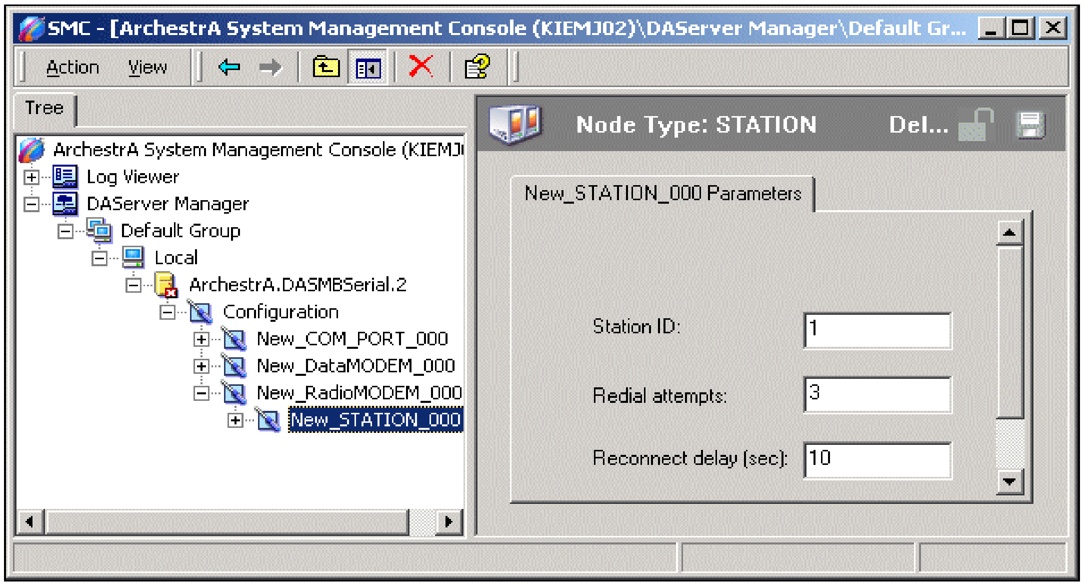
The STATION object is configured from the **RadioMODEM** branch of the DAServer Manager hierarchy. Under this **STATION** branch, the following PLC objects can be configured:

- QuantumPLC Object
- TSXMomentumPLC Object
- ModiconMicroPLC Object

- ModbusPLC Object

To create STATION objects from the RadioMODEM branch

1. Right-click on **RadioMODEM**.
2. Select **Add STATION Object** from the shortcut menu.
   - A new STATION object is created as a node in the hierarchy tree, and it is named **New_STATION_000** by default.
   - 200 of these objects can be created for each RadioMODEM object.
3. Rename the newly created object as appropriate.
   - The **New_STATION_000 Parameters** configuration view is displayed at the right pane.



Station ID: 1

Redial attempts: 3

Reconnect delay (sec): 10

This configuration view has three configurable elements:

- **Station ID**: Enter the remote radio modem identification string.
  - The minimum number of characters is 1 (one) character.
  - The maximum number of characters is 255 characters.

Note The maximum number of Station nodes supported is 200.

- **Redial attempts**: Specify the maximum number of dialing attempts to be made to connect to the remote modem.
  - The minimum value is 0 (zero).
  - The maximum value is 100.
  - The default value is 3 (three).
- **Reconnect delay (sec)**: Specify the minimum amount of waiting time, in seconds, before a line is to be reconnected (after a failure of line is detected).

- The minimum value is 1 (one).
- The maximum value is 3600.
- The default value is 10 seconds.

PLC Object

The the following Modbus controllers are supported by the ModbusSerial DAServer:

- TSX Quantum PLCs
- TSX Momentum PLCs
- Modicon Micro PLCs
- Generic Modbus (4-Digit, 5-Digit, 6-Digit) PLCs

A total of 247 controllers can be supported by each COM_PORT, each Line, and each Station. They can be configured from either one of the following branches of the DAServer hierarchy or from all three:

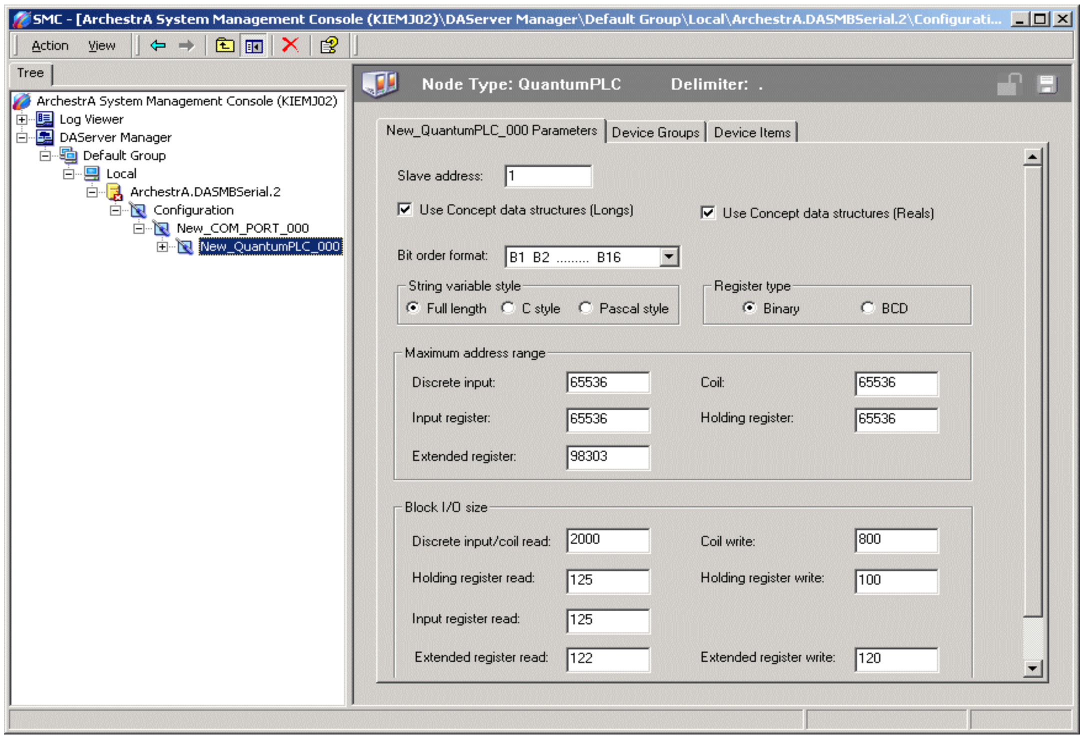
- COM_PORT
- DataMODEM via its LINE leaf
- RadioMODEM via its STATION leaf

QuantumPLC Object

The QuantumPLC object can be created from the COM_PORT, LINE, or STATION branch of the hierarchy.

To add QuantumPLC objects to your ModbusSerial hierarchy

1. Right-click on your **New_COM_PORT_000/New_LINE_000/ New_STATION_000** branch.
2. Select **Add QuantumPLC Object**.
   - A new QuantumPLC object is created as a node in the hierarchy tree, and it is named **New_QuantumPLC_000** by default.
3. Rename as appropriate.
   - The **New_QuantumPLC_000 Parameters** configuration view is displayed.


SMC - [ArchestrA System Management Console (KIEMJ02)\DAServer Manager\Default Group\Local\ArchestrA.DASMBSerial.2\Configurati...
Action
View
Tree
ArchestrA System Management Console (KIEMJ02)
Log Viewer
DAServer Manager
Default Group
Local
ArchestrA.DASMBSerial.2
Configuration
New_COM_PORT_000
New_QuantumPLC_000
New_QuantumPLC_000 Parameters
Device Groups
Device Items

Node Type: QuantumPLC Delimiter: .

Slave address: 1

☑ Use Concept data structures (Longs) ☑ Use Concept data structures (Reals)

Bit order format: B1 B2 .... B16

String variable style: [✓ Full length] C style Pascal style

Register type: [✓ Binary] BCD

Maximum address range

Discrete input:	65536	Coil:	65536
Input register:	65536	Holding register:	65536
Extended register:	98303		

Block I/O size

Discrete input/coil read:	2000	Coil write:	800
Holding register read:	125	Holding register write:	100
Input register read:	125		
Extended register read:	122	Extended register write:	120

This configuration view has eight configurable parameters:

- **Slave address**: Enter the address of the PLC.
  - The maximum value is 247.
  - The minimum value is 1 (one).
  - The default value is 1 (one).
- **Use Concept data structures (Longs)**: Select to read data from the PLC in concept data structure format for Long item types. If checked, the DAServer will process the data in the same register order as the Concept programming software.
  - Checked — selected
  - Not checked — not selected
  - The default is Checked.
- **Use Concept data structures (Reals)**: Select to read data from the PLC in concept data structure format for Real item types. If checked, the DAServer will process the data in the same register order as the Concept programming software.
  - Checked — selected
  - Not checked — not selected
  - The default is Checked.
- **Bit order format**: The format of the bit order entered into the PLC.

  - When the Bit order format is selected as B1 B2 … B16, it means the bit order starts from left to right (the Most Significant Bit = Bit 1 and the Least Significant Bit = Bit 16).
  - When the Bit order format is selected as B16 B15 … B1, it indicates that the bit order starts from right to left (MSB = Bit 16 and LSB = Bit 1).
  - The default is bit order starts from left to right.
- **String variable style**: PLC string-data format. Select the option for the style used by the device to store strings in its registers.
  - Full length (space padded)
  - C style (null terminated)
  - Pascal style (includes length specifier)
  - The default style is Full length.
- **Register type**: Select either Binary or BCD for the register type being used.
  - Binary
  - BCD
  - The default register type is Binary.
- **Maximum address range**: There are five sub-elements in this Maximum address range box. The maximum address ranges can be obtained from the Modicon Concept or Modsoft configuration programs. The PLC will return an error if a register within the configured range is used to read data but does not exist in the PLC. The ModbusSerial DAServer filters out registers outside of this range and logs error messages.
  - **Discrete input**: Enter the maximum number of addressable discrete inputs (read coils) in the PLC.
    - The minimum value is 1 (one).
    - The maximum value is 65536.
    - The default value is 65536.
  - **Coil**: Enter the maximum number of addressable write coils in the PLC.
    - The minimum value is 1 (one).
    - The maximum value is 65536.
    - The default value is 65536.
  - I**nput register**: Enter the maximum number of addressable input registers in the PLC.
    - The minimum value is 1 (one).
    - The maximum value is 65536.
    - The default value is 65536.
  - **Holding register**: Enter the maximum number of addressable holding registers in the PLC.

        - The minimum value is 1 (one).
        - The maximum value is 65536.
        - The default value is 65536.
    - E**xtended register**: Enter the maximum number of addressable extended registers in the PLC.
        - The minimum value is 1 (one).
        - The maximum value is 98303.
        - The default value is 98303.
- **Block I/O size**: The Block I/O Size box contains seven sub-elements. The DAServer uses the Block I/O size to maximize data throughput. The ModbusSerial DAServer uses a 256-byte buffer to read or write data to the PLC. The maximum value is the maximum number of registers that can be read or written from/to the PLC in one command.
    - **Discrete input/coil read**: Enter the maximum number of discrete inputs or coils to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 2000.
        - The default value is 2000.
    - **Coil write**: Enter the maximum number of coils to write at one time.
        - The minimum value is 1 (one).
        - The maximum value is 800.
        - The default value is 800.
    - **Holding register read**: Enter the maximum number of holding registers to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 125.
        - The default value is 125.
    - **Holding register write**: Enter the maximum number of holding registers to write at one time.
        - The minimum value is 1 (one).
        - The maximum value is 100.
        - The default value is 100.
    - **Input register read**: Enter the maximum number of input registers to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 125.
        - The default value is 125.
    - **Extended register read**: Enter the maximum number of extended registers to read at one time.

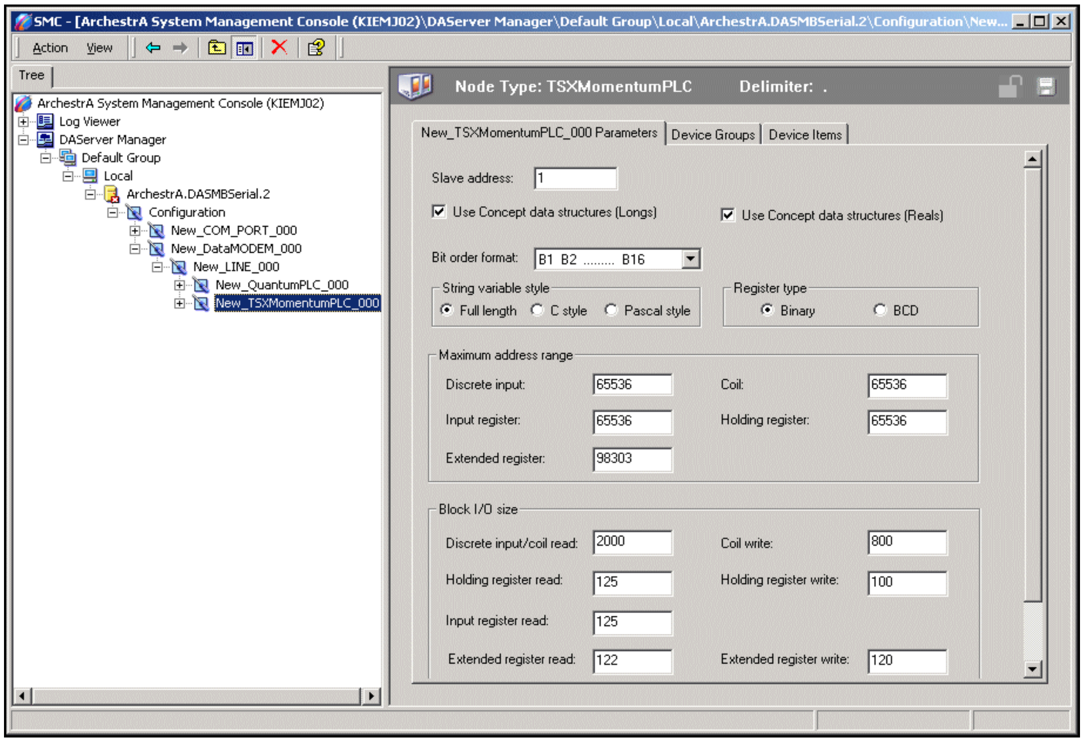
    - The minimum value is 1 (one).
    - The maximum value is 122
    - The default value is 122.
- **Extended register write**: Enter the maximum number of extended registers to write at one time.
    - The minimum value is 1 (one).
    - The maximum value is 120.
    - The default value is 120.

TSXMomentumPLC Object

From the COM_PORT/LINE/STATION branch of the DAServer hierarchy, the TSXMomentumPLC object can be created.

To add TSXMomentumPLC objects to your ModbusSerial hierarchy

1. Right-click on your **New_COM_PORT_000/New_LINE_000/ New_STATION_000** branch.
2. Select **Add TSXMomentumPLC Object**.
    - A new TSXMomentumPLC object is created as a node in the hierarchy tree.
    - It is named **New_TSXMomentumPLC_000** by default.
3. Rename as appropriate.
    - The **New_TSXMomentumPLC_000 Parameters** configuration view is displayed.



SMC - [ArchestrA System Management Console (KIEMJ02)\DAServer Manager\Default Group\Local\ArchestrA.DASMBSerial.2\Configuration\New...

ArchestrA System Management Console (KIEMJ02)
- Log Viewer
- DAServer Manager
  - Default Group
    - Local
      - ArchestrA.DASMBSerial.2
        - Configuration
          - New_COM_PORT_000
          - New_DataMODEM_000
            - New_LINE_000
              - New_QuantumPLC_000
              - New_TSXMomentumPLC_000

Node Type: TSXMomentumPLC Delimiter: .

New_TSXMomentumPLC_000 Parameters | Device Groups | Device Items

Slave address: 1

[✓] Use Concept data structures (Longs) [✓] Use Concept data structures (Reals)

Bit order format: B1 B2 .... B16

String variable style: [✓ Full length] C style Pascal style

Register type: [✓ Binary] BCD

Maximum address range

Discrete input:	65536	Coil:	65536
Input register:	65536	Holding register:	65536
Extended register:	98303		

Block I/O size

Discrete input/coil read:	2000	Coil write:	800
Holding register read:	125	Holding register write:	100
Input register read:	125		
Extended register read:	122	Extended register write:	120

This configuration view has eight elements that are configurable:

- **Slave address**: Enter the address of the PLC.
  - The maximum value is 247.
  - The minimum value is 1 (one).
  - The default value is 1 (one).
- **Use Concept data structures (Longs)**: Select to read data from the PLC in concept data structure format for Long item types. If checked, the DAServer will process the data in the same register order as the Concept programming software.
  - Checked — selected
  - Not checked — not selected
  - The default is Checked.
- **Use Concept data structures (Reals)**: Select to read data from the PLC in concept data structure format for Real item types. If checked, the DAServer will process the data in the same register order as the Concept programming software.
  - Checked — selected
  - Not checked — not selected
  - The default is Checked.
- **Bit order format**: The format of the bit order entered into the PLC.

  - When the Bit order format is selected as B1 B2 … B16, it means the bit order starts from left to right (the Most Significant Bit = Bit 1 and the Least Significant Bit = Bit 16).
  - When the Bit order format is selected as B16 B15 … B1, it indicates that the bit order starts from right to left (MSB = Bit 16 and LSB = Bit 1).
  - The default is bit order starts from left to right.
- **String variable style**: PLC string-data format. Select the option for the style used by the device to store strings in its registers.
  - Full length (space padded)
  - C style (null terminated)
  - Pascal style (includes length specifier)
  - The default style is Full length.
- **Register type**: Select either Binary or BCD for the register type being used.
  - Binary
  - BCD
  - The default register type is Binary.
- **Maximum address range**: There are five sub-elements in this Maximum address range box. The maximum address ranges can be obtained from the Modicon Concept or Modsoft configuration programs. The PLC will return an error if a register outside of this range is used to read data. The ModbusSerial DAServer filters out registers outside of this range and logs error messages.
  - **Discrete input**: Enter the maximum number of addressable discrete inputs (read coils) in the PLC.
    - The minimum value is 1 (one).
    - The maximum value is 65536.
    - The default value is 65536.
  - **Coil**: Enter the maximum number of addressable write coils in the PLC.
    - The minimum value is 1 (one).
    - The maximum value is 65536.
    - The default value is 65536.
  - I**nput register**: Enter the maximum number of addressable input registers in the PLC.
    - The minimum value is 1 (one).
    - The maximum value is 65536.
    - The default value is 65536.
  - **Holding register**: Enter the maximum number of addressable holding registers in the PLC.

        - The minimum value is 1 (one).
        - The maximum value is 65536.
        - The default value is 65536.
    - E**xtended register**: Enter the maximum number of addressable extended registers in the PLC.
        - The minimum value is 1 (one).
        - The maximum value is 98303.
        - The default value is 98303.
- **Block I/O size**: The Block I/O Size box contains seven sub-elements. The DAServer uses the Block I/O size to maximize data throughput. The ModbusSerial DAServer uses a 256-byte buffer to read or write data to the PLC. The maximum value is the maximum number of registers that can be read or written from/to the PLC in one command.
    - **Discrete input/coil read**: Enter the maximum number of discrete inputs or coils to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 2000.
        - The default value is 2000.
    - **Coil write**: Enter the maximum number of coils to write at one time.
        - The minimum value is 1 (one).
        - The maximum value is 800.
        - The default value is 800.
    - **Holding register read**: Enter the maximum number of holding registers to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 125.
        - The default value is 125.
    - **Holding register write**: Enter the maximum number of holding registers to write at one time.
        - The minimum value is 1 (one).
        - The maximum value is 100.
        - The default value is 100.
    - **Input register read**: Enter the maximum number of input registers to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 125.
        - The default value is 125.
    - **Extended register read**: Enter the maximum number of extended registers to read at one time.

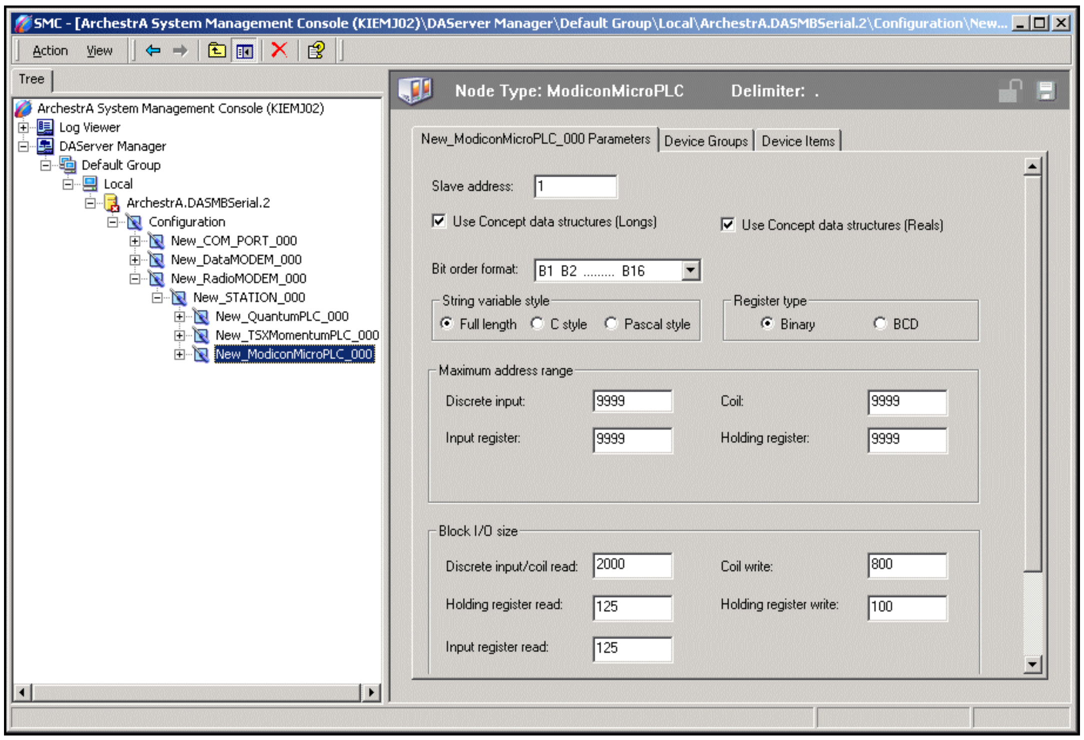
    - The minimum value is 1 (one).
    - The maximum value is 122.
    - The default value is 122.
- **Extended register write**: Enter the maximum number of extended registers to write at one time.
    - The minimum value is 1 (one).
    - The maximum value is 120.
    - The default value is 120.

ModiconMicroPLC Object

The ModiconMicroPLC object can be created from the COM_PORT/LINE/STATION branch of the DAServer hierarchy.

To add ModiconMicroPLC objects to your ModbusSerial hierarchy

1. Right-click on your **New_COM_PORT_000/New_LINE_000/New_STATION_000** branch.
2. Select **Add ModiconMicroPLC Object**.
    - A new ModiconMicroPLC object is created as a node in the hierarchy tree.
    - It is named **New_ModiconMicroPLC_000** by default.
3. Rename as appropriate.
    - The **New_ModiconMicroPLC_000 Parameters** configuration view is displayed.



Node Type: ModiconMicroPLC Delimiter: .

Slave address: 1

[✓] Use Concept data structures (Longs) [✓] Use Concept data structures (Reals)

Bit order format: B1 B2 .... B16

String variable style: [✓ Full length] C style Pascal style

Register type: [✓ Binary] BCD

Maximum address range

Discrete input:	9999	Coil:	9999
Input register:	9999	Holding register:	9999

Block I/O size

Discrete input/coil read:	2000	Coil write:	800
Holding register read:	125	Holding register write:	100
Input register read:	125		

This configuration view has eight configurable elements:

- **Slave address**: Enter the address of the PLC.
  - The maximum value is 247.
  - The minimum value is 1 (one).
  - The default value is 1 (one).
- **Use Concept data structures (Longs)**: Select to read data from the PLC in concept data structure format for Long item types. If checked, the DAServer will process the data in the same register order as the Concept programming software.
  - Checked — selected
  - Not checked — not selected
  - The default is Checked.
- **Use Concept data structures (Reals)**: Select to read data from the PLC in concept data structure format for Real item types. If checked, the DAServer will process the data in the same register order as the Concept programming software.
  - Checked — selected
  - Not checked — not selected
  - The default is Checked.
- **Bit order format**: The format of the bit order entered into the PLC.

    - When the Bit order format is selected as B1 B2 … B16, it means the bit order starts from left to right (the Most Significant Bit = Bit 1 and the Least Significant Bit = Bit 16).
    - When the Bit order format is selected as B16 B15 … B1, it indicates that the bit order starts from right to left (MSB = Bit 16 and LSB = Bit 1).
    - The default is bit order starts from left to right.
- **String variable style**: PLC string-data format. Select the option for the style used by the device to store strings in its registers.
    - Full length (space padded)
    - C style (null terminated)
    - Pascal style (includes length specifier)
    - The default style is Full length.
- **Register type**: Select either Binary or BCD for the register type being used.
    - Binary
    - BCD
    - The default register type is Binary.
- **Maximum address range**: There are five sub-elements in this Maximum address range box. The maximum address ranges can be obtained from the Modicon Concept or Modsoft configuration programs. The PLC will return an error if a register outside of this range is used to read data. The ModbusSerial DAServer filters out registers outside of this range and logs error messages.
    - **Discrete input**: Enter the maximum number of addressable discrete inputs (read coils) in the PLC.
        - The minimum value is 1 (one).
        - The maximum value is 9999.
        - The default value is 9999.
    - **Coil**: Enter the maximum number of addressable write coils in the PLC.
        - The minimum value is 1 (one).
        - The maximum value is 9999.
        - The default value is 9999.
    - **Input register**: Enter the maximum number of addressable input registers in the PLC.
        - The minimum value is 1 (one).
        - The maximum value is 9999.
        - The default value is 9999.
    - **Holding register**: Enter the maximum number of addressable holding registers in the PLC.

    - The minimum value is 1 (one).
    - The maximum value is 9999.
    - The default value is 9999.
- **Block I/O size**: The Block I/O Size box contains seven sub-elements. The DAServer uses the Block I/O size to maximize data throughput. The ModbusSerial DAServer uses a 256-byte buffer to read or write data to the PLC. The maximum value is the maximum number of registers that can be read or written from/to the PLC in one command.
  - **Discrete input/coil read**: Enter the maximum number of discrete inputs or coils to read at one time.
    - The minimum value is 1 (one).
    - The maximum value is 2000.
    - The default value is 2000.
  - **Coil write**: Enter the maximum number of coils to write at one time.
    - The minimum value is 1 (one).
    - The maximum value is 800.
    - The default value is 800.
  - **Holding register read**: Enter the maximum number of holding registers to read at one time.
    - The minimum value is 1 (one).
    - The maximum value is 125.
    - The default value is 125.
  - **Holding register write**: Enter the maximum number of holding registers to write at one time.
    - The minimum value is 1 (one).
    - The maximum value is 100.
    - The default value is 100.
  - **Input register read**: Enter the maximum number of input registers to read at one time.
    - The minimum value is 1 (one).
    - The maximum value is 125.
    - The default value is 125.

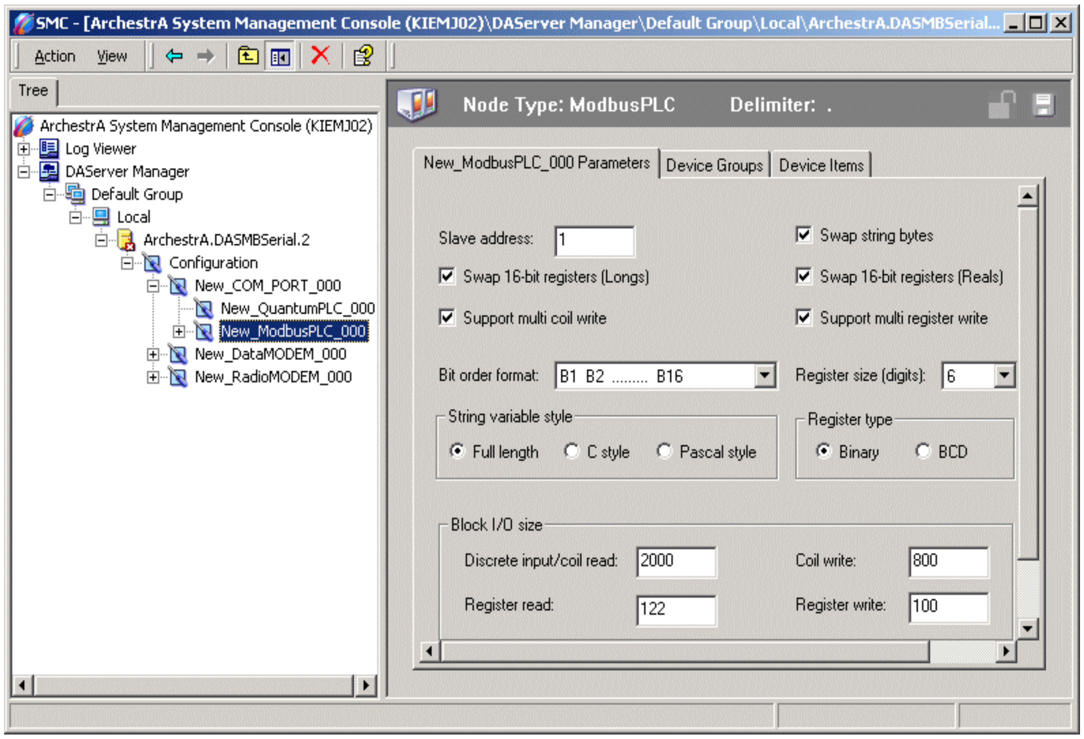
ModbusPLC Object

From the COM_PORT/LINE/STATION branch of the DAServer hierarchy, the ModbusPLC object can be created.

To add ModbusPLC objects to your ModbusSerial hierarchy

1. Right-click on your **New_COM_PORT_000/New_LINE_000/ New_STATION_000** branch.

2. Select **Add ModbusPLC Object**.
   - A new ModbusPLC object is created as a node in the hierarchy tree.
   - It is named **New_ModbusPLC_000** by default.
3. Rename as appropriate.
   - The **New_ModbusPLC_000 Parameters** configuration view is displayed.



Slave address: 1

☑ Swap string bytes

☑ Swap 16-bit registers (Longs)

☑ Swap 16-bit registers (Reals)

☑ Support multi coil write

☑ Support multi register write

Bit order format: B1 B2 .... B16

Register size (digits): 6

String variable style: [✓ Full length] C style Pascal style

Register type: [✓ Binary] BCD

Block I/O size

Discrete input/coil read: 2000

Coil write: 800

Register read: 122

Register write: 100

This configuration view has 11 configurable elements:

- **Slave address**: Enter the address of the PLC.
  - The maximum value is 247.
  - The minimum value is 1 (one).
  - The default value is 1 (one).
- **Swap string bytes**: Select to swap string bytes on data read and data poked.
  - Checked — selected
  - Not checked — not selected
  - The default is Checked.

- **Swap 16-bit registers (Longs)**: Select to read data from the PLC where the two 16-bit registers making up the Long value are swapped or used in the order read. If checked, the DAServer will swap the registers. If unchecked, the DAServer will use the registers in the order read.
  For example:
  If you request data for "408001 L," the contents of registers 408001 and 408002 are displayed as one value, a Long value.
  If the value of the PLC register 408001 is 0722 and the value of the PLC register 408002 is 18838, the value displayed is 1234567890 if the **Swap 16-bit registers (Longs)** box is checked. This is how the Schneider Concept programming software displays it.
  If the **Swap 16-bit registers (Longs)** box is not checked, the value displayed is 47335830.
  - Checked – selected
  - Not checked – not selected
  - The default is Checked.
- **Swap 16-bit registers (Reals)**: Select to read data from the PLC where the two 16-bit registers making up the Real value are swapped or used in the order read. If checked, the DAServer will swap the registers. If unchecked, the DAServer will use the registers in the order read.
  For example:
  If you request data for "408003 F," the contents of registers 408003 and 408004 are displayed as one value, a Real value.
  If the value of the PLC register 408003 is 16056 and the value of the PLC register 408004 is 17530, the value displayed is 1000.98 if the **Swap 16-bit registers (Reals)** box is checked. This is how the Schneider Concept programming software displays it.
  If the **Swap 16-bit registers (Reals)** box is not checked, the value displayed is 0.3598974.
  - Checked – selected
  - Not checked – not selected
  - The default is Checked.
- **Support multi coil write**: Select for the PLC to write to multiple coils in one message with the Modbus protocol function code 15 (0x0F). If not selected, the PLC will write to a single coil in one message with the Modbus protocol function code 5 (0x05).
  - Checked – selected
  - Not checked – not selected
  - The default is Checked.
- **Support multi register write**: Select for the PLC to write to multiple registers in one message with the Modbus protocol function code 16 (0x10). If not selected, the PLC will write to a single register in one message with the Modbus protocol function code 6 (0x06).
  - Checked – selected
  - Not checked – not selected
  - The default is Checked.

Note When this parameter is not selected, it implies that the PLC does not support multiple register writes and the server will only write single registers to the PLC. As one register is 16-bit long, the 32-bit integer value will be downcast to 16-bit integer before writing to the PLC. Since the value is larger than 65535 for unsigned integer or 32767 for signed integer, the write operation will fail.
Additionally when this parameter is not selected, item name suffix that has the 32-bit indication will also fail.
For example, items such as 4xxxxx L, 4xxxxx I, 4xxxxx U, 4xxxxx F, 4xxxxx-4xxxxx M, 5 HRL, 5 HRF, 5 PV, 5 HRU, and 4xxxxx-4xxxxx cannot be written. When you try to write to the PLC with this parameter not selected, the following error message will be logged to the logger, "Cannot write to multiple register item: 4xxxxx L on Node: COMPort.GenPLC. The PLC configurable parameter Support Multiple Register Write is not checked."

- **Bit order format**: The format of the bit order entered into the PLC.
  - When the Bit order format is selected as B1 B2 … B16, it means the bit order starts from left to right (the Most Significant Bit = Bit 1 and the Least Significant Bit = Bit 16).
  - When the Bit order format is selected as B16 B15 … B1, it indicates that the bit order starts from right to left (MSB = Bit 16 and LSB = Bit 1).
  - The default is bit order starts from left to right.
- **Register size (digits)**: Select the correct register size for addressing the PLC.
  - 4-digit is used for addressing the Generic Modbus 4-Digit PLCs.
  - 5-digit applies to the Generic Modbus 5-Digit PLCs.
  - 6-digit is used for addressing the Generic Modbus 6-Digit PLCs.
  - The default value is 6, for the Generic Modbus 6-Digit PLCs.

Note The selection for the Register size determines the maximum address range. They are changeable as in other supported PLCs listed in Appendix A, Supported DASModbusSerial Hardware and Firmware.
For 4-digit, the value is clamped to 999.
For 5-digit, the maximum value is 9999.
For 6-digit, the maximum value is 65536.

- **String variable style**: PLC string-data format. Select the option for the style used by the device to store strings in its registers.
  - Full length (space padded)
  - C style (null terminated)
  - Pascal style (includes length specifier)
  - The default style is Full length.
- **Register type**: Select either Binary or BCD for the register type being used.

    - Binary
    - BCD
    - The default register type is Binary.
- **Block I/O size**: The Block I/O Size box contains seven sub-elements. The DAServer uses the Block I/O size to maximize data throughput. The ModbusSerial DAServer uses a 256-byte buffer to read or write data to the PLC. The maximum value is the maximum number of registers that can be read or written from/to the PLC in one command.
    - **Discrete input/coil read**: Enter the maximum number of discrete inputs or coils to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 2000.
        - The default value is 2000.
    - **Coil write**: Enter the maximum number of coils to write at one time.
        - The minimum value is 1 (one).
        - The maximum value is 800.
        - The default value is 800.
    - **Register read**: Enter the maximum number of extended registers to read at one time.
        - The minimum value is 1 (one).
        - The maximum value is 122.
        - The default value is 122.
    - **Register write**: Enter the maximum number of holding registers to write at one time.
        - The minimum value is 1 (one).
        - The maximum value is 100.
        - The default value is 100.

The logical endpoint for each branch of the ModbusSerial hierarchy tree is a hardware device (PLC).

Note The default name created from adding a hierarchy object is in the format of **New_ObjectName_###**, where ObjectName is the name of the object type and ### is a numeric value starting from "000" enumerated sequentially per hierarchy object. The hierarchy object name is up to 32 characters long. The link name for the OPC items is constructed by assembling the respective object names of the nodes along the hierarchy tree in the logical order, starting from this DAServer's root down to the leaf. Therefore, the link name is always unique for the DAServer.

Note In order to use the DAServer, you must activate it. See the DAServer Manager documentation for information about how to activate and deactivate the DAServer.

String-Data Handling

The ModbusSerial DAServer can process three different configurable string variable styles:

- Full Length
- C Style
- Pascal

Depending on what string style the PLC is using, you can configure the server using the user interface in the PLC configuration view in order to use the appropriate string variable style.

Full Length Style

If strings are read using the Full Length Style, each string always uses all of the registers allocated. The PLC string is stored in the server string as is.

If a string is written and the string is shorter than the allocation of registers, it is padded with ASCII space characters (hex 20).

For example:

If the string "Wonderware" is stored in the string item "400001-400010 m," registers 400001 through 400005 contain the string "Wonderware" and registers 400006 through 400010 contain spaces.
If the string "Wonderware" is stored in the string item "400001-400005 m," registers 400001 through 400005 contain the string "Wonderware" and no spaces are stored.
If the string "Wonderwareee" is stored in the string item "400001-400005 m," registers 400001 through 400005 contain the string "Wonderware" and no spaces are stored.
A message is placed in the logger indicating that the string was truncated.

C Style

If a string is read using the C Style, the string always uses all of the registers allocated. The PLC string is stored in the server string as is, except that the last character contained in the last register of the string is replaced with a null character (hex 00).

If a string is written and the string is shorter than the allocation of registers, it is padded with ASCII null characters (hex 00).

For example:

If the string "Wonderware" is stored in the string item "400001-400010 m," registers 400001 through 400005 contain the string "Wonderware" and registers 400006 through 400010 contain nulls.

If the string "Wonderware" is stored in the string item "400001-400005 m," registers 400001 through 400005 contain the truncated string "Wonderwar0" with a null character replacing the last character "e."
A message is placed in the logger indicating that the string was truncated.

Pascal Style

If strings are read using the Pascal Style, each string uses a length obtained from the first byte of the string to store data in the server. The PLC string is stored in the server string as is, up to the length obtained from the first byte of the string. If the length is greater than the number of registers defined in the item, then the PLC string is stored in the server string as is, up to the maximum number of registers.

The first byte written of any string of this style contains the character count. The string being written starts in the second byte. If a string is written and the string plus the character count are shorter than the allocation of registers, it is padded with ASCII null characters (hex 00).

For example:

If the string "Wonderware" is stored in the string item "400001-400010 m," registers 400001 through 400006 contain the string "(10)Wonderware0" and registers 400007 through 400010 contain nulls. The (10) in the string implies one byte containing the character count.
If the string "Wonderware" is stored in the string item "400001-400005 m," registers 400001 through 400005 contain the truncated string "(9)Wonderwar."
A message is placed in the logger indicating that the string was truncated.

Message Optimization

The MBSerial DAServer uses Multi read and write commands to optimize PLC read/write messages. The MBSerial DAServer optimizes the reading and writing of data by grouping points that are in consecutive registers. The Block I/O sizes parameters control the buffer size. The default is to maximize the buffer size.

Note The number of bytes for the query and response buffers must not exceed the Modbus maximum buffer size of 256 bytes.

Configuring Device Group and Device Item Definitions

The **Device Groups** tab in the DAServer Manager user interface is used for creating new, modifying, or deleting device group definitions for an object.

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

To create new, modify, or delete device item definitions for PLC items (addresses), and define aliases to actual PLC items you use the **Device Items** tab.

Device Group Definitions

The **Device Groups** dialog box, which is displayed by clicking the **Device Groups** tab in the **New_<Name>PLC_000 Parameters** configuration view, is used to perform the following activities:

- Adding, defining, and deleting device groups.

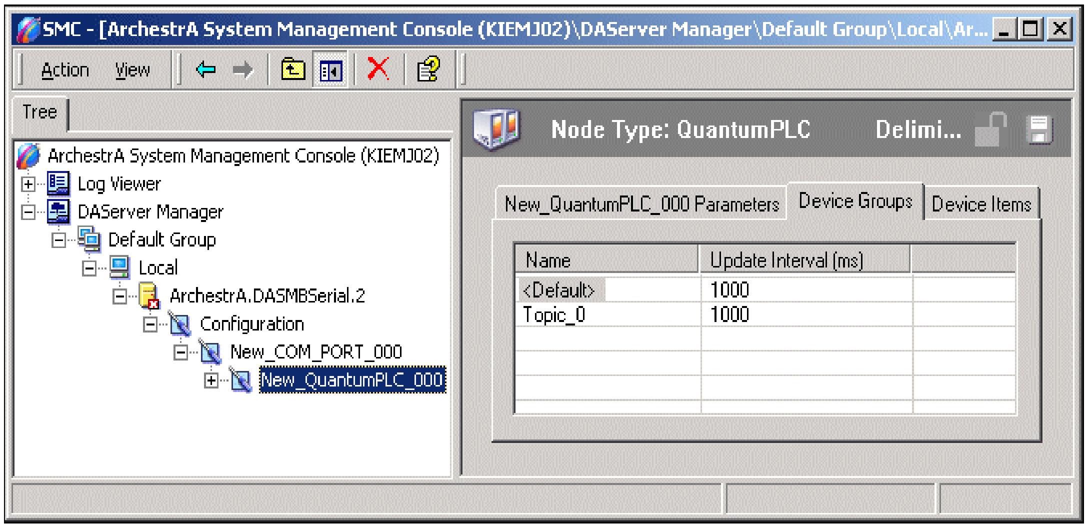
  Note When you add a new device group, enter a unique name.

- Configuring default update intervals.
- Editing update intervals for the objects.

Note When you select another part of the DAServer 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** dialog box.
2. Select the **Add** command from the shortcut menu.
   - When you add a new device group, enter a unique name (up to 32 characters long).



Name	Update Interval (ms)
<Default>	1000
Topic_0	1000

To make changes on device groups' names

Make changes on a device group's name for an object as follows:

- In the **Name** column, double-click on the device group's name to be modified and make the change.

To delete device groups

Deleting a device group from the list can be performed as follows:

1. Right-click on the device group to be deleted.
2. Select the **Delete** command from the shortcut menu.

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

To configure default update intervals

1. To configure a default update interval for the object, right-click in the **Device Groups** dialog box.
2. Select **Config Default Update Interval** from the shortcut menu.

To make changes on update intervals

- Double-click on the device group's value to be modified in the **Update Interval** column and make the change.
  - Update Interval is the frequency (in milliseconds) that the DAServer 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 ModbusSerial DAServer tree hierarchy, you are prompted to save the modifications to the configuration set.

Each configuration view associated with nodes/objects in the DAServer hierarchy tree has a common feature, the **Save** button.

1. When you modify any parameters in the **Device Groups** dialog box, click **Save** to save and implement the new modifications.
   - If you do not click **Save**, the configuration is reset to its original condition (since the last save).
2. After all modifications, you must save when prompted for the new data to be saved to the configuration set.

Device Item Definitions

The **Device Items** tab in the **New_<Name>PLC_000 Parameters** configuration view is used to define aliases to actual PLC items (addresses). The **Device Items** dialog box is the place where the following activities are performed:

- Creating new device item definitions for PLC items.
- Modifying the existing device items.
- Deleting device items.

- Archiving the created list of device items to a .csv file, an ASCII file of the list with values separated by commas.
- Bringing a .csv file into the **Device Items** tabbed page.

Each device item definition should contain a unique name for the PLC associated with it.

The **Device Items** dialog box has the following two columns:

- **Name**: This column defines the alias names to actual PLC items (addresses).
- **Item Reference**: The actual PLC item names, linked to the created aliases, are defined in this column.

For example:

For Modicon holding register 400001, the following entries can be created.

Name	Item Reference
Holding1	400001
Holding10F	400010 F

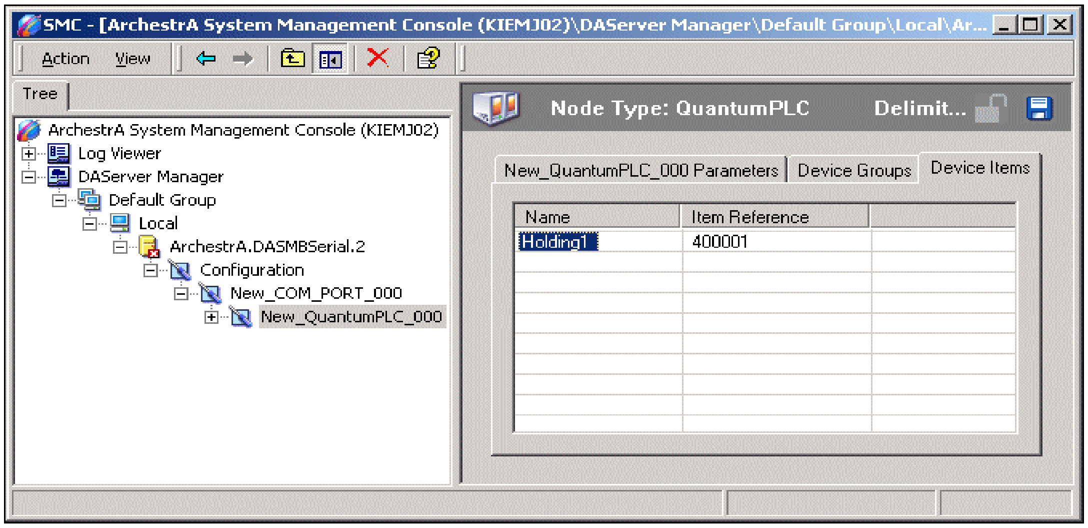
Note When you create or add a new device item, a unique name needs to be entered for it.

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

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

To create or add device items

1. Right-click in the **Device Items** dialog box.
2. Select the **Add** command from the shortcut menu.
   - A device item is created in the **Name** column, and it is numerically named by default.
     For example, Item_0, Item_1, and so on.
3. Change the default name by double-clicking on it and entering the new name.
   - Enter a unique name for the new device item.
     For example, "Holding1."



Name	Item Reference
Holding1	400001

To add item references

Item references for each of the device items that have been created can be added as follows:

1. In the **Item Reference** column, double-click on the area in the same horizontal line as the selected device item.
2. Type in the actual PLC item name in the frame that appears.
   - For example, "400001."
3. Click anywhere in the dialog box or press the ENTER key to have the change take effect.

Note System items are not valid item reference, but DAServer-specific system items are ok.

To rename a device item from the list

1. Right-click on the device item to be renamed.
2. Select the **Rename** command from the shortcut menu and enter the new device item name.
3. Click anywhere in the dialog box or press the ENTER key to apply the change.

To delete a device item from the list

1. Right-click on the device item to be deleted.
2. Select the **Delete** command from the shortcut menu.
   - The device item and its corresponding actual PLC item name will be deleted from the dialog box.

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

To clear all device items

1. Right-click anywhere in the **Device Items** dialog box.
2. Select the **Clear All** command from the shortcut menu.
   - All the device items listed in the dialog box, including their corresponding actual PLC item names, will be deleted.

The **Export** and **Import** features on the shortcut menu of the **Device Items** dialog box enable you to export and import the DAServer device item data to and from a CSV file, after the configuration of the Device Items has been completed. These features provide you with the following capabilities:

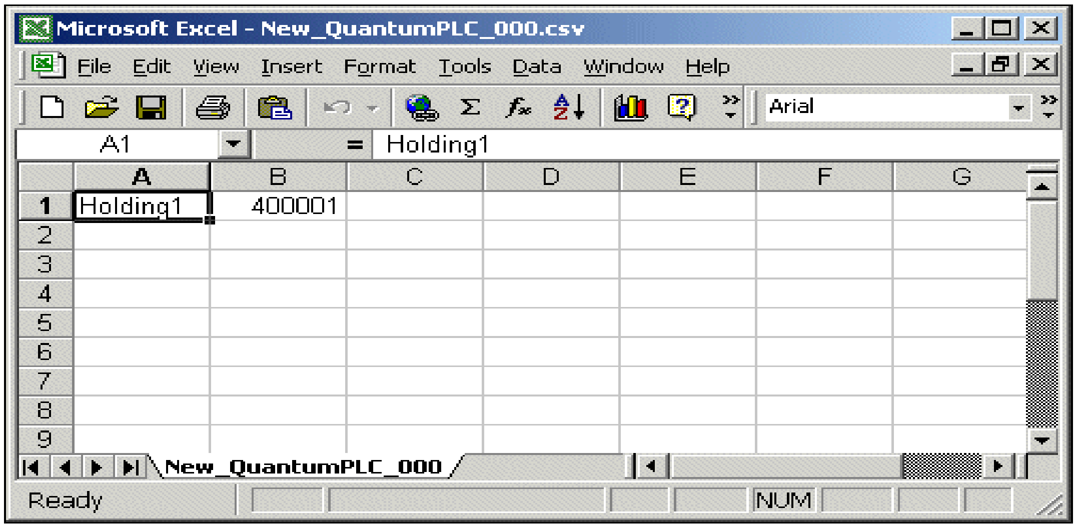
- Archive lists of device items.
- Import an archived list of device items into the **Device Items** dialog box when you need to utilize or reconfigure any of the device items on the archived list.
- Perform an off-line, large-scale edit on the item data configured for a device item list.
- Import what has been edited back into the Device Items configuration.

To export device items

1. Right-click anywhere in the **Device Items** dialog box.
2. Select the **Export** command from the shortcut menu.
   - The standard **Save As** dialog box appears.
   - The file name has defaulted into "PLC Hierarchyname.csv," within the current-system-configured default directory.
3. Accept the defaults to save the file.
   - The file is saved as New_<PLC Name>_000.csv.
   - It is editable in Microsoft Excel.

However, if you prefer to save the list someplace else and rename it, perform the following steps after step 2.

4. Select the folder into which the list is to be saved.
5. Name the list to be archived.
6. Click the **Save** button.
   - The whole list will be saved as a .csv file in Excel.

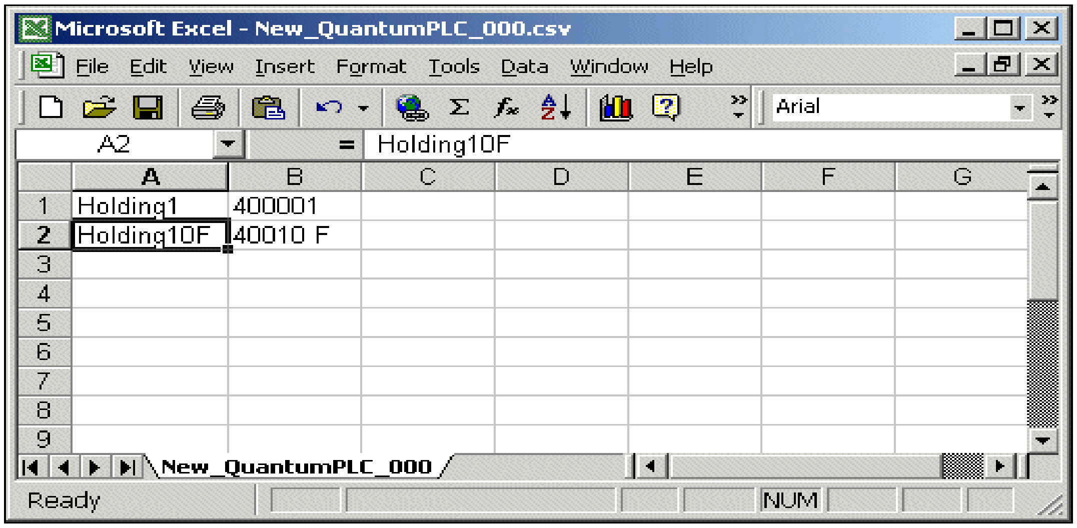


Microsoft Excel - New_QuantumPLC_000.csv

A1 = Holding1

	A	B
1	Holding1	400001

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



Microsoft Excel - New_QuantumPLC_000.csv

A2 = Holding10F

	A	B
1	Holding1	400001
2	Holding10F	40010 F

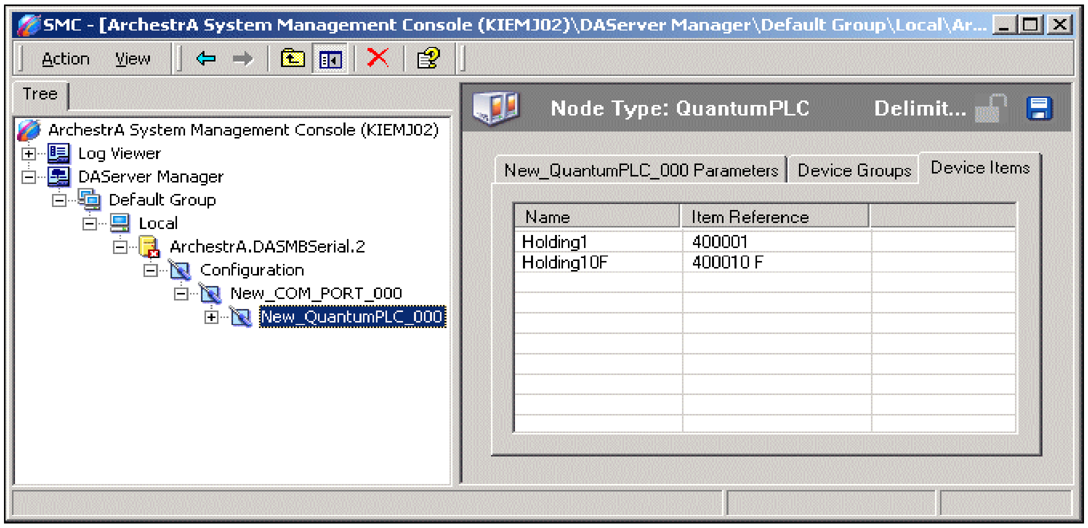
To import device items

1. To import the list, right-click anywhere in the **Device Items** dialog box.
2. Select the **Import** command from the shortcut menu.
3. Select the archived list (.csv file) to be imported from the folder in which it is saved.
4. Click the **Open** button.
   - The whole list will be brought into the **Device Items** dialog box.

Note When the list to be imported contains duplicate names as found in the current list but the Item References are different, a dialog box will appear to prompt you to make a selection.

To import device item data that has been edited off-line

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



Name	Item Reference
Holding1	400001
Holding10F	400010 F

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

Scan-Based Message Handling

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

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

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

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

Archiving Configuration Sets

After your DAServer has been configured, you can archive that specific configuration. You can archive more than one configuration set, and subsequently choose different configurations for different purposes.

To archive configuration sets

1. In the DAServer Manager, right-click on the **Configuration** node in the hierarchy below your DAServer.
2. Select **Archive Configuration Set** from the shortcut menu.
3. In the **Archive Configuration Set** configuration view, provide a Configuration Set Name.
4. Click **Archive**.
    - All current configuration values are saved to the archived set.

Once you have archived at least one configuration set, you can select it for use.

To use different configuration sets from the current one

1. Make sure the DAServer is not running.
2. In the DAServer Manager, right-click the **Configuration** node in the hierarchy below your DAServer.
3. Select **Use Another Configuration Set** from the shortcut menu and click on a configuration set in the sub-menu.
    - All parameters in the DAServer configuration hierarchy change to the chosen configuration set.

Hot Configuration

If a parameter value change takes effect right away while the DAServer is running, the parameter is a hot-configurable parameter. Certain parameters in the ModbusSerial DAServer are hot-configurable. Incorporated in the DAServer are the following hot-configuration functionalities:

- Modifying Global Configuration parameters.
- Adding, deleting, or modifying device nodes (without affecting any other device nodes, excluding the children of the modified device nodes).
- Adding, deleting, or modifying device groups in the **Name** and **Update Interval** columns in the **Device Groups** tab.
- Adding, deleting, or modifying Name and Item Reference in the **Device Items** tab.

Limited support is provided for the hot configuration for the server-specific configuration parameters in this release. You can modify server-specific parameters while the server is active. However, to have those changes take effect, you have to restart the DAServer.

The following parameters are hot configurable. They can be modified online and changes will take effect without restarting the DAServer.

- Reply timeout
- String variable style
- Register type

Note If changes are made to server-specific parameters while the server is active, the DAServer will issue a warning message to the logger.

C H A P T E R 3

Item Names

The Wonderware ModbusSerial DAServer supports a variety of data types for the Modicon controllers. It also supports item names that follow the conventions described for the Modicon PLCs, the TSX Quantum (6-Digit), TSX Momentum (6-Digit), ModiconMicro (6-Digit), and Generic Modbus (4-Digit, 5-Digit, and 6-Digit).

Contents

Data and Register Types

When a client sends a read/write request to the ModbusSerial DAServer, the DAServer needs to know its data type and size. In order to determine this information, the ModbusSerial DAServer parses the item name to get the register number, data type, and size. The DAServer builds messages with items sorted by PLC, register type, register number, and topic name, allowing the DAServer to optimize the number of registers that can be read in one-scan command.

The following table contains the types of data supported by the ModbusSerial DAServer for the Modicon controllers, the TSX Quantum (6-Digit), TSX Momentum (6-Digit), ModiconMicro, and Generic Modbus (4-Digit, 5-Digit, and 6-Digit).

Data Type supported for TSX Quantum/TSX Momentum/Modicon Micro/Generic Modbus (4-, 5-, and 6-Digit) PLCs	Range
Discrete (bit)	0 (zero), 1 (one)
Signed Short Integer (signed 16-bit integer)	-32678 to 32767
Unsigned Short Integer (unsigned 16-bit integer)	0 (zero) to 65535

Data Type supported for TSX Quantum/TSX Momentum/Modicon Micro/Generic Modbus (4-, 5-, and 6-Digit) PLCs	Range
Signed Long Integer (signed 32-bit integer)	-2147483648 to 2147483647
Unsigned Long Integer (unsigned 32-bit integer)	0 (zero) to 4294967295
REAL (32-bit float)	32-bit IEEE
String	250 characters
Elements of arrays of BOOLs, SINTs, INTs, DINTs, and REALs	

Note The unsigned integer data type "U" has a valid range of 0 (zero) to 2147483647 when accessed through DDE/SL client; for OPC clients the valid range is from 0 to 4294967295.

Note System-defined types will not be supported as block reads. A read on any of these data types will return only the first element.

Note This ModbusSerial DAServer supports the reading/writing of single elements of arrays only. Reads/writes of complete arrays are not supported.

The following table lists the PLC register types, the data types contained in the registers, and what each is processed as.

PLC Register Type	Data Type Contained in the Register	Processed As
Discrete Output (Coil)	Discrete	Real Time Data
Discrete Input	Discrete	Real Time Data
Holding Register	Discrete, Integer, Float, and String	Real Time Data
Input Register	Discrete, Integer, Float, and String	Real Time Data
Extended Register	Discrete, Integer, Float, and String	Real Time Data
FIFO Queue	Register addresses	Diagnostic Data
Exception Status	Status bits	Diagnostic Data
Comm Event Counter	Status words and counters	Diagnostic Data
Comm Event Log	Status bytes, words, and counters	Diagnostic Data

PLC Register Type	Data Type Contained in the Register	Processed As
Communication Status	Status bytes	Diagnostic Data (Function code 8)
Controller Information	Status bytes, words, counters, and slave IDs	Initialization Data

Modbus Item Naming

The Modbus-family controllers store data in the Registers. The ModbusSerial DAServer supports item names that are consistent with the point naming conventions used by the Modicon PLCs.

The following item naming conventions are described in this section:

- Register-Number Item Names
- Item Names Using the Modicon PLC Register Addresses
- Absolute-Notation Item Names
- Modulo-10000-Point Item Names
- Modulo-10000 Items, BCD Register Type, and Concept Data Structures

Note The tag-name length with SuiteLink is limited to 32 characters.

Register-Number Item Names

The register number, which is consistent with the point naming convention used by Modicon PLCs, is used as the item name. The Modbus-family PLC address ranges, supported by the DAServer, for the TSX Quantum, TSX Momentum, Modicon Micro, and Generic Modbus 4-Digit, 5-Digit, and 6-Digit PLCs are shown in the following table.

The ModbusSerial DAServer will adhere to this address range for native mode.

Register Type	TSX Quantum/ TSX Momentum/Generic Modbus 6-Digit	Modicon Micro	Generic Modbus 5-Digit	Generic Modbus 4-Digit	Tag Type	Access
Output Coils	1-65536	1-9999	1-9999	1-999	Discrete	Read/Write
Contacts	100001-165536	10001-19999	10001-19999	1001-1999	Discrete	Read-Only
Input	300001-365536	30001-39999	30001-39999	3001-3999	Analog	Read-Only
Holding	400001-465536	40001-49999	40001-49999	4001-4999	Analog	Read/Write
Extended	6x0000-6x9999		60000-69999		Analog	Read/Write

Note The x in the Extended register number indicates the file number, where x = 0 implies file number 1, x = 1 implies file number 2, up to x = 9 implies file number 10. The extended memory size in the PLC determines how many extended memory files exist. Each file contains up to 10000 registers. The last file in the PLC will always contain less than 10000 registers.

For example:

A 24K-extended-memory-size PLC contains three (3) files, where the last file contains 4576 registers.
A 72K-extended-memory-size PLC contains eight (8) files, where the last file contains 3728 registers.
A 96K-extended-memory-size PLC contains 10 files, where the last file contains 8304 registers.

1K is 1024 bytes, or 1024 registers.

Item Names Using the Modicon PLC Register Addresses

The following table lists other item names that are consistent with the register addresses used by the Modicon PLCs.

Item Name	Description
400001	When no spaces and no letters follow the register number, the register contents are treated as a 16-bit unsigned quantity.
400001 S	When a space and the letter "S" follow the register number, the register contents are treated as a 16-bit signed quantity.
400001 I	When a space and the letter "I" follow the register number, the register contents are treated as a 32-bit unsigned quantity. This takes up two consecutive registers.
400001 L	When a space and the letter "L" follow the register number, the register contents are treated as a 32-bit signed quantity. This takes up two consecutive registers.
400001 U	When a space and the letter "U" follow the register number, the register contents are treated as a 32-bit unsigned quantity. This takes up two consecutive registers.
400001 F	When a space and the letter "F" follow the register number, the register contents are treated as a floating-point quantity. This takes up two consecutive registers.

Item Name	Description
400001-400003 M	When a space and the letter "M" follow the register number or register number pair separated by a dash, the register contents are treated as ASCII data. Each register contains up to two (2) ASCII characters. This example represents six (6) ASCII characters.
300001:10	When a colon and a number from 1 (one) to 16 follow the register number, the register contents are treated as discrete data. This example represents bit 10 of the input register 300001.

Note The bit ordering (left-to-right or right-to-left) is determined by the setting of the Bit order format parameter during the PLC configuration. For more information on Bit order format, see ModbusSerial Hierarchy in the DAServer Manager.

Absolute Notation Item Names

The ModbusSerial DAServer also uses another naming convention called the Absolute Notation. This naming convention is independent of the PLC model numbers.

Absolute Notation allows access to the four Modbus data types, each with an address from 0 to 65535. The data types are indicated by the item name suffix characters.

Item Name	Description
nnnnn **DO**	Discrete Output Refers to the same data Modbus calls "coils." Valid range is 0 (zero) **DO** through 65535 **DO**.
nnnnn **DI**	Discrete Input Refers to the same data called "contacts" by Modbus. Valid range is 0 (zero) **DI** through 65535 **DI**.
nnnnn **IR**	Input Register Refers to the same data called "input register." Valid range is 0 (zero) **IR** through 65535 **IR**.
nnnnn **HR**	Holding Register Refers to the same data Modbus calls "holding register." Valid range is 0 (zero) **HR** through 65535 **HR**.
nnnnn **PV**	Process Variable Refers to holding register, but treated as floating points and assumes two (2) registers per floating-point number. Valid range is 0 (zero) **PV** through 32767 **PV**.

The **IR** and **HR** absolute notation can also be combined with the following conversions: **L** (long), **U** (unsigned long), **F** (floating), or **S** (signed).

For example:

- 219 **HRS** 16-bit signed integer
- 000 **HRL** 32-bit signed integer
- 000 **HRU** 32-bit unsigned integer
- 100 **HRF** 32-bit floating point

Modulo-10000 Point Item Names

The ModbusSerial DAServer uses the Modulo-10000 Points naming convention, where the item name is two registers separated by a dash, with no spaces and no letters following the registers.

Two or three consecutive registers may be interpreted as a single numeric quantity, and each of the component registers must be in the range of 0-9999.

Item Name	Description
400001-400002	Can represent numbers between 0 and 99,999,999. Register 400001 = <9999> and Register 400002 = <9999>.
400005-400007	Can represent numbers between 0 and 2,147,483,646. Register 400005 = <21>, Register 400006 = <4748>, and Register 400007 = <3646>.

- When grouping three consecutive registers for interpretation as a single numeric quantity, overflow becomes a possibility.
- The largest number that may be represented in the PLC with three consecutive Modulo-10000 registers is 999,999,999,999; however, the largest number that can be contained in an integer-type variable is 2,147,483,647. The latter number is used by the DAServer to represent an overflow condition.
- Therefore, the maximum usable value represented in three Modulo-10000 registers is 2,147,483,646 or (<21><4748><3646>). Any number larger than this will be clamped at 2,147,483,647.

Modulo-10000 Items, BCD Register Type, and Concept Data Structures

All the integer holding registers, 16- and 32-bit, signed and unsigned, and Modulo-10000 item types honor the configuration parameters Register type and Concept data structures.

When the Register type parameter is BCD and the Use Concept data structures (Longs) parameter is selected, the data is displayed in BCD and written to the PLC in the BCD format. In addition, data that takes up two registers (Longs), except Reals, is displayed and written in the Concept-data-structure format.

The same applies when the Register type parameter is Binary and the Use Concept data structures (Longs) parameter is selected; the data is displayed in Binary and written to the PLC in the Binary format. In addition, data that takes up two registers (Longs), except Reals, are displayed and written in the Concept-data-structure format.

When the Register type parameter is BCD and the Use Concept data structures (Longs) parameter is not selected, the data is displayed in BCD and written to the PLC in the BCD format. In addition, data that takes up two registers (Longs), except Reals, is displayed and written in the non-Concept-data-structure format.

The same applies when the Register type parameter is Binary and the Use Concept data structures (Longs) parameter is not selected; the data is displayed in Binary and written to the PLC in the Binary format. In addition, data that takes up two registers (Longs), except Reals, is displayed and written in the non-Concept-data-structure format.

The Concept-data-structure format implies that data is displayed and written the same way that the Concept program from Schneider Automation handles data.

Concept-data-structure format is where the data is displayed and written in the last-register-to-first-register order.
For example:
When writing the value 2147483646 to the Modulo-10000 item 400001-400003, the value 21 is written first to register 400003, then the value 4748 is written to register 400002, and then the value 3646 is written to register 400001.

The non-Concept-data-structure format is the opposite of the Concept-data-structure format.
The value 21 is written first to register 400001, then the value 4748 is written to register 400002, and then the value 3646 is written to register 400003.

Modulo-10000 items can be displayed and written in the BCD and Binary formats. When the Modulo-10000 item occupies two registers, the maximum value that can be displayed and written is 99999999. When the Modulo-10000 item occupies three registers, the maximum value that can be displayed and written is 2147483646.

Warning messages are logged and the client value status is updated when data is clamped high when reading or writing data. Warning messages will also be displayed when the PLC data does not convert to BCD correctly.

DAServer Standard System Items

System items provide you with easy access to the DAServer status and diagnostics information. They are treated just like ordinary items with respect to the client. However, in most cases these items are not directly acquired via the communications layer. System item values are usually generated through internal calculations, measurements, and the tracking of the DAS Engine.

No DAServer-specific system items are provided in this ModbusSerial DAServer.

System items, like ordinary items, are defined by the following properties:

- **Group** (client group/OPC group): The arbitrary collection of items, not correlated.
- **Hierarchical location** (link name/OPC path. The hierarchical node section of the fully qualified OPC item ID.): The device the item is attached to.
- **Device group** (OPC access path/topic, or a Scan Group on a hierarchical branch.): A collection of items on the same physical location with the same protocol update rate.

Example:

To check the status of an external device, the reference might be:

```
ModbusSerial.ModiconPLC1.$SYS$Status
```

Note This syntax does not refer to the access path/device group. As long as the data requested is from the same external device, the value will always be the same.

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

In the ArchestrA context, the device group plays the most important role of identifying the scope of any item (the device group defines the hierarchical location implicitly when using globally unique device-group names, which is required for DDE/SuiteLink compatibility).

All system items follow the same naming convention:

- All system items start with $SYS$.
- The DAS Engine scans and parses the name for system items.
  - Parsing of the name is case-insensitive.

All system items can be accessed through subscriptions to a device group. However, while some system items return data for that device group, others are server-wide.

DAServer Global System Item

The following system item refers to specific information regarding a global condition of the DAServer.

System Item Name	Type/ Access Rights	Description	Values
$SYS$Licensed	Boolean/ Read	Binary status indication of the existence of a valid license for the DAServer. If FALSE, this item causes the DAServer to stop updating existing tags, to refuse activation of new tags, and to reject write requests in addition to setting quality for all items to BAD. If TRUE, the DAServer functions as configured. All instances have the same value.	RANGE: 0, 1 1: Valid license exists. 0: No valid license exists.

DAServer Device-Specific System Items

The following system items refer to specific information regarding the device(s) the DAServer is connected to.

System Item Name (Type)	Type/ Access Rights	Description	Values
$SYS$Status	Boolean/ Read	Binary status indication of the connection state to the device (hierarchy level) the item is attached to. The device group (OPC access path/topic) does not affect the value. The status can be good even if individual items have errors. For DDE/SuiteLink clients, $SYS$Status always comes from the leaf level of a DAServer hierarchy branch, which is the destination PLC node. For OPC clients, $SYS$Status can be accessed at all hierarchy levels. $SYS$Status at the root level of the whole hierarchy tree is always good, as it represents the quality status of the local computer itself. Hence, for practical application, OPC clients should reference $SYS$Status at any hierarchy levels other than the root.	RANGE: 0, 1 1: DAServer connection to the device is intact. 0: Error communicating with the device.

System Item Name (Type)	Type/ Access Rights	Description	Values
$SYS$ErrorCode	Longint/ Read	Detailed error code of the communications state to the device. The device group (OPC access path/topic) does not affect the value.	>= 0: Good status (0 is the default state) – connected. >0: is some device state like: connecting, initializing, and so on. <0: Error status (value indicates the error).
$SYS$ErrorText	String/ Read	Detailed error string of the communications state of the device. The device group (OPC access path/topic) does not affect the value.	Descriptive text for the communications state corresponding to the error code.
$SYS$StoreSettings	Integer/ ReadWrite	Used to make the temporary update interval changes via the $SYS$UpdateInterval item permanent. If the client pokes a value of 1 into this system item, the currently set update interval is written to the server's configuration file. The value of this system item clears to 0 after being set, if the configuration file write is successful. If the write fails, then the value is set to -1. If the update interval has been changed via the $SYS$UpdateInterval item and this item is not poked to 1, the DAServer uses the original update interval for that topic the next time it is started. Reading the item always provides 0. Read/Write values are persisted only if the user sets this system item. The values other than this persist only for the life of the DAServer.	RANGE: -1, 0, 1 -1: Error occurred during saving the configuration file. 0: Read value always if status is OK. 1: Persist settings (cleared immediately).

DAServer Device-Group-Specific System Items

The following system items refer to specific information regarding device groups that have been configured in the DAServer.

System Item Name (Type)	Type/ Access Rights	Description	Values
$SYS$UpdateInterval	DWord/ ReadWrite	Used to access the currently set update interval. It is the current update interval of the device group in milliseconds. A client can poke new values into this item. The value of zero indicates that no non-system items on that topic are updated (data for these items are not acquired from the device).	RANGE: 1…2147483647 0: Topic inactive, no items are updated. Data acquisition is stopped. >0: Expected updated interval for the set of all items in the device group.
$SYS$MaxInterval	DWord/ Read	Used to access the currently measured maximum update interval, in milliseconds, of all items of the corresponding device group. This item is read-only. The value of the slowest item is displayed.	RANGE: 0…2147483647 0: If update interval is 0 or if the status is false. >0: Measured update interval
$SYS$WriteComplete	Integer/ ReadWrite	Used to access the state of pending write activities on the corresponding device group. On device group creation (adding items to an OPC group), the value of this system item is initially 1, indicating all write activities are complete – no pokes are pending. If values are poked into any items of the device group, the value of this item changes to 0, indicating write activity is currently in progress. If the server has completed all write activities, the value of this item changes to 1 if all pokes were successful or to -1 if at least one poke has failed. If the value of this item is not zero, the client can poke 1 or -1 to it (poke a 1 to clear errors, or a -1 to test a client reaction on write errors). If the value of this item is zero, it cannot be poked.	RANGE: -1, 0, 1 1: Write complete (no writes are pending – initial state). 0: Writes are pending. -1: Writes completed with errors.

System Item Name (Type)	Type/ Access Rights	Description	Values
$SYS$ReadComplete	Integer/ ReadWrite	Used to access the state of initial reads on all items in the corresponding device group. The value is 1 if all active items in a device group have been read at least once. If at least one item in the device group is activated, this item changes to 0. It changes to 1 if all items have been read successfully, or to -1 if at least one item has a non-good quality. Poking a 0 to this item resets the internal read states of all items in this device group. This resets this item to 0. If all items are read again after this poke, this item changes back to 1 or -1.	RANGE: -1, 0, 1 1: Read complete (all values have been read). 0: Not all values have been read. -1: All values have been read but some have a non-good quality.
$SYS$ItemCount	DWord/ Read	Used to access the number of items in the corresponding device group. This item is read-only.	RANGE: 0…2147483647 >=0: Number of active items.
$SYS$ActiveItemCount	DWord/ Read	Used to access the number of active items in the corresponding device group. This item is read-only.	RANGE: 0…2147483647 >=0: Number of active items.
$SYS$ErrorCount	DWord/ Read	Used to access the number of all items (active and inactive) that have errors (non-good OPC quality) in the corresponding topic. If the communications status of a device group is bad, all items have errors. This item is read-only.	RANGE: 0…2147483647 >=0: Number of all items (active and inactive) with errors.
$SYS$PollNow	Boolean/ ReadWrite	Poking a 1 to this item forces all items in the corresponding device group to be read immediately (all messages in this device group become due). This is useful if you want to force to get the newest values from the device, regardless of its update interval. This also works on device groups with a zero update interval (manual protocol triggering).	RANGE: 0, 1

DAServer-Specific System Items

The following system items refer to specific information regarding the DAServer. These system items are available at each PLC node level. They are all available as read items, and some of them are writable.

System Item Name (Type)	Type/ Access Rights	Description	Values
$SYS$PLCExceptionStatus	Integer/ Read	Coil Exception Status. Each bit status could be user-defined or pre-defined by the PLC.	RANGE: 1…2147483647 0: Topic inactive, no items are updated. Data acquisition is stopped. >0: Expected updated interval for the set of all items in the device group.
$SYS$PLCEventCount	Integer/ Read	PLC event counter value.	RANGE: 0…2147483647 0: If update interval is 0 or if the status is false. >0: Measured update interval
$SYS$PLCStatus	Integer/ Read	Last PLC Status word.	RANGE: -1, 0, 1 1: Write complete (no writes are pending – initial state). 0: Writes are pending. -1: Writes completed with errors.
$SYS$PLCMessageCount	Integer/ Read	PLC message count.	RANGE: -1, 0, 1 1: Read complete (all values have been read). 0: Not all values have been read. -1: All values have been read but some have a non-good quality.
$SYS$PLCClearCounters	Integer/ ReadWrite	Used to clear all PLC diagnostic counters. Reading this item returns the last write value, timestamp, and write status.	RANGE: 0…2147483647 >=0: Number of active items.
$SYS$PLCEventStatus	Integer/ Read	Last 64 PLC send and receive event status.	RANGE: 0…2147483647 >=0: Number of active items.
$SYS$PLCID	Integer/ Read	PLC Slave ID.	RANGE: 0…2147483647 >=0: Number of all items (active and inactive) with errors.
$SYS$PLCRunStatus	Integer/ Read	PLC run indicator status.	RANGE: 0, 1

System Item Name (Type)	Type/ Access Rights	Description	Values
$SYS$PLCSpecificInfo	Integer/ Read	PLC Specific information from 2 to 72 bytes, depending on the model of the PLC.	
$SYS$PLCFIFO	Integer/ Read	Register data in the PLC FIFO queue (up to 31 registers of data, that is, 62 bytes). The item requires a register number to be specified right after the item name. The format is "$SYS$PLCFIFO-4xxxxx" for a 6-digit holding register. For example, if "$SYS$PLCFIFO-405000" is the request to the PLC and the PLC FIFO queue has 10 registers, the PLC register 405000 would have a count of 10, then the contents of registers 405001 through 405010 are returned and displayed to you.	
$SYS$PLCDiagLoopbackWrite	Integer/ ReadWrite	Write 2 bytes of data to the PLC. Reading this item returns the last write value, timestamp, and write status.	
$SYS$PLCDiagLoopbackRead	Integer/ Read	2 bytes of data received from the PLC from the last loopback write. If loopback write has not been performed, this system item will not have any value.	
$SYS$PLCRestartComm	Integer/ ReadWrite	Restart communications of the PLC. If a value of 0 is written, the PLC communications event log will be left prior to the restart. If a hex value of 'FF00' is written, the PLC communications event log will be cleared. Reading this item returns the last write value, timestamp, and write status.	
$SYS$PLCDiagnosticRegister	Integer/ Read	Used to get the PLC diagnostic Register values. Note that the value of the registers depends on the type of PLC connected.	
$SYS$PLCSetListenMode	Integer/ ReadWrite	Used to set the PLC to Listen-Only mode. The value of the input value is irrelevant. Reading this item returns the last write value, timestamp, and write status.	
$SYS$PLCErrorCount	Integer/ Read	Used to get the number of CRC errors encountered by the PLC since the last counter reset.	

System Item Name (Type)	Type/ Access Rights	Description	Values
$SYS$PLCException Count	Integer/ Read	Used to get the number of Modbus exception responses encountered by the PLC since the last counter reset	
$SYS$PLCNAKCount	Integer/ Read	Used tp get the number of NAK (negative acknowledgement) exception responses detected by the PLC since the last counter reset	
$SYS$PLCBusyCount	Integer/ Read	Used to get the number of messages destined for the PLC from which it returned a Save Device Busy exception response.	
$SYS$PLCBusOverru n	Integer/ Read	Used to get the number of messages destined for the PLC that it could not handle due to a character overrun condition.	

Generic OPC Syntax

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

The fully qualified name for an OPC item is called the Item ID (equivalent to Item Name). The syntax for specifying a unique Item ID is DAServer-dependent. In OPC data acquisition DAServers, the syntax can be as follows:

```
AREA10.VESSEL1.TIC1.PLC.400001
```

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

In this example:

- AREA10.VESSEL1.TIC1 is the link name for a DAServer.
- PLC is the name of the target PLC.
- 400001 is the specific data point (Item) desired.
- An item is typically a single value such as an analog, digital, or string value.

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

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

C H A P T E R 4

Troubleshooting

This chapter describes troubleshooting tools that can be used to deal with the ModbusSerial DAServer problems you may encounter.

The DAServer Manager provides access to diagnostics and other statistical data, and the Log Viewer provides access to event messages logged during the operation of a DAServer. Also, your client (for example, InTouch) can monitor connectivity with the PLC through the $SYS$Status item. Use these tools together with the information in this section to troubleshoot your ModbusSerial DAServer.

Note In order to determine the version of your DAServer, perform the following steps. Search for DASMBSerial.dll, right-click on the **File Name**, select **Properties** on the shortcut menu, and select the **Version** tab on the **Properties** dialog box. The version of your DAServer is listed under **File Version**.

Contents

Monitoring Connectivity Status with the PLC

The built-in discrete item, $SYS$Status, can be used to monitor the status of communications with the PLC. This item is set to the following:

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

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

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

=DASMBSerial|ModiconPLC!$SYS$Status

where:

DASMBSerial	is the name of the DAServer application.
ModiconPLC	is the exact device group defined in the DAServer for the PLC.
$SYS$Status	is the discrete item used to monitor the status of connectivity with the PLC.

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

YourLinkName.$SYS$Status

where:

YourLinkName	is the assembly of hierarchy node names leading to a specific controller device.
$SYS$Status	is the discrete item used to monitor the status of connectivity with the controller device.

Note In the case of a PLC disconnect, the DAServer will retry three times before entering into slow poll mode. In the case of reply time-out, the DAServer will go into slow poll mode immediately.

Monitoring the Status of DAS Conversations

The **InTouch WindowViewer** supports built-in topic names, called **DDEStatus** and **IOStatus**, that can be used to monitor the status of specific DAS conversations.

For example, let us assume that **WindowViewer** (**VIEW**) is communicating with the ModbusSerial DAServer to a PLC that has been defined in the DAServer with the topic name **ModiconPLC**. The discrete items, **DDEStatus** and **IOStatus**, are set to:

- 0 (zero) when this DAS conversation failed.
- 1 (one) when this DAS conversation is successful.

Using DDEStatus and IOStatus in Excel

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

=view|DDEStatus!ModiconPLC

or

=view|IOStatus!ModiconPLC

where:

view	is the name of the InTouch application.
[DDE][IO] Status	is the built-in topic name used to monitor the status of communications between the DAServer and InTouch.
ModiconPLC	is the exact topic name defined in the server for the PLC.

Reading Values from the DAServer into Excel

Values may be read directly into Excel spreadsheets from the DAServer by entering a DDE formula into a cell using the following format:

=applicationname|<devicegroup>!itemname

Example formula:

=DASMBSerial|ModiconPLC!'<tagname>'

where:

DASMBSerial	is the name of the DAServer application.
ModiconPLC	is the exact device group name defined in the DAServer for the PLC.
<tagname>	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 **<tagname>** changes in the PLC, the DAServer will automatically send the new value to the cell containing the formula in Excel.

Note Refer to the Microsoft Excel manual for complete details on entering Remote Reference formulas for cells.

Writing Values to the DAServer from Excel

Values may be written to the DAServer from Microsoft Excel 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 (defined in the DAServer) in a particular application name (the executable name less the **.exe**).
- Assigns the number of that opened channel to **channel**.

Note By 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 (actual location in the PLC), via the **channel** number 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.
- Some applications have a limited number of channels; therefore, they should be closed when finished.
- **Channel** is the channel number returned by the previously executed **INITIATE** function.

=RETURN()

- Marks the end of the macro.

Note Refer to the **.xlm** sample Excel poke macro provided on the DAServer CD. Also refer to the Microsoft Excel manual for complete details on entering Remote Reference formulas for cells.

Error Messages and Codes

Generic DAServer error messages and ModbusSerial-DAServer-specific messages are supported. Use the Log Flag data to customize the messages logged to the Log Viewer. See the Log Viewer online documentation for more information about using log flags.

To troubleshoot DAServer problems, use the following error messages together with the DAServer Manager Diagnostics root data.

In the following DAServer Error Messages table:

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

DAServer Error Messages

The following table lists all the generic-DAServer and ModbusSerial-DAServer-specific error messages that are logged to the Log Viewer.

Error Message	Explanation	Probable Cause	Solution	Log Flag
"<PortPlcname>" does not allow extended register "<Register>"	An extended register address (starting with "6") is used for polling or poking.	Extended registers are not defined in the Config. file or the PLC does not support extended registers.	If the PLC supports extended registers, make sure the config. file contains their definitions.	DASProtFail
"BaudRate" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <BaudRate> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"BitOrderFormat" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <BitOrderFormat> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"CoilRead" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <CoilRead> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"CoilWrite" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <CoilWrite> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"DataBits" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <DataBits> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"EnableConnectDisconnectMode" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <EnableConnectDisconnectMode> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
"EnableDSR_DTRSupport" is missing from DSMBSerial.aacfg file under <COM Port Name>.	The mandatory <EnableDSR_DTRSupport> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"EnableRTS_CTSSupport" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <EnableRTS_CTSSupport> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"ExtendedRegisterRead" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <ExtendedRegisterRead> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"ExtendedRegisterWrite" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <ExtendedRegisterWrite> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"HoldingRegisterRead" is missing from Config. file under <PLC Name>.	The mandatory <HoldingRegisterRead> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"HoldingRegisterWrite" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <HoldingRegisterWrite> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"InputRegisterRead" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <InputRegisterRead> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
"MaxAddrExtendedRegisters" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <MaxAddrExtendedRegisters> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"MaxAddrHoldingRegisters" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <MaxAddrHoldingRegisters> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"MaxAddrInputRegisters" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <MaxAddrInputRegisters> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"MaxAddrReadCoils" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <MaxAddrReadCoils> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"MaxAddrWriteCoils" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <MaxAddrWriteCoils> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"ModemConnectionTimeout" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <ModemConnectionTimeout> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"Parity" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <Parity> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
"PortName" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <PortName> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"ReadIntervalTimeout" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <ReadIntervalTimeout > field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"RegisterSize" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <RegisterSize> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"RegisterType" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <RegisterType> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"ReplyTimeout" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <ReplyTimeout> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"SlaveAddress" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <SlaveAddress> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"StopBits" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <StopBits> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"StringVariableStyle" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <StringVariableStyle> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
"SupportMultiCoilWrite" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <SupportMultiCoilWrite> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"SupportMultiRegisterWrite" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <SupportMultiRegisterWrite> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"TransmissionMode" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <TransmissionMode> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"TurnaroundDelay" is missing from DASMBSerial.aacfg file under <COM Port Name>.	The mandatory <TurnaroundDelay> field is absent from the DeviceNode named <COM Port Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to build the file.	DASProtFail
"UseLongConceptDataStruct" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <UseLongConceptDataStruct> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
"UseRealConceptDataStruct" is missing from DASMBSerial.aacfg file under <PLC Name>.	The mandatory <UseRealConceptDataStruct> field is absent from the DeviceNode named <PLC Name> in the configuration file.	The entry is deleted from the file manually.	Use the DAServer Manager to rebuild the file.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 01: ILLEGAL FUNCTION	The function code in the query sent to the PLC is not an allowable action for this PLC.	The communications data is corrupted.	Check the item definition and PLC configuration.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
<COM Port> Slave <PLC> encountered exception error 02: ILLEGAL DATA ADDRESS (<Category Name>: range was [<Start Addr>...<End Addr>])	The data address in the query sent to the PLC is not allowed for this PLC.	The data addresses <Start Addr> … <End Addr> are not allowable addresses for the PLC.	Check the item definition and PLC configuration.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 03: ILLEGAL DATA VALUE.	The data value in the query sent to the PLC is not an allowable value for this PLC.	An illegal data value was sent to the PLC.	Check the data value sent to the PLC.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 04: SLAVE DEVICE FAILURE.	An unrecoverable error occurred while the PLC was attempting to perform the requested action.	The PLC failed.	Check the PLC.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 05: ACKNOWLEDGE.	The PLC is still working on an action but returned ACKNOWLEDGE so that the query will not time out.	The PLC program is taking too long to respond to the DAServer.	Check the PLC.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 06: SLAVE DEVICE BUSY.	The PLC is engaged in a long duration program command.	The PLC program is taking too long to respond to the DAServer.	Check the PLC.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 07: NEGATIVE ACKNOWLEDGE.	The PLC cannot perform the program function received in the query.	The PLC program did not complete the requested operation.	Check the PLC.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 08: MEMORY PARITY ERROR.	The PLC cannot perform the program function received in the query because of an extended memory error.	The PLC might have bad extended memory.	Check the PLC's extended memory.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 0A: GATEWAY PATH UNAVAILABLE.	The PLC cannot perform the program function received in the query because of a Bridge error.	The Bridge connected to the PLC might have a problem.	Check the Bridge connected to the PLC.	DASProtFail
<COM Port> Slave <PLC> encountered exception error 0B: GATEWAY TARGET DEVICE FAILED TO RESPOND.	The PLC cannot perform the program function received in the query because of a Bridge error.	The Bridge connected to the PLC might have a problem.	Check the Bridge connected to the PLC.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
<COM Port> Slave <PLC> encountered exception error XX:	The PLC cannot perform the program function received in the query. The PLC returned an unknown error code.	The communications data is corrupted.	Check the PLC.	DASProtFail
<Item Name> is read-only. No poking allowed.	The item is a read-only item and cannot be written.	The client is writing to a read-only item.	Check the client application.	DASProtFail
<Item Name> is write-only. No polling allowed.	The item is a write-only item and cannot be polled.	The client is polling a write-only item.	Check the client application.	DASProtFail
<Port.PLC> Clamping a write to <Clamped Value> - Received <Received Value>, writing <Final Value>.	The DAServer encountered write data that was out of range.	The DAServer clamped a write value because it was out of range.	Check the client application.	DASProtWarn
A floating point value read for Item: <Register> on Node: <Port.PLC> was NOT A VALID NUMBER (!NaN!), therefore, it was converted to +3.4e38	Invalid floating-point number. Set the item value to 3.4e+38.	The values read from the PLC registers cannot be converted to a float number.	Check the PLC for the value in the registers.	DASProtFail
A floating point value read for Item: <Register> on Node: <Port.PLC> was NOT A VALID NUMBER (Negative Infinity), therefore, it was converted to -3.4e38	Invalid floating-point number (Negative Infinity). Set the item value to Negative Infinity -3.4e+38.	The values read from the PLC registers cannot be converted to a valid float number.	Check the PLC for the value in the registers.	DASProtFail
A floating point value read for Item: <Register> on Node: <Port.PLC> was NOT A VALID NUMBER (Positive Infinity), therefore, it was converted to 3.4e38	Invalid floating-point number (Positive Infinity). Set the item value to Positive Infinity 3.4e+38.	The values read from the PLC registers cannot be converted to a valid float number.	Check the PLC for the value in the registers.	DASProtFail
A valid license CAN NOT BE retrieved	License to use the DAServer has not been installed.	No license is available for this DAServer.	Report it to the system administrator.	DASProt Warn
Bit mask out of range for register "<Register>:<Mask>".	The bit mask value <Mask> is not in the range of 1 (one) to 16.	<Mask> must be between 1 (one) and 16.	Correct the bit mask value.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
DASMBSerial failed to allocate memory.	The DAServer cannot allocate memory to continue running.	The DAServer requires more memory than is available.	Add more memory to the computer or other programs that may be taking up memory.	DASProtFail
Demo mode expired, halting operation.	Demo mode expired and the DAServer stays in idle mode.	There is no valid license for this DAServer and the demo mode expired.	Report it to the system administrator.	DASProtWarn
Failed to build the query.	The DAServer encountered an internal error in building a query for a write operation.	Internal error.	Check CPU memory.	DASProtWarn
Failed to connect in <Number Attempts> attempts. Revoking all messages.	The DAServer failed to connect to the modem in the configured redial attempts.	The modem failed.	Check the PLC, the modem, and the configuration parameters.	DASProtFail
Failed to get handle for COM port.	The DAServer had problems dialing a modem.	The modem did not respond or the Control Panel settings are incorrect.	Check the PLC, the modem, and the configuration parameters.	DASProtWarn
Failed to get valid poke Data from Engine, cancel this write!	Invalid data was encountered by the DAServer.	The client application wrote invalid data.	Check the client application.	DASProtFail
Failed to store the message in internal queue.	The DAServer encountered an internal error in building a query for a write operation.	Internal error.	Check CPU memory.	DASProtWarn
Failed to trigger an event of MessageDone.	The DAServer encountered an internal error in building a query for a read/write operation.	Internal error.	Check CPU memory.	DASProtWarn
Function code 6-'Preset Single Register' cannot be used for multiple registers for item: <Item Name> on Node: <Port.PLC>.	Support Multi Register Write was not selected and multiple registers are being written.	The Support Multi Register Write configuration parameter was not selected.	Select the Support Multi Register Write configuration parameter.	DASProtFail
Invalid bit mask for register "<Register size>:nn".	The bit mask value nn is invalid. It must be between 1 (one) and 16.	nn has non-numeric characters.	Correct the bit mask value.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
Invalid call state - Call state must be CS_IDLE.	The DAServer encountered a problem with the local modem.	The local modem failed.	Check the local modem.	DASProtWarn
Invalid data type for register "<Register> <T>".	<T> is not one of the allowed types.	<T> is not one of the following: S, L, F, U, M, DO, DI, IR, HR, PV.	Correct the data type.	DASProtFail
Invalid Modulo10k point: "<Register1>-<Register2>". Please check range.	<Register1> and <Register2> do not form a valid Modulo10k point.	The range is not in correct order, or there are more than 3 (three) registers in the point.	Correct the beginning and ending register values.	DASProtFail
Invalid Modulo10K point: <Port.PLC.Register>	The <Register> does not form a valid Modulo10K point.	The client application defined an invalid item name.	Check the client application.	DASProtFail
Invalid port & plc names: <Port.PLC>	The port and PLC name are invalid.	The client application defined an invalid item name path.	Check the client application.	DASProtWarn
Invalid register <Register> for register size of <Register size>.	The number of digits in the register number does not match the number of digits defined by the Register Size configuration parameter.	The client application defined an invalid item name.	Check the client application.	DASProtFail
Invalid register size: <Register size>.	A register size of neither 4 (four), 5 (five), nor 6 (six) is detected.	The user has changed the register size to a value other than 4 (four), 5 (five), or 6 (six) in the DAServer Manager.	Change the register size to 4 (four), 5 (five), or 6 (six) in the DAServer Manager.	DASProtFail
Invalid Register type.	The defined register number is invalid.	The client application defined an invalid item name.	Check the client application.	DASProtWarn
Invalid register(non-numeric): <Register>	The value in <Register> contains non-numeric characters.	Register names must be numeric.	Use the correct name.	DASProtFail
Invalid register: <Register>.	<Register> does not belong to any category in the Modicon controller.	The server could not find a category. For example, 200001.	Use the correct register address.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
Invalid sub function code.	The DAServer encountered an invalid diagnostic item name.	The client application defined an invalid diagnostic item name.	Check the client application.	DASProtWarn
Invalid value read for Modulo-10000 point: <Item Name> on Node: <Port.PLC>. Not a valid BCD-value. Convert to <Value>	The values in the consecutive component registers of the Modulo-10000 point are not in the valid range. The Modulo-10000 point is set to 9999,9999 or 2,147,483,647.	The values in some component registers are larger than 9999.	Check the PLC for the value in the registers.	DASProtWarn
License is invalid, starting demo mode.	The DAServer will operate in demo mode.	There is no valid license for this DAServer.	Report it to the system administrator.	DASProt Warn
Message <Message Address>: Failed to verify Modbus ASCII frame.	The PLC responded with an invalid message.	Invalid response by the PLC.	Check the PLC.	DASProtFail
Message <Message Address>: Failed to verify Modbus RTU frame.	The PLC responded with an invalid message.	Invalid response by the PLC.	Check the PLC.	DASProtFail
Message Queue pointer is equal to NULL.	The DAServer encountered an internal error in building a query for a read/write operation.	Internal error.	Check CPU memory.	DASProtWarn
PLC address in received packet is not matching.	The PLC responded with an invalid PLC address.	Invalid response by the PLC.	Check the PLC.	DASProtFail
PLC poke message timed out on port <Port Name>, revoking message <Message Address> at <Current Time>.	In a write operation the PLC did not respond within the configured reply time.	The configured reply timeout may be too short.	Check the configured reply timeout and the PLC.	DASProtFail
Pointer of call object is NULL.	The DAServer encountered an internal error in connecting to a modem.	Internal error.	Check CPU memory.	DASProtWarn
Property Value of Line is changed - Need to restart the server.	A Line configuration parameter has changed.	A Line configuration parameter was changed and the DAServer was not restarted.	Restart the DAServer.	DASProtWarn

Error Message	Explanation	Probable Cause	Solution	Log Flag
Property Value of PLC is changed - Need to restart the server.	A PLC configuration parameter has changed.	A PLC configuration parameter was changed and the DAServer was not restarted.	Restart the DAServer.	DASProtWarn
Property Value of Port is changed - Need to restart the server.	A PLC configuration parameter has changed.	A PLC configuration parameter was changed and the DAServer was not restarted.	Restart the DAServer.	DASProtWarn
Property Value of Station is changed - Need to restart the server.	A PLC configuration parameter has changed.	A PLC configuration parameter was changed and the DAServer was not restarted.	Restart the DAServer.	DASProtWarn
Read value beyond limits for Modulo-10000 point: <Item Name> on Node: <Port.PLC>. Value clamped to <Value>.	The Modulo-10000 point value overflows. The value is clamped.	The maximum usable value represented in three Modulo-10000 registers is 2,147,483,646.	Check the PLC for the value in the registers.	DASProtWarn
ReadChars error = <Error Code>.	The DAServer encountered an error when communicating with the COM port.	The connection to the PLC may have been lost.	Check the PLC, the modem, and the configuration parameters.	DASProtFail
Received partial packet: <Packet Data>.	The PLC responded with partial data.	The PLC responded with an incomplete message.	Check the PLC.	DASProtWarn
Redial after <Number Seconds> Sec.	The DAServer is redialing after <Number Seconds> have elapsed.	The modem may be having problems.	Check the PLC, the modem, and the configuration parameters.	DASProtWarn
Register "<Register>" out of range according to configuration.	<Register> is not in range according to specifications in the configuration file.	<Register> is out of range for its category according to the maximum addressable register value in the configuration file.	Correct the maximum addressable value in the config. file or change <Register>.	DASProtFail
Register Type is hot-configured to <New Register Type>.	The PLC configuration parameter Register Type was modified, and will take effect without restarting the DAServer.	A PLC configuration parameter was modified.	Check the PLC configuration parameters.	DASProtWarn

Error Message	Explanation	Probable Cause	Solution	Log Flag
Reply Timeout is hot-configured to <Number Seconds> secs.	The PLC configuration parameter Reply Timeout was modified, and will take effect without restarting the DAServer.	A PLC configuration parameter was modified.	Check the PLC configuration parameters.	DASProtWarn
Revoking all messages.	The DAServer could not connect to the PLC.	The PLC or the modem may be offline.	Check the PLC, the modem, and the configuration parameters.	DASProtWarn
String <Item Name> item cannot be more than <Register Limit> registers: <Port.PLC.Register>	The string item name exceeds the maximum number of registers.	The client application defined an invalid item name.	Check the client application.	DASProtFail
String Variable Style is hot-configured to <New string style>.	The PLC configuration parameter String Variable Style was modified, and will take effect without restarting the DAServer.	A PLC configuration parameter was modified.	Check the PLC configuration parameters.	DASProtWarn
TAPI CoCreateInstance failed.	Using the TAPI software the DAServer encountered an internal error in connecting to a modem.	Internal error.	Check CPU memory.	DASProtFail
TAPI could not create a call object: <Telephone Number or Radio Modem Number>.	The DAServer using the TAPI software could not connect to the PLC modem.	Invalid modem configuration parameters or the modem settings are invalid.	Check the PLC, the modem, and the configuration parameters.	DASProtFail
TAPI could not get the Comm Port handle for the call object: <Telephone Number or Radio Modem Number>.	The DAServer, using the TAPI software, could not connect to the PLC modem.	Invalid modem configuration parameters or the modem settings are invalid.	Check the PLC, the modem, and the configuration parameters.	DASProtFail
TAPI dialing failed.	The DAServer, using the TAPI software, had problems dialing a modem.	The modem did not respond or the Control Panel settings are incorrect.	Check the PLC, the modem, and the configuration parameters.	DASProtFail
TAPI- failed to find an address. Please check Modem settings using control panel.	The DAServer using the TAPI software could not connect to the PLC modem.	Invalid modem configuration parameters or the modem settings are invalid.	Check the PLC, the modem, and the configuration parameters.	DASProtFail

Error Message	Explanation	Probable Cause	Solution	Log Flag
TAPI failed to initialize.	Using the TAPI software the DAServer encountered an internal error in connecting to a modem.	Internal error.	Check CPU memory.	DASProtFail
The call is disconnected.	The PLC on the modem gets disconnected.	The PLC dropped off the modem line.	Check the PLC, the modem, and the configuration parameters.	DASProtFail
The COM port handle is invalid.	The DAServer encountered an error when communicating with the COM port.	The connection to the PLC may have been lost.	Check the PLC, the modem, and the configuration parameters.	DASProtFail
This is not a holding Register. Failed to write a BOOL value.	Attempt to write a bit of a non-holding register.	A non-holding register was defined to write a bit value.	Check the client application.	DASProtWarn
Unable to connect the call.	The DAServer could not communicate with the modem.	Invalid modem configuration parameters or the modem settings are invalid.	Check the PLC, the modem, and the configuration parameters.	DASProtWarn
Unable to decode received packet.	The PLC responded with an invalid message.	Invalid response by the PLC.	Check the PLC.	DASProtFail
Unable to initialize Modem connection.	The DAServer could not communicate with the modem.	Invalid modem configuration parameters or the modem settings are invalid.	Check the PLC, the modem, and the configuration parameters.	DASProtWarn
Valid license is available	The DAServer will operate in the normal mode.			DACmnProtWarn
Write attempt beyond limits for Modulo-10000 register: <Register> on Node: <PLC>. Value clamped to <Value>	The value attempted to write to the Modulo-10000 point is out-of-range. Clamp the value, and then write to the PLC.	The attempted write value is greater than the maximum usable value for Modulo-10000 point.	Check the client application that writes to the DAServer.	DASProtWarn

Error Message	Explanation	Probable Cause	Solution	Log Flag
Write attempt beyond limits for string item: <Register> on Node: <PLC>. value truncated to <Value>	The string attempted to write to the PLC is too long. Truncate it to the maximum length that the registers can hold.	An attempt was made to write a string to consecutive registers with the string length longer than the registers can hold.	Check the client application that writes to the DAServer.	DASProtFail
WriteChars error = <Error Code>.	The DAServer encountered an error when communicating with the COM port.	The connection to the PLC may have been lost.	Check the PLC, the modem, and the configuration parameters.	DASProtFail

Note In the preceding table, <Register> refers to the complete pathname of a register: "Portname.Plcname.Register."
For example, COM1.QuantumPlc1.400001.

Server-Specific Error Codes

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

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

C H A P T E R 5

Reference

Contents

DAServer Architecture

Note DAServers are supported only on Microsoft Windows 2003, Windows 2000, Windows XP, and Windows XP embedded. NetDDE protocol is not supported by DAServers.

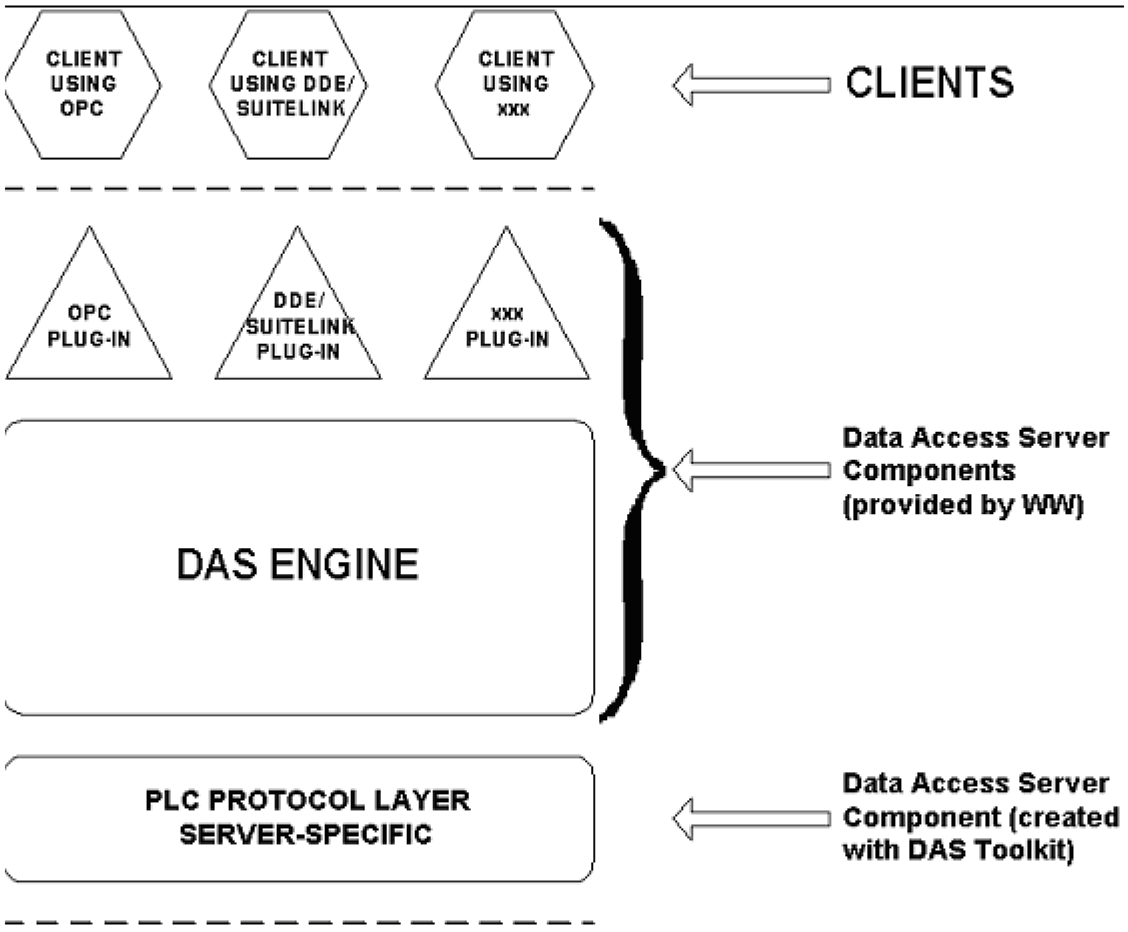
This DAServer is a collection of components that work in concert to provide communications access with hardware field devices. These components include:

- **DAServer Manager**: This is the Microsoft Management Console (MMC) snap-in, which is part of the ArchestrA System Management Console suite of utilities, supplied with the DAServer. It provides the necessary user-interface for diagnostics, configuration, and activation.
- **Client Plug-ins**: These are the components that are added to a DAServer to enable communications with clients.
  Examples are: OPC, DDE/Suitelink, and so on.
- **DAS Engine**: This is the library that contains all the common logic to drive data access.
- **Device Protocol**: This is the custom code provided by this DAServer to define the communications with a particular device.

DAServers

A DAServer is comprised of three physical parts (see the following figure). They are the following:

- **Plug-in Component(s)**: Responsible for communicating with clients.
- **DAS Engine**: This common component is used by all DAServers.
- **PLC Protocol Layer**, DAServer-specific: This component is responsible for communicating with the hardware.


CLIENT USING OPC
CLIENT USING DDE/ SUITELINK
CLIENT USING xxx
CLIENTS
OPC PLUG-IN
DDE/ SUITELINK PLUG-IN
xxx PLUG-IN
Data Access Server Components (provided by WW)
DAS ENGINE
PLC PROTOCOL LAYER SERVER-SPECIFIC
Data Access Server Component (created with DAS Toolkit)

DAServer Architecture

Each physical part of a DAServer is comprised of a set of .exe and/or .dll modules. Wonderware provides the Plug-ins and the DAS Engine. The DAS Toolkit user creates the PLC Protocol Layer (DAServer-specific) modules. All three sets of modules are required for a fully functioning DAServer.

Plug-ins

Plug-ins provide a protocol-translation function for device integration clients. Typical Plug-ins communicate in DDE, SuiteLink, or OPC protocol, and serve as interfaces between their clients and the DAS Engine.

Note Items of an array are not supported in the DDE/SL plug-in. These arrays are converted to HEXASCII strings, which provide legacy behavior for DAServers that support this in the DAServer-specific code.

DAS Engine

The DAS Engine is a middleware component that exposes two sets of unique interfaces, one for communicating with the Plug-ins and the other one for communicating with the PLC Protocol Layer components.

PLC Protocol Layer

The PLC Protocol Layer provides a protocol-translation function for specific hardware, such as ModBus; and it serves as an interface between the DAS Engine and the hardware.

Component Environments

Stand-alone DAServers have the following characteristics:

- The DAS Engine is dynamically linked to the other DAServer components. In other words, a new DAS Engine (feature enhancement or bug fix) would not require relinking to the other components. When deployed to the system, the new DAS Engine would attach to all existing DAServer components.
- Newly deployed Plug-ins (feature enhancements or bug fixes) do not require relinking. Even new Plug-ins (for example, OPC Alarm & Events) would not require any development changes to the other components, and therefore no relinking in a customer- installed base. In fact, it is feasible to implement new functionality in a Plug-in to enhance the DAServer without any involvement of the code of the other components.
- DAServers can be configured in one stand-alone configuration utility (DAServer Manager). The DAServer Manager is capable of displaying specific configuration views for all DAServers. This utility allows the browsing and editing of DAServers on different nodes.
- The DAServer Manager diagnostic tool displays generic diagnostic objects common to all DAServers, in addition to the DAServer-specific/ DAServer-developer-defined diagnostic data.

The DAServer data configuration format is XML. Any XML-enabled program (for example, XML Editor) can read this format.

A P P E N D I X A

Supported DASModbusSerial Hardware and Firmware

The following table lists the hardware and firmware supported by the ModbusSerial DAServer Version 2.5.

Note The ModbusSerial DAServer was tested using the ELPRO 905U-D Hayes-compatible Radio Modem and the US Robotics Phone Modem.

Legend	Model	Description	Firmware
TSX Quantum	140 CPU 21304	Controller	Exec ID 0871 HID 0405 Rev 02.10
TSX Momentum		Controller	Exec ID v.1.02, 1.03, 1.04, 1.06, 1.07
Modicon Micro	110 CPU 311 01	Controller	Exec ID 0863 HID 0701 Rev 01.10 Model 311/01
Generic Modbus Controller (4-Digit, 5-Digit, 6-Digit)		Generic Modbus Controller. Controller must conform to the Modbus "Application Protocol Specifications."	

Legend	Model	Description	Firmware
Hayes Data Modem		Hayes-compatible serial data (phone) modem. Modem that supports the Modbus protocol is highly recommended.	
Hayes Radio Modem		Hayes-compatible serial radio modem; must be able to accept commands issued to the phone modems. Modem that supports the Modbus protocol is highly recommended.	

A P P E N D I X B

The Modbus Protocol

This appendix describes the Modbus codes, serial communications, data types, and message frame checking that are supported by the ModbusSerial DAServer.

Contents

Controller Function Codes

The ModbusSerial DAServer uses function codes to communicate with the various controllers supporting the Modbus protocol. The implementation for the MBSerial DAServer uses the document from Modbus.org, "MODBUS Application Protocol Specification V1.1a", dated June 4, 2004 as a reference.

In communicating with the Generic Modbus controllers, the ModbusSerial DAServer acts as a master and sends out the following function codes to the controllers.

Function Code	Name	Description
01 (0x01)	Read Coils	Reads the ON/OFF status of discrete outputs (0X references, coils) in the slave.
02 (0x02)	Read Discrete Inputs	Reads the ON/OFF status of discrete inputs (1XXXXX references) in the slave.
03 (0x03)	Read Holding Registers	Reads the binary contents of holding registers (4XXXXX references) in the slave.

Function Code	Name	Description
04 (0x04)	Read Input Registers	Reads the binary contents of input registers (3XXXXX references) in the slave.
05 (0x05)	Write Single Coil	Forces a single coil (0XXXXX reference) to either ON or OFF.
06 (0x06)	Write Single Register	Presets a value into a single holding register (4XXXXX reference).
07 (0x07)	Read Exception Status	Reads the contents of eight Exception Status coils within the slave controller.
08 (0x08)	Diagnostics	Provides a series of tests for checking the communications system between the master and the slave, or for checking various internal error conditions within the slave.
12 (0x0C)	Fetch Comm Event Log	Returns a status word, event count, message count, and a field of event bytes from the slave.
15 (0x0F)	Write Multiple Coils	Forces each coil (0XXXXX reference) in a sequence of coils to either ON or OFF.
16 (0x10)	Write Multiple Registers	Presets values into a sequence of holding registers (4XXXXX references).
17 (0x11)	Report Slave ID	Returns a description of the type of controller present at the slave address, the current status of the slave-run indicator, and other information specific to the slave device.
20 (0x14)	Read General Reference	Returns the contents of registers in the Extended Memory file (6XXXXX references).
21 (0x15)	Write General reference	Writes the contents of registers in the Extended Memory file (6XXXXX references).
22 (0x16)	Mask Write Holding Register	Modifies the contents of a specified (4XXXXX reference) holding register using a combination of an AND mask, an OR mask, and the register's current contents. The function can be used to set or clear individual bits in the register.
24 (0x18)	Read FIFO Queue	Reads the contents of a First-In-First-Out (FIFO) queue of 4XXXXX registers.
Sub-function Codes of Function Code 08		

Function Code	Name	Description
00 (0x00)	Return Query Data	The data passed in the query data field is to be returned (looped back) in the response.
01 (0x01)	Restart Communications Option	The slave's peripheral port is to be initialized and restarted, and all of its communications event counters are to be cleared.
02 (0x02)	Return Diagnostic Register	The contents of the slave's 16-bit diagnostic register are returned in the response.
03 (0x03)	Change ASCII Input Delimiter	The character 'CHAR' passed in the request data field becomes the end of message delimiter for future messages (replacing the default LF character). This function is useful in cases of a Line Feed is not required at the end of ASCII messages.
04 (0x04)	Force Listen Only Mode	Forces the addressed slave to its Listen Only Mode for Modbus communications.
10 (0x0A)	Clear Counters and Diagnostic register	For controllers other than the 584 or 984, clears all counters and the diagnostic register. For the 584 or 984, clears the counters only.
12 (0x0C)	Return Bus Communications Error Count	The response data field returns the quantity of CRC errors encountered by the slave since its last restart, clear counters operation, or power-up.
13 (0x0D)	Return Bus Exception Error Count	The response data field returns the quantity of Modbus exception responses returned by the slave since its last restart, clear counters operation, or power-up.
16 (0x10)	Return Slave NAK Count	The response data field returns the quantity of messages addressed to the slave for which it returned a Negative Acknowledge (NAK) exception response, since its last restart, clear counters operation, or power-up.
17 (0x11)	Return Slave Busy Count	The response data field returns the quantity of messages addressed to the slave for which it returned a Slave Device Busy exception response, since its last restart, clear counters operation, or power-up.

Function Code	Name	Description
18 (0x12)	Return Bus Character Overrun Count	The response data field returns the quantity of messages addressed to the slave that it could not handle due to a character overrun condition, since its last restart, clear counters operation, or power-up.
20 (0x14)	Clear Overrun Counter and Flag	Clears the overrun error counter and resets the error flag.

Modbus Exception Codes

The list of exceptions that the MBSerial DAServer expects follows what is specified in the document "MODBUS Application Protocol Specification." The accompanying table shows these exceptions and their explanations.

Exception Code (Hex)	Name	Explanation
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the slave. This may be because the function code is only applicable to newer controllers, and was not implemented in the unit selected. It could also indicate that the slave is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the slave. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed, a request with offset 96 and length 5 will generate exception 02.

Exception Code (Hex)	Name	Explanation
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for the slave. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04	ILLEGAL RESPONSE LENGTH	Indicates that the request as framed would generate a response whose size exceeds the available MODBUS data size. Used only by functions generating a multi-part response, such as functions 20 and 21.
05	ACKNOWLEDGE	Specialized use in conjunction with programming commands.
06	SLAVE DEVICE BUSY	Specialized use in conjunction with programming commands.
07	NEGATIVE ACKNOWLEDGE	Specialized use in conjunction with programming commands.
08	MEMORY PARITY ERROR	Specialized use in conjunction with function codes 20 and 21, to indicate that the extended file area failed to pass a consistency check.
0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate a Modbus Plus PATH to use to process the request. It usually means the gateway is misconfigured.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. It usually means the device is not present on the network.

Serial Communications

The MBSerial DAServer supports both RS232 and RS485 multi-drop connections and modem connection (Data Modem and Radio Modem).

Data Types

The MBSerial DAServer supports the following data types:

- Boolean
- 16-bit signed integer
- 16-bit unsigned integer
- 32-bit signed integer
- 32-bit unsigned integer
- ASCII string
- 32-bit single precision floating point
- 32-bit double-precision Decimal (BCD)

Note For detailed information on these data types, see Data and Register Types in Chapter 3.

Message Frame Checking

There are two types of message frame checking:

- RTU mode
- ASCII mode

In RTU mode, Cyclic Redundancy Checking (CRC) is used. The CRC field checks the contents of the entire message. It is applied regardless of any parity checking method used.

In ASCII mode, Longitudinal Redundancy Checking (LRC) is used. LRC checks the contents of the message, exclusive of beginning 'colon' and ending CRLF pair. It is applied regardless of any parity checking method used.

Index