OPC Factory Server V3.60
User Manual

11/2015

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents 📖

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

This manual describes the software installation of the OPC Factory Server (OFS) product.

Validity Note

The update of this documentation takes the latest version of OFS into account.

Related Documents

Title of Documentation	Reference Number
System Time Stamping - User Guide	EIO0000001217 (Eng), EIO0000001707 (Fre), EIO0000001708 (Ger), EIO0000001710 (Ita), EIO0000001709 (Spa), EIO0000001711 (Chs)
Vijeo Citect User Guide	Supplied with Vijeo Citect installation files and installed with Vijeo Citect.
Vijeo Citect Help	Installed with Vijeo Citect.
Applicative Time Stamping with Unity Pro, User Guide	EIO0000001268 (Eng), EIO0000001702 (Fre), EIO0000001703 (Ger), EIO0000001705 (Ita), EIO0000001704 (Spa), EIO0000001706 (Chs)
Modicon M580 System Planning Guide	HRB62666 (Eng), HRB65318 (Fre), HRB65319 (Ger), HRB65320 (Ita), HRB65321 (Spa), HRB65322 (Chs)

Title of Documentation	Reference Number
Modicon M580 Hardware Reference Manual	EIO0000001578 (Eng), EIO0000001579 (Fre), EIO0000001580 (Ger), EIO0000001582 (Ita), EIO0000001581 (Spa), EIO0000001583 (Chs)
Unity Pro System Bits and Words Reference Manual	EIO0000002135 (Eng), EIO0000002136 (Fre), EIO0000002137 (Ger), EIO0000002138 (Ita), EIO0000002139 (Spa), EIO0000002140 (Chs)
Unity Pro-Operating Modes	33003101 (Eng), 33003102 (Fre), 33003103 (Ger), 33003696 (Ita), 33003104 (Spa), 33003697 (Chs)

You can download these technical publications and other technical information from our website at http://download.schneider-electric.com

Part I
Introduction to the OFS product

Chapter 1
Uses of the OFS product

Aim of this Chapter

The aim of this chapter is to introduce to you the uses of the OFS (OPC Factory Server) product.

What Is in This Chapter?

This chapter contains the following topics:

Introducing the OFS Server

General

The OFS product (OPC Factory Server) is a multi-controller data server which is able to communicate with PLCs of the M580, Unity Momentum, TSX/PCX Premium, Quantum, M340, TSX Compact, TSX Micro, TSX Momentum, TSX Series 7, and TSX S1000 families to supply the OPC clients with data.

The OFS product provides client applications with a group of services (methods) for accessing to variables of a target PLC.

OFS is compatible with versions OPC 1.0 A and OPC 2.0. It will function with an OPC client up to version 2.0a and with two types of OPC software, i.e.:

● Supervisory software (see distributor offer): The OFS server plays the role of a driver providing communication with all devices supported by Schneider Electric SA,
● Custom developed supervisory software, using either the OLE Automation interface or the OLE Custom interface.

NOTE: Knowledge of one of the following languages is required when creating a client application for the OFS, in particular for OLE Automation, OLE Custom programming and exception management:

● Microsoft Visual Basic, version 6.0 SP3 or later,
● Microsoft Visual C++, version 6.0 SP3 or later,
● Microsoft VBA in Excel, version 8.0 (Office 97) or later,
● Microsoft Visual C#,

The illustration shows an OFS interface:
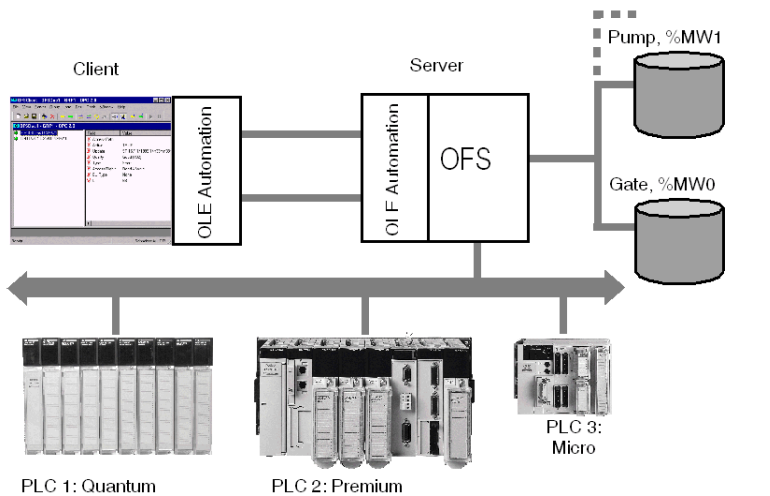
The OFS server provides the interface between Schneider Electric PLCs and one or more client applications. These applications are used to view and/or edit the data values of target devices.

The main features of the OFS product are the following:

- multi-device,
- multiple communication protocols,
- multi-client,
- access to devices and variables by address or symbol,
- access to the server in local or remote mode,
- use of a notification mechanism, enabling values to be sent to the client only when they change state. The server offers two modes for exchanges with the PLC: The default "classic" (polling) mode, or the "Push Data" mode where data are sent at the initiative of the PLC. This mode is recommended when changes of state are infrequent,
- automatic determining of the size of network requests according to the devices,
- availability of services via both the OLE Automation and OLE Custom interfaces,
- compatibility with both OPC DA (Data Access) Standard Version 1.0 A and 2.0.

The OFS server offers the following services:

- reading and writing of variables in one or more PLCs present on one or more different networks,
- a user-friendly configuration tool, giving a greater understanding of the parameters needed for the server to function efficiently, and a tool enabling parameters to be modified online, in order to maximize utilization flexibility,
- the possibility of using a list of symbols for the PLC application,
- a browser interface enabling the user to obtain a graphic understanding of the accessible devices and their associated symbols,
- a list of 'specific' items depending on the devices that enable specific functions to be executed: status and starting/stopping the PLC, alarm supervisory function.

Communication with PLCs

At a Glance

The OFS server operates on the following networks:

- Modbus Serial (RTU),
- Modbus TCP IP (IP or X-Way addressing),
- Modbus Plus,
- Uni-Telway,
- Fipway,
- Ethway,
- ISAway
- PCIway
- USB
- USB Fip

The OFS server is compatible with the Nano on Uni-Telway only, with the following restrictions:

- read operations only,
- access to a single word or x bits within 16 consecutive bits.

The tables below describes the OFS compatibility with devices in the Schneider Electric SA range and the different networks:

	PREMIUM	MICRO	Series 7	Series 1000
Ethway	TSX ETY 110• (Ethway)		TSX ETH107 TSX ETH 200	ETH030•
Modbus TCP/IP	TSX ETY110• (TCP/IP) TSX ETY410• (TCP/IP) Built-in channel TSX ETY510• (TCP/IP)	TSX ETZ410 TSX ETZ510		
Uni-Telway	Built-in channel TSX SCP11•	Built-in channel TSX SCP11•	TSX SCM22•	
USB Fip Fipway	TSX FPP20	PCMCIA TSX FPP20	TSX P•7455 TSX FPP20	
ISAway	ISA Bus			
PCIway	PCI bus			
Modbus	TSX SCP11• (Not supported on Unity Premium)	TER CPU port	TSX SCM22•	JB cards•
Modbus Plus	TSX MBP100	TSX MBP100		
USB	Built-in channel			

	QUANTUM	MOMENTUM	COMPACT	M340
Modbus TCP/IP	140NOE 771•• Built-in channel	171CCC96030 171CCC98030		BMX NOE0100 BMX NOE0100 Built-in channel
Modbus	Built-in channel	171CCC760•• 171CCC780••	Built-in channel	Built-in channel
Modbus Plus	Built-in channel 140NOM211••		Built-in channel	

	M580	M80	Unity Momentum
USB	Built-in channel	Built-in channel	Built-in channel
Modbus TCP/IP	Built-in channel	Built-in channel	Built-in channel
Modbus	BMX NOM0200	Built-in channel	Built-in channel

Different Access Modes for Server or Simulator

Description

There are three access modes for the OFS server:

- A purely local mode,
- A mode via a classic "DCOM" network,
- A mode via a "web" http interface.

Local access

The client application and the OFS server are on the same extension.
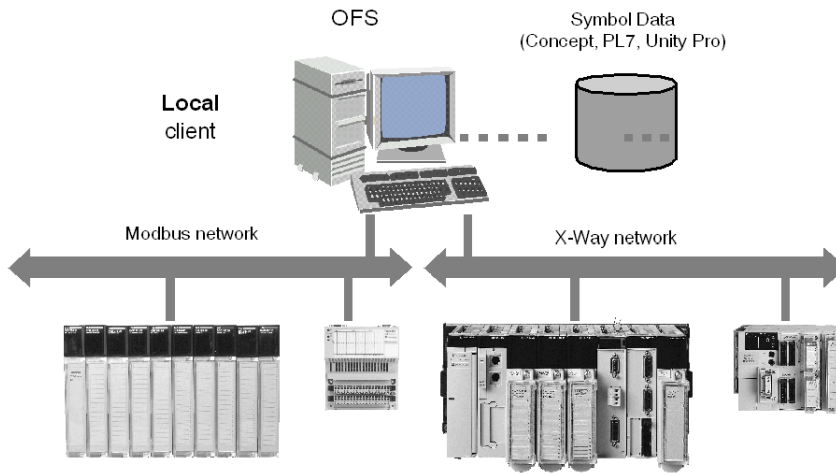
Remote access by DCOM

Remote access on the Internet via DCOM
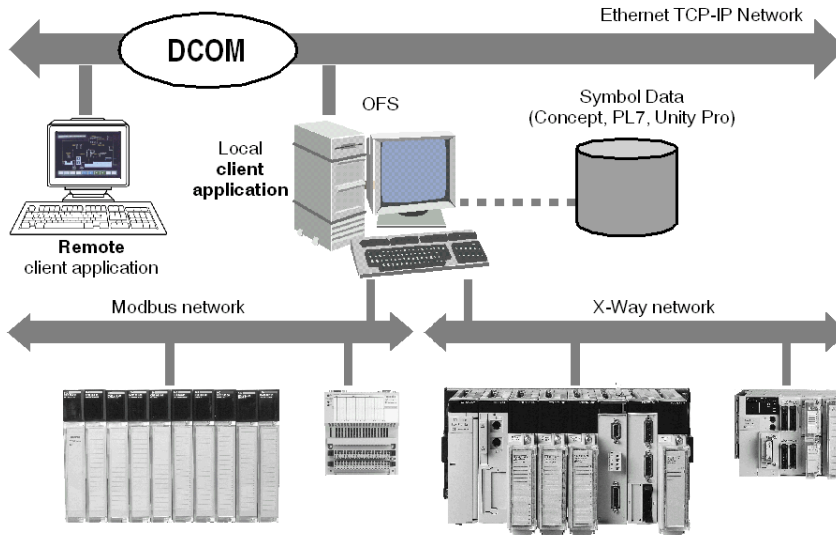
The client application and the OFS server are on separate extensions, connected via the Microsoft TCP-IP network:

NOTE: DCOM *(see page 52)* must be configured correctly before launching remote operation.

Remote Access by IIS

Remote access on the Internet via IIS (Internet Information Services)

The site server and the OFS server are on the same extension.

The site server and the client application are on separate extensions, communicating via Internet:

Site Server and OFS
extension

WWW

Internet
Firewall

Remote
client application

IIS

Symbol Data
(Concept, PL7, Unity Pro)

Industrial network

PLCs
Programmable
Industrial
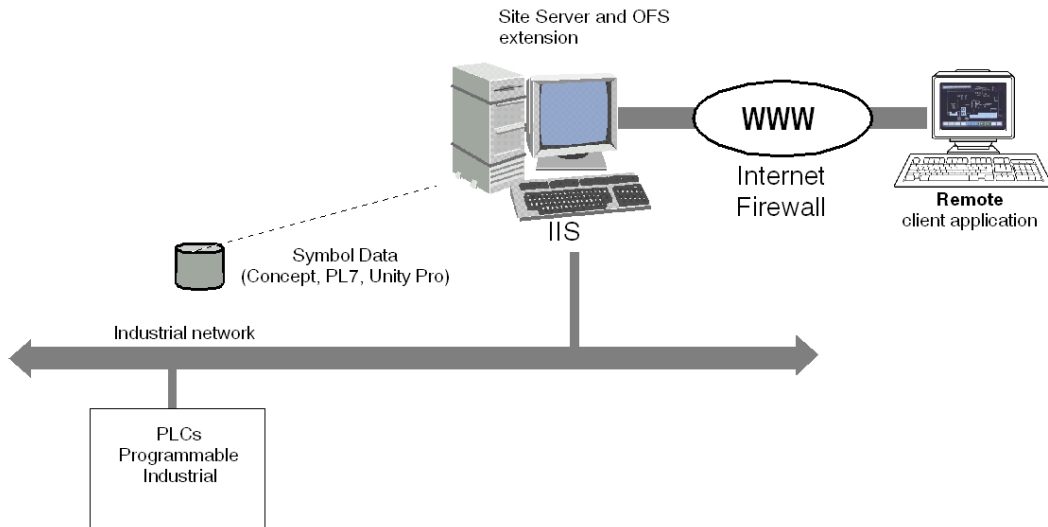
NOTE: IIS *(see page 58)* must be configured correctly before launching the remote operation.

Software components and terminology

At a Glance

To comply with the OPC foundation standard, the OPC Factory Server program includes a set of specific software components.

.NET

.NET is a set of Microsoft software used to connect information, systems, and devices. It enables a high level of software integration through the use of Web services which connect both to one another and to other, wider applications over the Internet.

The .NET platform:

- makes it possible to have all devices work together and to have user information updated automatically and synchronized across all devices,
- increases the capacity for Web sites to interact, enabling a better use of XML,
- enables the creation of reusable modules, thereby increasing productivity and reducing the number of programming inconsistencies,
- centralizes data storage, increasing the effectiveness and simplicity of access to information, and enabling the synchronization of information among users and devices.

Framework.NET

The .NET framework comprises 2 main parts:

- the common execution language with a library of unified, hierarchical classes. It includes an advance to Active Server Pages (ASP .NET), an environment for building intelligent client applications (Windows Forms),
- a data access subsystem (ADO .NET).

Web service

Web services are applications located on the server side. They are queried by Web or thick Client applications located on the network. The entire system communicates via standardized messages in XML format. Web services enable to delocalize processing, among other things.

Access of a .NET client

Description

An OPC .NET client can access data on the OFS server via an intranet in a .NET environment.

Illustration:
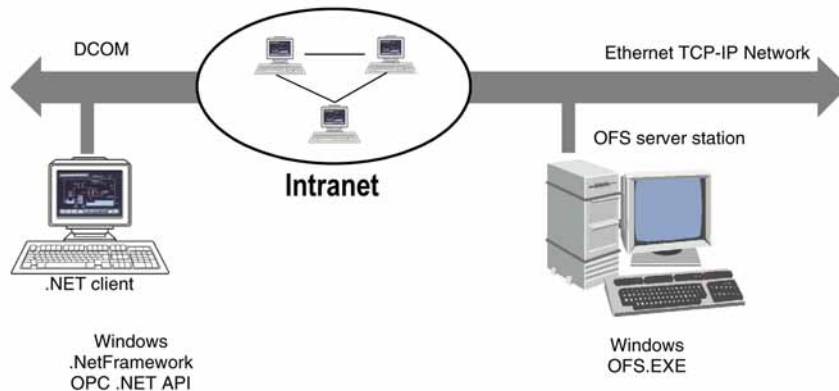
NOTE: DCOM *(see page 65)* must be configured correctly before launching remote operation.

NOTE: Communication between the OPC .NET client and the OFS server is managed by the DCOM layer (or COM in a local configuration). The standard OPC DA protocol is used for this communication.

Access for a SOAP/XML Client

Description

A SOAP/XML Client can access data on the OFS server via the SOAP/XML protocol while respecting the OPC XML DA specification of the OPC foundation.

NOTE: DCOM *(see page 65)* and IIS *(see page 58)* must be configured correctly before launching the operation.

SOAP/XML Client Via the Internet

This architecture illustrates a possible Internet configuration for a SOAP/XML Client:
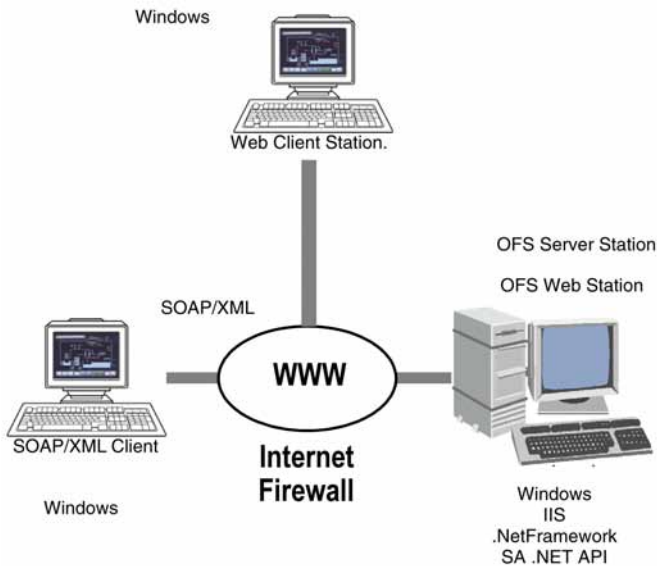
Introducing the OPC UA Wrapper

General

The OPC UA wrapper communicates with an OPC Factory server (works as a wrapper for OFS server) to supply the OPC UA clients with data. Therefore, there is an indirect connection between OPC UA clients and OFS version 3.50 server via OPC UA wrapper. The reference of the OPC UA security model for administrators is on http://www.opcfoundation.org/.

Final Result

After a successful installation, the OFS setup creates links to the each `.exe` in the location **Start → All programs → Schneider Electric → SoCollaborative → Ofs**.

This figure shows the links for **OPC UA Client**, **OPC UA Configuration Tool** and **OPC UA Wrapper**:
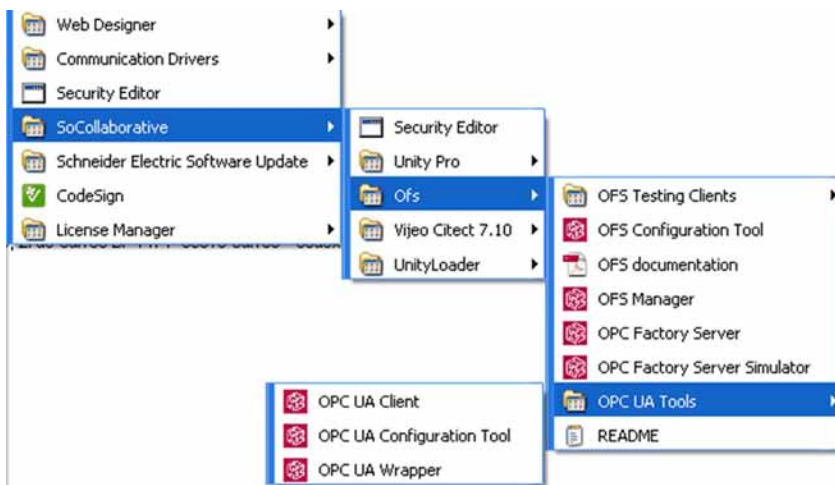
Part II
Installing the OFS product

Aim of this section

The aim of this section is to introduce you to the procedures for installing the product.

What Is in This Part?

This part contains the following chapters:

Chapter 2
Contents of the OPC Factory Server Product

OFS Contents

Contents of the product

OFS and OPC are delivered on a DVD that contains:
- installation instructions
- drivers
- OFS server
- OFS manager
- OFS configuration tool
- OPC UA wrapper
- System Time Stamping User Guide
- OFS TimeStamp Helper
- OPC Factory Server V3.60 User Manual (English/French/German languages)
- sample symbol tables and sample applications
- two OPC test clients (Win32 and .NET)
- a Web client (for accessing the page view, data editor and server status)

NOTE: The OFS product does not contain cables for communication between the PC and PLC.

Chapter 3
Product installation procedure

Aim of this Chapter

The aim of this section is to guide users as they use the product.

What Is in This Chapter?

This chapter contains the following topics:

Installing the OFS Product

Install Prerequisite

The Microsoft .Net framework v3.5 SP1 must be installed. If not, the OFS setup stops installing.

Preparing to Install

In Windows 7, Windows 8, Windows 8.1, Windows 10, Windows Server 2008 R2,
Windows Server 2012 and Windows Server 2012 R2, you need administrator rights to install OFS.

NOTE: If the Vijeo Citect product is already installed on the machine with its own OFS release, do not install OFS as a standalone product. Contact the customer support to get the adapted version.

The following components can be installed:
● OPC Data Access Station
● OPC Data Access Remote Client
● .Net Interface
● OPC XML server
● OPC UA wrapper
● Web client JVM checking

The OPC Data Access Station install option is used when a machine supports the OFS server and/or the OPC client(s).

The OPC Data Access Remote Client install option is used when a machine receives one or more OPC client(s) and remotely accesses the OFS server via DCOM.

The .Net Interface install option provides .Net OPC test client or .Net OPC XMLDA test client.

The OPC XML server install option provides the Web services (SOAP/OPC XMLDA) for the OPC function as well as the Schneider Electric Web site used for diagnostics and access to OFS server data.

The Web client JVM checking option checks for the compatibility level of the JVM to ensure that the OFS Web site is accessible on the machine via the Internet using the OPC XML standard.

NOTE: The OPC Data Access Remote Client, .Net Interface, OPC XML server and Web client JVM Installation options may be installed as many times as necessary on as many different machines. No product registration is required.

NOTE: If the following message appears when installing OFS, click **Continue**.
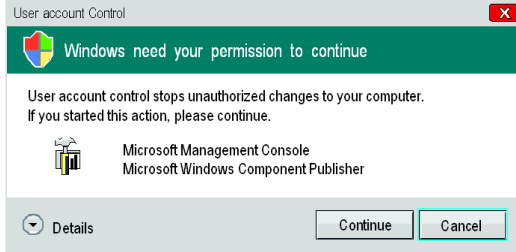
Launching the Installation

To install the OFS product, you have to follow this procedure:

Step	Action
1	Put the DVD in the drive. The installation automatically starts. Follow the on-screen messages.
2	In the **Customer Information window**, enter the part number and the serial number provided on the DVD box. *Result*: The product install selection screen appears:

Step	Action
3	Among the available features and depending on the customer information, select the ones you want to install and click **Next.** **Note**: You can display the description of a feature as well as its components by putting the focus on it. **Important**: The OPC Data Access Station and the OPC Data Access Remote Client cannot be both installed on the same machine. These options are exclusive.

OPC Data Access Station

Installation Options

The installation offers the following options:

- OPC DA Server
  - **OFS Server**: Multi-controller data server which is compatible with versions OPC 1.0 and OPC 2.0, allowing you to communicate with the Schneider PLCs in order to supply one or more OPC client applications.
  - **OFS Server Simulator**: Enables to test the client OPC application without any PLC being present. It provides a simple animation of all created variables and is identical to the actual server.
  - **OPC DA Server Manager** (*OFSManager.exe*): The OFS manager *(see page 143)* is a utility application used to access debugging information from the OFS server, either locally or remotely. It is also used to request the server to execute certains actions on line (create a new alias, reload symbol tables, etc.).
  - **Error Encoder Tool** *(see page 393)* (*scoder.exe*) : Utility allowing you to decode the detected error code returned by OLE, OPC and the OFS Server.
  - **OFS registration tool:** Allows you to register the server after its installation.
  - **OPC proxy DLLs**: Updates your registry base and some system files (*OPCproxy.dll*&*OPCcommon.dll*).
  - **OPC Automation interface 1.0 and 2.0**: This option installs the files required to use the Automation Interface 1.0 and 2.0 of the OFS Server.

- OPC DA Sample Client (*OFSClient.exe*): The Sample Client *(see page 148)* is used to access and test any kind of OPC server. It is not specific to the OFS Server.

- OFS Configuration tool: Allows you to define the devices and their properties accessible through the OFS Server and the global settings of the OFS Server *(see page 71)*.

- OFS User Documentation: Allows you to get access to the online documentation.

NOTE:
- Once the server is installed, you can use it in evaluation mode for 21 days. During this period, you have to register your version of OFS, otherwise the server will stop at the end of the evaluation period.
  You may perform the subscription when the installation ends, or at any time during the evaluation period.
- In DEMO mode, all server functions are available, but the product use may not exceed 3 days, the server should then be stopped and restarted.
- In particular, avoid spaces in file names.

OPC Data Access Remote Client

Description

To perform the installation, you have just to follow the on-screen messages.

The installation offers the following components:

- **OFS remote server registration and OPC proxy DLLs**: An update of your registry base and of some system files will be performed (OPCproxy.dll & OPCcommon.dll),
- **OFS server test client**: The Test Client *(see page 148)* (OFSClient.exe) is used to access and test any kind of OPC server. It is not specific to the OFS server,
- **OFS Manager**: The OFS Manager *(see page 143)* (OFSManager.exe) is a utility application used to access debugging information from the OFS server, either locally or remotely. It is also used to request the server to execute certain actions on-line (create new alias, reload symbol tables, etc.),
- **OPC Automation interface 1.0 and 2.0** : This option installs the files required to use the Automation Interface 1.0 and 2.0 of the OFS server.

To operate correctly, the remote extension must have been the subject of a DCOM configuration *(see page 52)* on both the remote extension and the server extension.

Installation of a .NET Interface

Description

Follow the messages on the screen to perform the installation. The installation program offers the following option:

.NET Sample Client: this is an OFS test utility client, operating in a .NET environment.

Installation of an OPC XML Server

Description

This installation option is only available with the 'Large' version of the OFS product. If this has not already been done, it is preferable to first install IIS on the machine using the operating system installation DVD. Then follow the messages on the screen to perform the installation. The first message is the verification of the IIS service:

- **IIS (verification)**: if the IIS service is not installed on the machine, the installation is suspended. You must then install IIS and restart the Web extension installation procedure *(see page 58)*.

The installation program then proposes:

- **OFS Web Site**: provides data via tables to the Web Clients such as the editor, data viewer, and the server status pages.
- **OPC XMLDA 1.01 Web services**: provide the Web services specified by the OPC Foundation in version 1.01 based on the standardized XML data format, protocol, and exchanges between clients and servers.

At this stage of installation, you must configure *(see page 58)* IIS according to the site's security and access.

NOTE: The OFS server must also be installed on the station that hosts the OFS Web site.

Web Client JVM checking

Description

This installation checking is only available with the 'Large' version of the OFS product:

- **JVM install policy (verification)**: An information message appears displaying the installation status of the JVM.
- **Internet Explorer (verification)**: if no Internet Explorer is installed on the machine, or if the version is too old (earlier than IE 5.1), the installation program asks the user to update the machine to the required level.

Driver Installation

Description

The OFS server can use drivers already installed on your machine if they are not too old. The compatibility following table indicates the minimum version that you should have installed to be sure that OFS operates properly. Use of OFS with older versions is not supported and not guaranteed.

NOTE: It is compulsory to install the **Driver Manager**, except for use with TCP/IP in direct IP addressing.

NOTE: It is recommended to update drivers (drivers setup are located on OFS DVD communication drivers folder). The Uni-Telway, Modbus Serial, and USB drivers are automatically updated during OFS setup. They can be also installed through OFS setup.

The table shows the compatibility table:

Drivers	Minimum version depending on operating system			
	Windows 7 Windows Server 2008 R2 32 bits	Windows 7 Windows Server 2008 R2 64 bits	Windows 8 32 bits Windows 8.1 32 bits	Windows 8 Windows 8.1 Windows Server 2012 Windows Server 2012 R2 64 bits
Uni-Telway	2.5	3.5	2.5	3.5
FIPway FPC10	Not supported	Not supported	Not supported	Not supported
FIPway PCMCIA	Not supported	Not supported	Not supported	Not supported
ISAway	Not supported	Not supported	Not supported	Not supported
Ethway	Not supported	Not supported	Not supported	Not supported
X-Way / TCP/IP	2.6	3.6	2.6	3.6
PCIway PCI 57	2.1	3.1 (Windows Server 2008 R2 64 bits only)	Not supported	Not supported
USB HE terminal port PCX 57	2.6	3.6	2.6	3.6
Modbus Serial	2.9	3.9	2.9	3.9
USB Fip	2.3	3.4	Not supported	Not supported

Installing the OPC UA Wrapper Product

Installation

Follow this procedure to install the OPC UA wrapper product:
- Select the OPC UA wrapper checkbox in the options at the time of OFS installation. It installs the OPC UA wrapper.
- This figure shows how to install the OPC UA wrapper:
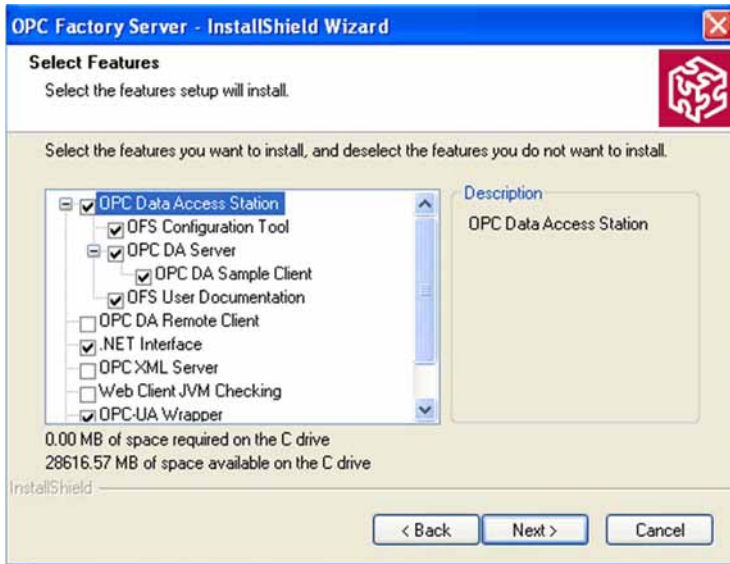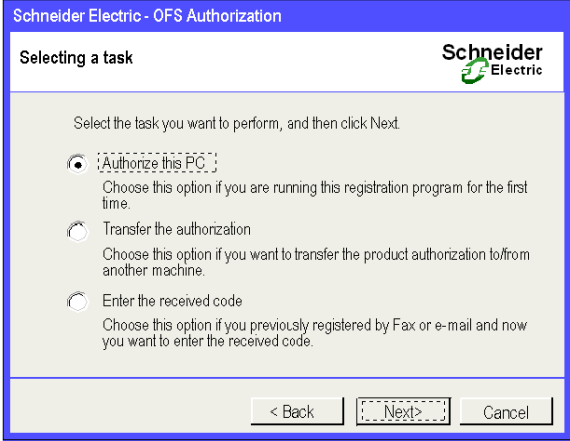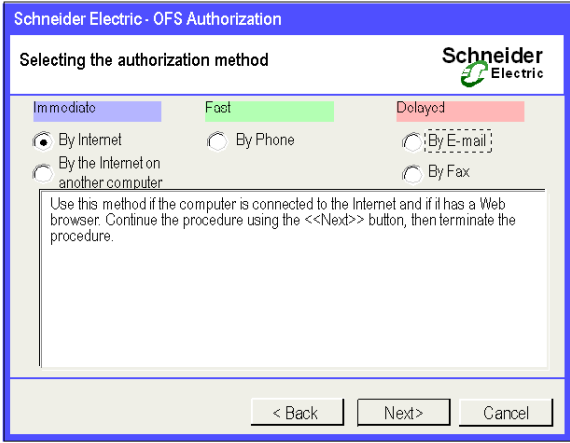
OFS Authorization
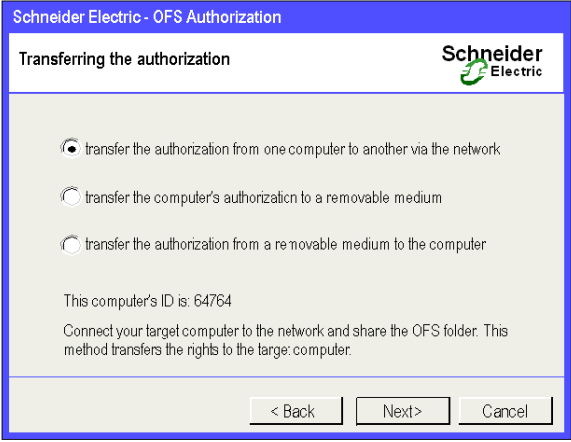
Authorize OFS

The following table describes the PC authorization procedure to use OFS without any restrictions:

Step	Action
1	The window below is displayed when the SAOFS.exe file is run:

Step	Action
2	Click **Next**, a new window is displayed with 3 choices: ● the PC's authorization to know the various ways to make an authorization request ● the transfer of the authorization to or from a computer storage device ● the entry of the code after receiving it The window below displays the 3 options: Click **Next**.

Step	Action
3	By choosing the **Authorize this PC** option in step 2, a new window is displayed. This window offers 5 ways to request the authorization: Click one of the methods according to your requirements (**Immediate**, **Fast**, or with a **Delayed**) then click **Next** and follow the remaining procedures.

Step	Action
4	By choosing the **Transfer the authorization** option in step 2, a new window is displayed. The window below offers 3 types of authorization transfer: ● **transfer the PC's authorization to a PC on the network** ● **transfer the PC's authorization to a removable medium** ● **transfer the PC's authorization from a removable medium to this PC** **Note**: For an authorization transfer on the network, the **OFS** installation directory of the PC receiving the authorization should be shared with the **OFS** share name and with the Full Control permission rights, during the transfer operation. **Note**: The removable medium must not be write-protected.

Schneider Electric - OFS Authorization

Transferring the authorization   Schneider
                                                   Electric

⦿ transfer the authorization from one computer to another via the network

○ transfer the computer's authorization to a removable medium

○ transfer the authorization from a removable medium to the computer

This computer's ID is: 64764

Connect your target computer to the network and share the OFS folder. This method transfers the rights to the target computer.

< Back   Next>   Cancel

Step	Action
5	By choosing the **Enter the received code** option in step 2, a new window is displayed. The window below prompts you to enter the authorization code: Schneider Electric - OFS Authorization Entering the received code Schneider Electric Serial number: 0 Reference: ID code: 238760719 Computer ID: 64764 Authorization code received: [ ] Print < Back Next> Cancel Enter the code then click **Next**. By clicking **Print**, the displayed window is printed on the default printer.

Part III
Workstation setup

Aim of this section

The aim of this section is to introduce you to the machine configuration.

What Is in This Part?

This part contains the following chapters:

Chapter 4
Workstation Configuration

Aim of this Chapter

The aim of this chapter is to introduce the workstation configuration for operation in remote intranet or Internet mode.

What Is in This Chapter?

This chapter contains the following sections:

Section 4.1
COM/DCOM Station Configuration

DCOM Configuration

Description

The OFS server can operate in local mode (the server and the OPC client are located on the same machine) or in remote mode (the OPC client and the server are on different machines connected by DCOM generally via Ethernet TCP-IP).

The remote execution mode requires an additional adjustment using the DCOMCnfg.exe tool provided with the DCOM package.

The server and the client station should be configured appropriately.

Configuration

The configuration parameters must be defined while logged on to the machine with an account having the necessary administration rights to access and start up the server.

This chapter describes a possible configuration that allows the machine to be accessed through DCOM.

Services configuration:

From the **Start** menu, choose **Run** and run **Services.msc**.

The following services must have the following **Startup type**:

Service	Startup Type
Distributed Transaction Coordinator	Manual
Remote Procedure Call (RPC)	Automatic
Security Accounts Manager	Automatic

Double-click a service to edit and change the **Startup Type** if needed.

DCOM machine default configuration:

Step	Action
1	From the **Start** menu, choose **Run** and run **DCOMCNFG**.
2	Expand **Console Root/Component Services/Computers/My Computer**and right-click **My computer** to open the **My Computer Properties** dialog.
3	Click the **Default Properties** tab.

4	The following parameters must be set: • **Enable Distributed Com on this computer** must be checked, • **Default Authentication Level** is set to **Connect**, • **Default Impersonation Level** is set to **Identify**.
5	Click the **COM Security** tab.
6	Click the **Edit Defaults** in **Access Permissions**.
7	Click **Add**, enter **Everyone** local account then click **OK**. Check that permissions for **Everyone** are: • Local access: **Allow** checked. • Remote access: **Allow** checked.
8	If OFS is configured as an NT service *(see page 67)*, click **Add**, enter **OFService Account**, then click OK and verify that permissions for **OFService Account** are: • Local Access: **Allow** checked, • Remote Access: **Allow** checked.
9	Click **OK** to close the dialog box window.
10	Click **Edit Defaults** in **Launch and Activation Permissions**.
11	Click **Add**, enter **Everyone** local account then click **OK**. Check that permissions for **Everyone** are: • Local Launch: **Allow** checked. • Remote Launch: **Allow** checked. • Local Activation: **Allow** checked. • Remote Activation: **Allow** checked.
12	If OFS is configured as an NT service *(see page 67)*, click Add, enter **OFService Account**, then click OK and verify that permissions for **OFService Account** are: • Local Launch: **Allow** checked, • Remote Launch: **Allow** checked, • Local Activation: **Allow** checked, • Remote Activation: **Allow** checked.
13	Click **OK** to close the dialog box window.
14	Click **Edit Limits** in **Access Permissions**.
15	Click **Add**, enter **Everyone** local account then click OK. Check that permissions for **Everyone** are: • Local Access: **Allow** checked, • Remote Access: **Allow** checked. If OFS is configured as an NT service *(see page 67)*, click Add, enter **OFService Account**, then click OK and verify that permissions for **OFService Account** are: • Local Access: **Allow** checked. • Remote Access: **Allow** checked.
16	Click **OK** to close the dialog box window.
17	Click **Edit Limits** in **Launch and Activation Permissions**.

18	Click **Add**, enter **Everyone** local account then click OK. Check that permissions for **Everyone** are: ● Local Launch: **Allow** checked, ● Remote Launch: **Allow** checked, ● Local Activation: **Allow** checked, ● Remote Activation: **Allow** checked. If OFS is configured as an NT service *(see page 67)*, click add, enter **OFService Account**, then click OK and verify that permissions for **OFService Account** are: ● Local Launch: **Allow** checked, ● Remote Launch: **Allow** checked, ● Local Activation: **Allow** checked, ● Remote Activation: **Allow** checked.
19	Click **OK** to close the dialog box window.
20	Click **OK** in the **My Computer Properties** dialog to close it.

DCOM OFS configuration:

Step	Action
1	From the **Start** menu, choose **Run** and run **DCOMCNFG**.
2	Expand **Console Root/Component Services/Computers/My Computer/DCOM Config/** and right-click **Schneider-Aut OPC Factory Server** to display the properties.
3	Click the **Location** tab, the option **Run application on this computer** should be selected.
4	Click the **Identity** tab: ● If OFS is configured as an NT service *(see page 67)*, the option **This user** should be selected. Enter the name and password of the account referenced as **OFService Account**. ● If OFS is not configured as an NT service, the option **The interactive user** should be selected.
5	Click the **General** tab, the authentication level should be set to **Use Default**.
6	Click the **Security** tab ● the **Launch and activation Permissions** should be set to **Use Default**, ● the **Access Permissions** should be set to **Use Default**.
7	Close the **Properties** dialog with **OK**. **NOTE:** It is recommended to restart the system.

NOTE: If the client and the server do not belong to the same Windows domain or if there is no Windows domain, remember that identical users with identical passwords must be created on both machines (note case sensitivity).

Client:

The configuration parameters must be defined while logged on to the machine when you have an account with the necessary administration rights to access and start up the client.

Step	Action
1	• Run DCOMCnfg.exe from the c:\Windows\System32 folder and on the icon. Right-click the icon Root Console, Component Services, Computers, WorkStation to display the properties by right-clicking or, • Click **Control Panel**, **Administrative Tools**, **Component Services**. In the window that pops up, click **Component Services**, **Computers**. Right-click the icon **My Computer** to display the properties.
2	In the **Default Properties** tab, check that: • the **Enable Distributes COM (DCOM) on this computer** option is selected, • the field **Default Authentication Level** is set to **Connect**, • the field **Default Impersonation Level** is set to **Identify**.
3	In the **COM Security** tab, modify the **Default Access Permissions** list in order to make sure that the users SYSTEM, INTERACTIVE, NETWORK and EVERYONE are present. This last setting is only necessary to allow the server to send back notifications to the client machine.

Configuring Workgroup and Domain DCOM Access

These additional rules can be applied if OFS is installed on a Workgroup system (vs Domain) or, if installed on a Domain system, OFS must be accessed by a Workgroup system through DCOM.

Step	Action
1	From the **Start** menu, choose **Run** and run **secpol.msc**.
2	**Expand Security Settings\ Local Policies\ Security Options**.
3	Double-click the following Policy item **DCOM: Machine Access Restrictions in Security Descriptor Definition Language (SDDL) syntax**.
4	Click **Edit Security** button to open the **Access Permission** dialog.
5	Click **Add**, enter **Everyone** local account then click **OK**.
6	In the **Access Permission** dialog, check that permissions for **everyone** are: • Local Access: **Allow** checked, • Remote Access: **Allow** checked.
7	Click **OK** to close the dialog box window.
8	Click **OK** to close the **Policy Settings** dialog box.
9	Double-click the following Policy item **DCOM: Machine Launch Restrictions in Security Descriptor Definition Language (SDDL) syntax**.
10	Click **Edit Security** button to open the **Launch and Activation Permission** dialog.
11	Click **Add**, enter **Everyone** local account then click **OK**.

Step	Action
12	In the **Launch and Activation Permission** dialog, check that permissions for **Everyone** are: ● Local Launch: **Allow** checked, ● Remote Launch: **Allow** checked, ● Local Activation: **Allow** checked, ● Remote Activation: **Allow** checked.
13	Click **OK** to close the dialog box window.
14	Click **OK** to close the **Policy Settings** dialog box.
15	Double-click the following Policy item **Network access: Let Everyone permissions apply to anonymous users**.
16	Set the value to **Enabled**.
17	Click **OK** to close the **Policy Settings** dialog box.

Section 4.2
Configuration of Internet Information Services (IIS) Stations

Aim of this Section

This section describes the authorization process for connecting to a Web site with IIS and ASP.NET, as well as the vocabulary used.

What Is in This Section?

This section contains the following topics:
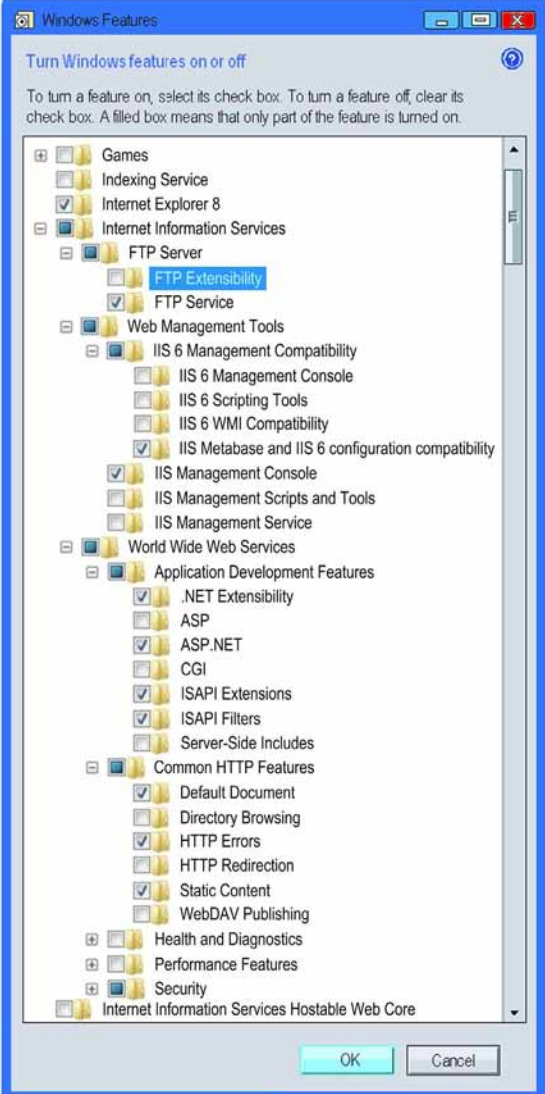
Topic	Page
Configuring the IIS Component	58
COM/DCOM Configuration	65

Configuring the IIS Component

Installing IIS

Follow this procedure:

Step	Action
1	In the **Start** menu, select **Control Panel**.
2	Select **Programs and Features**, then **Turn Windows features on or off** at the top left of the window. **Result**: The **Turn Windows features on or off** window appears:

Step	Action
3	Select the following item:
4	Confirm your choices by clicking **OK**.

Enabling ISAPI Extension

before to configuring ISS, you must enable the ASAPI extension for ASP.NET 3.5:
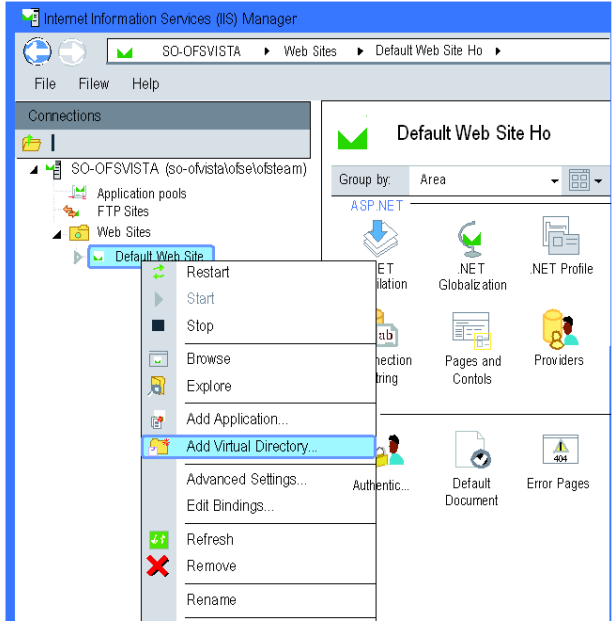
Step	Action
1	Click **Start → Run**. **Result**: The **Run** dialog box appears.
2	Type `%windir%\Microsoft.NET\Framework\v2.0.50727\aspnet_regiis -i -enable` in **Open** field. **NOTE:** Where `%windir%` is the environment variable corresponding to the installation folder of Windows.
3	Click **OK**.
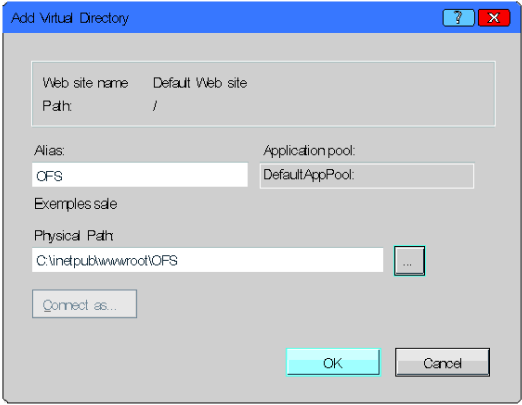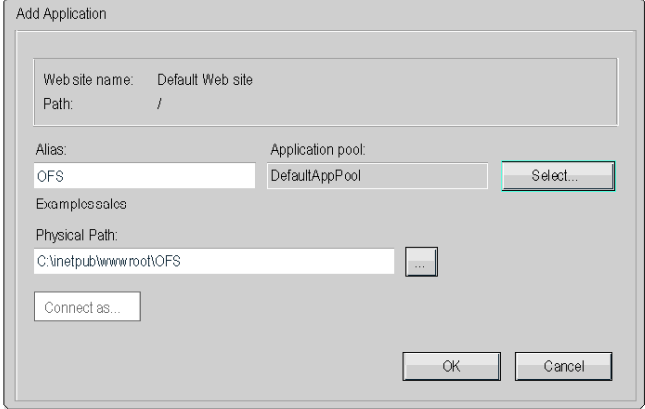
NOTE: The above command line must also be executed when updating OFS V3.x to OFS V3.60 with OPC XML server option installed.
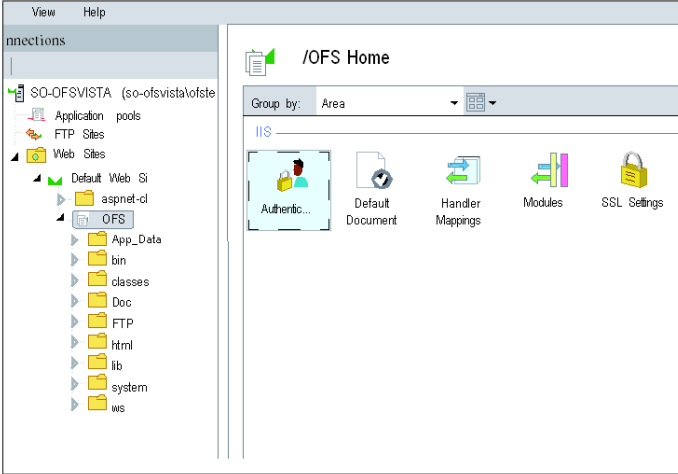
Configuring IIS
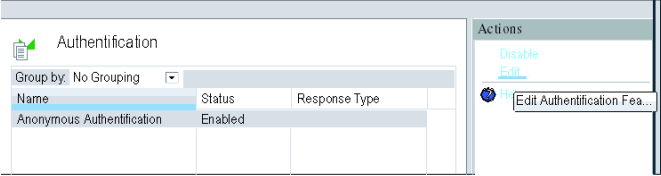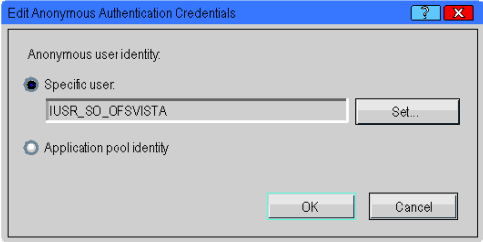
The following table describes configuration of IIS.

Step	Action
1	Create the *<root>\OFS*. **Example:** *c:\inetpub\wwwroot\OFS*.
2	In the **Start** menu, select **Control Panel**.

Step	Action
3	Click **Administrative Tools** then select **Internet Information Services Manager (IIS)**. **Result**: The **IIS Manager** window appears:

Step	Action
4	In the navigation panel, expand **Web Sites**. Right-click on **Default Web Site** and select **Add Virtual Directory...** **Result**: The following window appears: *Add Virtual Directory* Web site name Default Web site Path: / Alias: Application pool: OFS DefaultAppPool: Exemples sale Physical Path: C:\inetpub\wwwroot\OFS ... Connect as... OK Cancel ● Type **OFS** in the **Alias** field. ● Click **...** to specify the **Physical Path**. It must lead to the directory defined in step 1. Click **OK**.
5	In the navigation panel of the IIS Manager window, expand **Default Web Site**. Right click the **OFS** folder and select **Convert to application**.
6	Right-click the **OFS** folder again and select **Add Application**. **Result**: The following window appears: *Add Application* Web site name: Default Web site Path: / Alias: Application pool: OFS DefaultAppPool Select... Exemples sales Physical Path: C:\inetpub\wwwroot\OFS ... Connect as... OK Cancel

Step	Action
7	Click **Select...**. **Result**: A dialog box appears: On Windows 7 and Windows Server 2008 R2: **a** Select **DefaultAppPool** and click **OK**. **b** Click **OK** in the **Add Application** window. On Windows 8, Windows 8.1, Windows 10, Windows Server 2012 and Windows Server 2012 R2: **a** Select **.NET v2.0** and click **OK**. **b** Click **OK** in the **Add Application** window. **NOTE:** Execute this step when updating OFS V3.x to OFS V3.60 with OPC XML server option installed.
8	Click the **OFS** folder in the **Internet Information Services Manager (IIS)** window (the icon changed and represents now the earth). **Result**: the following window appears:

Step	Action
9	Double-click the **Authentication icon**. **Result**: the following window appears: Check that **Anonymous authentication** is enabled. If not, click **Enable** on the right side of the window.
10	Click **Edit...** on the right side of the window. **Result**: the following window appears: Click **Set...** and type the name of the low privilege anonymous account created by IIS. Typically the format of the name is *IUSR_WorkstationName*. Click **OK**.

COM/DCOM Configuration

Overview

By default, a Web service has limited rights, and is unable to launch a COM server. The OPC XML-DA Web service acts as an OPC-DA client and uses COM to access the assigned OPC-DA server.

The following configurations can be set up in combination with the Web Service security settings.

NOTE: You are advised to restart the machine after configuring the DCOM security settings.

Configure an Identified OFS Access with Impersonation

The following table shows how to configure an identified OFS access with impersonation.

Step	Action
1	Start **DCOMcnfg** from the **Start** menu, then choose **Run**.
2	To choose the properties right-click **Console Root/Component Services/Computers/My Computer**, right-click **My Computer** and choose **Properties**, in the **Default Properties** tab.
3	Set **Default Authentication Level** to **Connect**.
4	Set **Default Impersonation Level** to **Identify** or **Impersonate**.
5	Then click the **Default COM Security** tab.
6	Click **Edit Limits** in **Access Permissions**.
7	Click **Add**, enter the user name of the authorized user, then continue with step 10 of Configuring IIS *(see page 60)*. Then click **OK**.
8	Click **OK** to close the dialog box window.
9	Click **Edit Limits** in **Launch and Activation Permissions**.
10	Click **Add**, enter the user name of the authorized user, then click **OK**.
11	Click **OK** to close the dialog box window, and **Exit**.

DCOM OFS settings should be set to "Use Default" to inherit from machine default settings. DCOM OFS configuration *(see page 52)*.

Chapter 5
OFS as an NT service

OFS as NT Service

Description

The OFService NT service is used to start OFS Server automatically each time the service is started.

In this case, the icon for the OFS server is not visible; the server is operating in background mode.

It is always possible to start Windows and manually stop the server by using the Control Panel.

Configuration Setting

NOTE: NT services are controlled by executing *services.msc* command line.

To use the NT service, you need to make the following modifications to the configuration of your machine:

Step	Action
1	Configure your server (alias, time out, and so on) preferably using the hidden option in the diagnostic folder.
2	Run the *OFSService.bat* batch file, which can be found in the directory containing the executable file of the server.
3	Configure the **OFService** service. **1.** Execute *services.msc*. **2.** Double-click **OFService** service. **3.** Select the **Log On** tab in the **Properties** dialog box. **4.** Select **This Account** and enter an account (the account can be an administrator of the computer declared into workgroup or member of a Domain). This account is referred below as **OFService Account**.
4	Configure COM/DCOM OFS Server stations .
5	Start the Services tool. **OFService** should appear in the list.
6	Select **OFservice**. The default value is **Manual**. You can then start **OFservice** and OFS by using **Start** and stop it using **Stop**. You can also start it automatically by setting **Startup** type to **Automatic**.
7	Close the Services tool.
8	Reboot your machine and OFS should run (use the Windows NT Task manager to verify). Before rebooting, you can test that everything is OK by starting OFService manually.

NOTE: The NT service cannot operate on a server in evaluation mode (client not yet registered) or in DEMO mode.

NOTE: You can run OFS server without opening a Windows session, follow the DCOM OFS configuration step 4 *(see page 52)*.

Uninstalling Services

To uninstall the OFS product while OFservice is running, proceed as follows:

Step	Action
1	Stop OFservice.
2	To cancel OFservice registration on the NTservice, run the OFSNoService.bat batch file which can be found in the directory which contains the executable file of the server.
3	Uninstall the product.

If you want to delete OFS as an NT service, but want to keep OFS installed, proceed as follows:

Step	Action
1	Stop OFservice.
2	To cancel OFservice registration on the NTservice, run the OFSNoService.bat batch file which can be found in the directory which contains the executable file of the server.
3	Start the DCOMcnfg tool. Select the Schneider-Aut OPC Factory Serve application, then Properties, then Identity and check The Interactive user box. Confirm and close DCOMcnfg, and restart the machine.

Part IV
User Guide

Overview

The purpose of this section is to guide users in the product's various features.

What Is in This Part?

This part contains the following chapters:

Chapter 6
The OFS Configuration Tool

Overview

This chapter introduces the tool for configuring the OFS product.

What Is in This Chapter?

This chapter contains the following sections:

Section 6.1
Introducing the Configuration Tool

Aim of this Section

The aim of this section is to introduce you to the OFS server Configuration Tool.

What Is in This Section?

This section contains the following topics:

Topic	Page
OFS Configuration Tool	73
Configuration Tool Execution	74

OFS Configuration Tool

At a Glance

OFS is an OPC data access server that may be used to read or write data on devices (in general PLCs, but not exclusively).

To do this, the server must have the following information for each device:

- The network to use,
- The address of the device on the network,
- The symbols table file to use if the device variables are accessed using symbols.

Moreover, the server supports a group of configuration parameters in order to best adapt the communication with the devices.

All the parameters are processed by the configuration tool, which makes it an essential component of the OFS product. It allows the user to configure the OFS server to connect it to networks, devices and symbols tables.

To use the server, one must first create an **alias** for each device that will be accessed.

An **alias** is a shortcut that may be used when a network address for the device is necessary (single replacement string). The use of an alias is also a very practical way to disconnect your OPC application from network addresses of devices that may be modified when necessary.

As the server does not have any symbol support function, you may indicate to the server the name and path of the symbols table file to use (one per device). It enables the view symbols function for the device.

You may next configure other device parameters using the properties page for the device.

NOTE: All changes made to the server configuration parameters are static: To enable the changes, the server must be shut down, then restarted.

Configuration Tool Execution

Description

To launch the OFS Configuration Tool:

- Click the Start button in the task bar,
- Select Programs\Schneider Electric\OFS\OFS Configuration Tool.

The upper part of the window offers a set of menus and a toolbar:
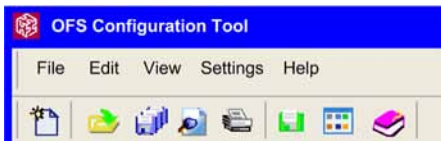
The upper part of the window offers a menu bar and a toolbar.

- File menu:
  - The **New Device Alias** option allows you to create new devices.
  - The **Open Archive** option allows you to restore a configuration from a backup file. See also the compatibility paragraph *(see page 128)*,
  - The **Save Archive as** option is used to save the server settings and the aliases and their properties in a file. This option is recommended if many aliases have been declared.
  - The **Save configuration option** allows you to save all the changes you have done, which will be taken into account after the next server starting up.
  - The **Reset Default device** settings option allows you to restore the default settings of the **Default devices** folder. Refer to The Default devices folder *(see page 100)*. This option is enabled only when **Default devices** is selected in the tree view. Refer to Introduction to the Standard parameters for editing aliases *(see page 79)*.
  - The **Print Preview** option allows you to preview your printing.
  - The **Print** option allows you to print or sent in a text file all the parameters.
- Edit menu: Access to copy, paste, and delete functions for the device Alias.

- View menu: Allows you to view the list of configured devices in several modes see below.
- Settings menu: Allows you to choose the soft language (English, French, or German) of the configuration tool.
- Help menu: Allows you to access to help while using OFS product.

NOTE: If a previous version of the configuration tool was installed and aliases already created, a compatibility dialog box appears the first time the program is run, which allows existing aliases to be restored. See the Compatibility *(see page 128)* paragraph for further details.

NOTE: To retrieve an OFS configuration created with a version of OFS older than version V3.60, run first the version V3.60 Configuration Tool and apply the settings before launching the OFS server.

Section 6.2
Configuration tool

Introducing the configuration tool

Presentation

When the configuration tool is opened, one or more aliases are created and the associated properties are set. The client application can create variables on the devices associated with these aliases. In most cases, the Alias definition is sufficient.

As with Windows explorer, you can view the list of configured devices in four modes:

- the tiles display mode,
- the icons display mode,
- the list display mode,
- the details display mode.

Tiles display mode

The tiles mode displays as follows:
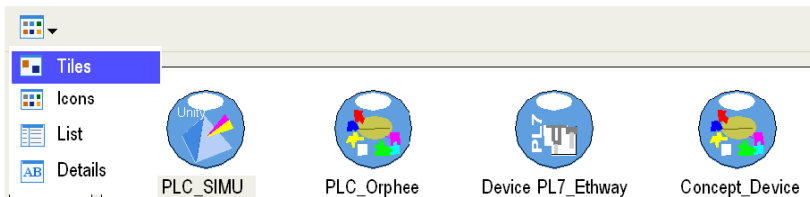
Icons display mode

The icons mode displays as follows:
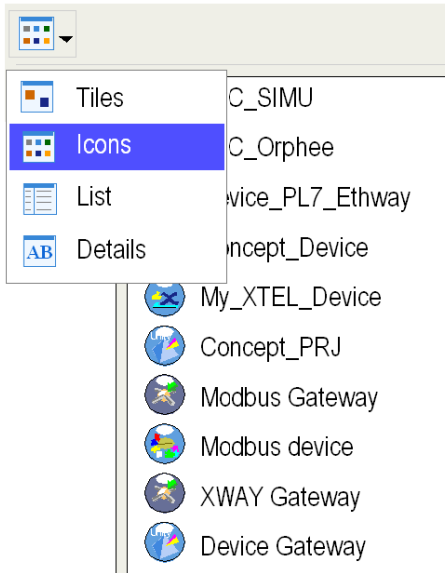
List display mode

The list mode displays as follows:
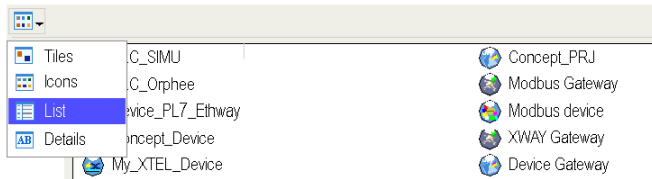
Details display mode

The details mode displays as follows:

Section 6.3
The Alias folder

Overview

This section describes alias management.

What Is in This Section?

This section contains the following topics:

Introduction to the standard parameters for editing aliases

Configuration tool: presentation

The browse in the configuration tool is presented by a tree view:
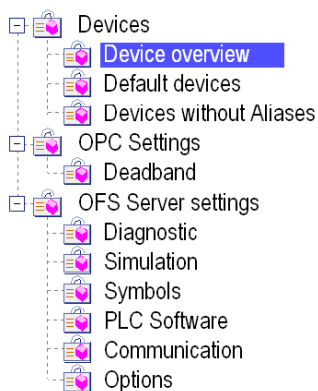
Tabs and folders

The table below describes the tabs and folders of the OFS Configuration Tool screen:

Tab	Folder(s)
Devices	**Device overview**: Allows you to view the list of configured devices and the properties of devices. **Default devices**: Defines the settings given by default when a new device is created. **Devices without Aliases**: Defines the settings applied for the devices without aliases, i.e. which are not in the device overview.
OPC settings	**Deadband**: Defines the settings for deadband feature.
OFS Server settings	**Diagnostic**: Defines the settings for diagnostic feature. **Simulation**: Defines the settings for simulation feature. **Symbols**: Defines the settings for symbols files. **PLC Software**: Defines the settings in relation with PLC software. **Communication**: Defines the settings for communication server feature. **Options**: Defines the optional settings for OFS server.

Editing the Device Network Address

Presentation

The Configuration tool offers a graphic assistant for configuring the network and the address].

To call the assistant, click on the **Device address** attached box [...] that corresponds to the alias selected:

Field	Description
Tree-view	**1st level**: X-Way or Direct addressing (depending on the network being used). The Other family is reserved for future extensions. **2nd level**: Type of network.
Alias address	Displays the alias string as the result of the selections. In Read Only for X-Way or Direct addresses, in Read/Write for other protocols.
Reset	Erases the string.
Cancel	Exit screen without taking into account the choices made.
OK	Exit screen taking into account the choices made.

Notes:

● The USBX driver is reserved for Unity Pro PLC.
● The X-Way drivers do not support the CONCEPT PLC.

X-Way Parameters

The illustration below shows the X-Way *(see page 343)* addressing modes:
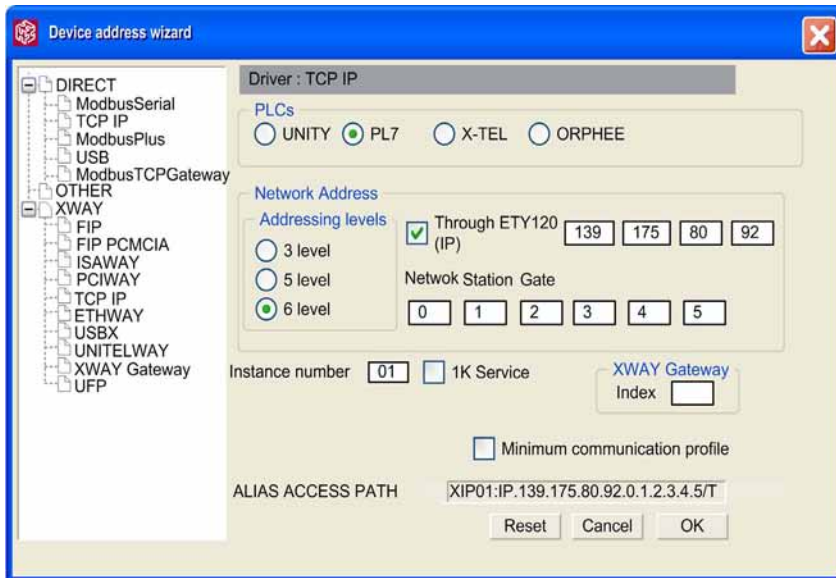
The table below describes the fields relative to X-Way parameters:

Field	Description
PLC's	This area enables you to identify the type of PLC in use: ● **UNITY** ≡ programming with Unity Pro ≡ /U at the end of the alias' address, ● **PL7** ≡ programming with PL7 ≡ /T at the end of the alias' address, ● **X-TEL** ≡ programming with X-TEL ≡ /S at the end of the alias' address, ● **ORPHEE** ≡ programming with ORPHEE ≡ /J at the end of the alias address.
Through ETY 120	Driver TCPIP only, reserved for certain modules (e.g.: TSX ETY120): If selected, enter an IP address.
Addressing levels	Addressing levels for X-Way addresses. See Communication *(see page 343)* section.
Network/Station/ Gate/Index	X-Way Address. The unlabelled text boxes are grayed out depending on the addressing level selected. For more details on X-Way addressing, see the section entitled Communication *(see page 343)*. Index is the XWAY gateway number. To have an index number, you must create a virtual alias (without associated symbol table file) by using Gateway XWAY *(see page 94)* as driver.
Instance number	One instance per driver installed. Generally equal to 1. Each driver corresponds to a communication card in the PC.

1 K Service (for PL7 equipment only)	This option increases the communication performance by raising the size of the frames to 1024 bytes. The PLC application must be set-up in periodic and not cyclic mode. It sets the gate to 7 which limits addressing to 3 levels. Moreover, the data are no longer accessed synchronously with the PLC cycle. In certain cases this may lead to a lack of data consistency.
Minimum communication profile	This option is accessible for PLCs other than Unity Pro: ● When checked, the server uses the default communication parameter values, and no adjustment is performed. The communication performance may not be optimum. ● When not checked (by default), the OFS server adjusts the communication parameters optimally, in order to dialogue with the associated device.

Direct Address Parameters

The direct addressing principles are explained in the Communication section:
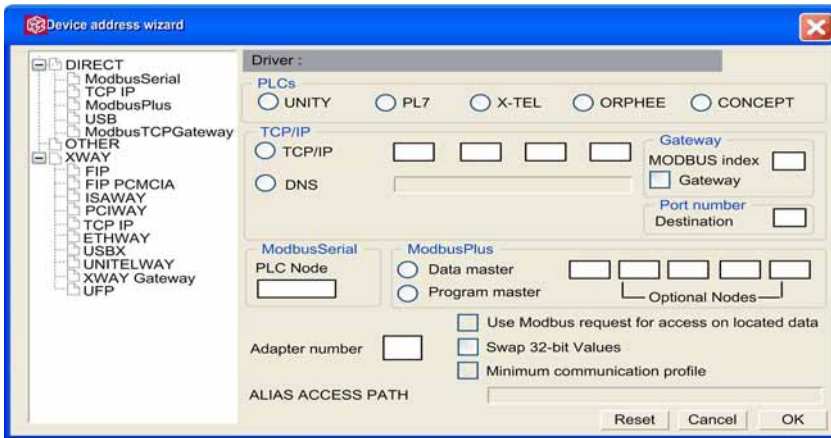
The table below describes the fields relative to direct address parameters:

Field	Description
PLC's	This area enables you to identify the type of PLC in use: ● **UNITY** ≡ programming with Unity Pro ≡ /U at the end of the alias' address, ● **PL7** ≡ programming with PL7 ≡ /T at the end of the alias' address, ● **X-TEL** ≡ programming with X-TEL ≡ /S at the end of the alias' address, ● **ORPHEE** ≡ programming with ORPHEE ≡ /J at the end of the alias address, ● **CONCEPT** ≡ programming with Concept ≡ /Q at the end of the alias' address.

ModbusSerial	The PLC node is only available for Modbus RTU. It defines the slave number of the device inside the Modbus network. Premium Unity PLCs are not accessible via the Modbus Serial Link.
Modbus Plus	Modbus Plus only: **Data master**: limited rights (read/write variables). **Program master**: unlimited rights, reserved for the Concept and Unity Pro programming workshop (read/write variables, program modification and configuration). Enter the Modbus address (the first value is compulsory, the others are optional depending on routing levels)
Modbus request for access on located data	This option can be used to increase dynamic performance when a Unity Pro PLC is accessed via an NOE (Ethernet TCP/IP) or NOM (Modbus Plus) or NWM (FactoryCast HMI Web Server) communication module and only in these three cases. When this box is checked, OFS uses Modbus requests to access located data and these modules are able to process four times more requests in a PLC cycle. Only the topological objects %MW, %MD and %MF are supported if the box is checked. However, Unity Pro protocol requests can sometimes prove more advantageous to use as they are more economical in terms of number of data exchanges with the PLC *(see page 366)*. If your performance is OK, it is recommended that you do not check this box.
Swap 32 bits values	This option is available for Concept PLCs. When checked, the real float or long values (%MF or %MD) are swapped in the Modbus requests to be consistent with certain specific devices.
Minimum communication profile	This option is accessible for PLCs other than Unity Pro: - When checked, the server uses the default communication parameter values, and no adjustment is performed. The communication performance may not be optimum. - When not checked (by default), the OFS server adjusts the communication parameters optimally, in order to dialogue with the associated device.
Adapter number	This setting is available for Modbus drivers. It defines the instance of the driver to be used.
Gateway	The MODBUS index is available for TCP/IP and Modbus TCPGateway drivers. It defines the slave number of the device in the Modbus network The Gateway checkbox is available for the Modbus TCPGateway driver. If it is checked, it defines the Gateway device. The MaxChannel and the MaxPending values of this gateway can be set. They are reported to all the devices that have the same IP address.
Port number	This setting is available for TCP/IP driver. It allows defining a TCP port different from the standard one (502) to manage complex networking.

Associating a Symbols Table File

Description

A symbols table file can be associated with the alias, in order to provide access to the symbols for the variables of this device. The symbols file is generated by the PLC programming software: Unity Pro for Premium and Quantum PLCs, PL7 for Premium/Micro PLCs or Concept for Quantum PLCs.

For the devices of the Series 7 and S1000 ranges, the symbols file can be obtained in the same way as for a Premium, but having first converted the application to Premium format. The only restriction is that no consistency check will be possible with the application loaded in the PLC.

Clicking in the "Symbols table file" attached box in 'General' part for the selected alias brings up a file explorer:
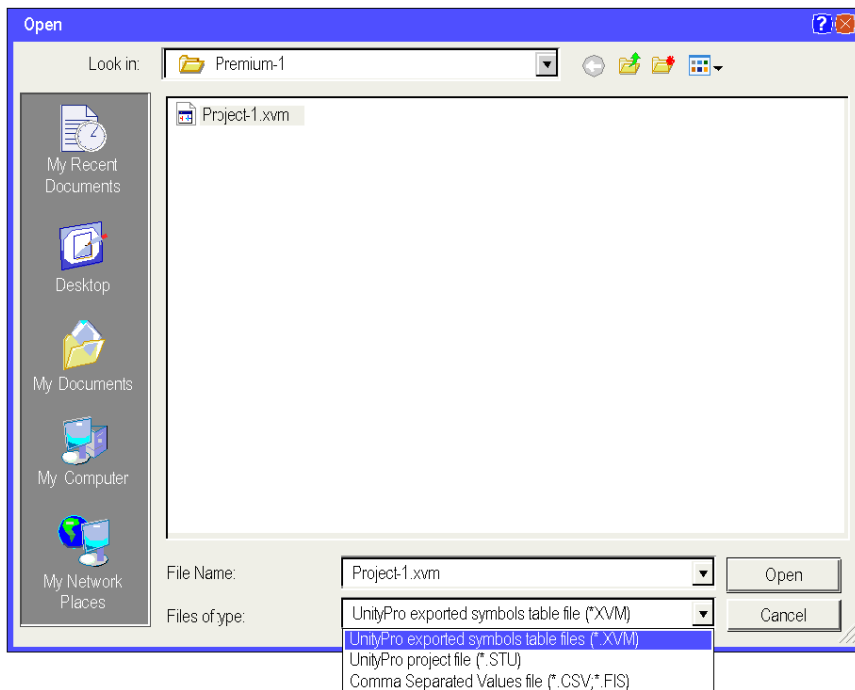
The file types that can be inserted are listed in the "type of files" list box. Select the appropriate file type.

Enter the file of your choice and click "Open". The file name and directory will then be displayed.

NOTE: The path locating the symbol file contains neither extended characters nor unicode characters.

Link with Unity Pro

Description

To install the Unity Pro link via the OFS server, select the .stu project file (see *Associating a Symbols Table File, page 84* and see *Symbol management, page 266*) as symbols file for any device or group.

This .stu file authorizes the consistency check (application name and version) between the symbol table file and the application in the PLC *(see page 102)*.

The direct link with Unity Pro can thus be used to:

● access the database of one or more Unity Pro projects,
● support symbols,
● consult the symbols,
● access the unlocated variables and structured data,
● perform a dynamic consistency check, with automatic reloading of the symbols where changes have been made to the PLC application.

NOTE: Unity Pro and OFS server may be on the same machine or different machines (DCOM link).

Link with Concept

Description

OFS supports the following versions of Concept:
● Concept 2.2 SR2,
● Concept 2.5 SR2 (and above).

To install the Concept link, select the .prj project file and as symbols file for any device or group.

This prj file authorizes the consistency check (application name and version) between the symbol table file and the application in the PLC *(see page 102)*.

The Concept workshop and the prj files should always be located on the same machine. The OFS Server can be located either on the Concept machine (usual case) or on another machine (Remote Concept Link feature).

In order to do this:
● edit the usual Concept shortcut properties,
● in the Shortcut tab, check the "Run in Separate Memory Space" box.

With OFS, more than one Concept project can be used at once, as long as they are from the same version of Concept. In order to do this, create the aliases required, and for each of these select a different project file.

OFS software, when used with the stripped Quantum executable file, will not read unlocated variables.

If you anticipate using unlocated variables:
● Use the full Quantum executable, not a stripped down version.
● Activate IEC runtime on the PC.
● Check the unlocated media option on the properties page.
 Otherwise, you will not be able to access unlocated variables.

Concept and OFS may run simultaneously on the same Concept project. Several Concept projects may be opened simultaneously, as long as they are all of the same version.

The Remote Client

The remote link offers exactly the same features as the normal Concept link. The only difference is that the Concept machine (where the Concept programming tool and the Concept project files are situated) is not the one on which the OFS server or the simulator is launched.

These machines must be linked by DCOM (usually on TCP/IP). An OFS server (with a license) or an OFS simulator (DEMO mode) must be installed on the Concept machine. An appropriate DCOM configuration must be performed to enable access to this server which is called "proxy server".

On the OFS machine, when specifying a Concept project, open the device properties page to check the appropriate remote Concept option (the proxy server is either an OFS server or an OFS simulator) and give the complete access path of the Concept machine.

The path of the Concept project must be the same as that seen by the proxy server on the Concept machine (it must begin with the letter of a drive, followed by the complete path).

Support of symbols

Description

This function enables you to replace the address of any variable by its name in the PLC application (e.g.: use of "Symbol" rather than the topological address "%MW1" or instead of State RAM location "400001"). It amounts to a string substitution, and has no effect on Read/Write operations.

The supported symbol table formats are:

- PL7 symbol table file or exported project file,
- Concept exported symbol table file,
- Concept project file (direct link with Concept database),
- Modsoft exported symbol table file,
- CSV symbol table file (Excel exported format),
- Taylor exported symbol table file (identical to Excel format),
- Unity Pro exported symbol table file,
- Unity Pro project file (direct link with Unity Pro).

NOTE: For old ranges: XTEL files must be converted to PL7 format in order to use symbols on the series 7 (using the "PL7-3 converter" function of the PL7 PRO). The series 1000 does not allow the use of symbols.

Setting the Alias Properties

Presentation

Now that the alias has an address, it is necessary to adjust its property settings.

These parameters will allow the behavior of the server to be adapted for the associated alias.

For each alias that is declared, it is possible to set up the following parameters:

- use of a symbols table file,
- access rights to the variables,
- simulation or real access to the device,
- consistency check between variables and data base,
- waiting time before "Time Out",
- symbol pre-load function for better performance during alias run-time,
- number of channel reservations,
- automatic data write operations from the device itself.

The Configuration Tool provides a property dialog box for this purpose.

Select the alias in the device overview.

General Settings

The general settings display this way:

Device name	DevExample_1	
Device address	MBT:139.192.80.65/U	...
Device Type	Default ▾	
⊟ General		
Symbol Access Mode	☐ Located only	
Symbol table file		...
PLC Embedded Data	☑ Using Data Dictionary	☑ No Communication Break
Preload settings	○ No Preload	○ Symbol table ⊙ Device
Dynamic consistency	☑ Dynamic consistency	☑ New Symbol Detection
Consistency level	○ Strict	⊙ Debug
Option	☐ Simulated	☐ Read Only
Comment		
⊞ Communication information		

The table describes the fields relative to the general settings:

Field	Description
Symbol Access Mode	For Unity devices only, when **Located Only** is checked, only direct (not symbolized) topological addresses are supported. This mode excludes **Symbol table file** and **Using Data Dictionary** configuration. It provides **No Communication Break** capability during Unity Pro build change *(see page 287)* without any other configuration, and whatever Unity Pro or Firmware versions. Refer to the chapter Unity Pro Variables on OFS\ Direct addressing data instances to have the list of supported objects *(see page 233)*. Objects arrays are also supported.
Symbol table file	Name and path of the **Symbol table file**. See dedicated section *(see page 84)*. It can be entered and changed here, or directly from the grid. The size of the character string is limited to 255 characters.
PLC Embedded Data	**Using Data Dictionary** option: Associated with a Unity Pro symbol file, this option allows the OFS server to automatically resynchronize the addresses of the variables in case of inconsistency detection after of an application on-line modification. In addition, without any symbol file, this option allows the OFS server to browse the application variables. This option is visible in case of TCP IP direct driver and Unity Pro PLC. **No Communication Break** option: A build change done through a connected Unity Pro breaks the communication during the symbol database reload and the inconsistency detection. As the result, all the quality of animated items is set to BAD. To avoid this disturbance, a synchronization mechanism is set up between OFS / Unity Pro and the CPU firmware by checking the option **No Communication Break**. Refer to the Version Compatibilities.
Database access	*Not available for the devices which can be programmed by X-TEL, ORPHEE and PL7 software workshops.* Allows you to set up the Concept or Unity Pro database in relation to the OFS server: ● either the server is on the same machine as the database: Local, ● or the server is on a remote machine: Remote server.
Preload settings	Allows data to be pre-loaded at server start-up rather than when running: ● **No preload**: by default. ● **Symbols table**: pre-loads the symbol table. ● **Device**: loads the symbols table if one exists and creates a continuous connection to the device.

Field	Description
Dynamic consistency	This option can greatly speed up the import of symbol for HMI-tools .PRJ (Concept) or .STU (Unity Pro) or .XVM (Unity Pro) symbol files. This option is used to define the rule to apply in the case of a detection of an inconsistency *(see page 122)*. A build change concerning a new variable creation does not imply a database resynchronization in Debug Consistency level. For Unity Pro PLC alias, the **New Symbol Detection** option allows to browse a new variable adding in Debug Consistency level if the Unity Pro is in version 6.0 or later.
Consistency level	*Available if the symbols table is .prj-type (Concept) or .stu-type (Unity Pro) or .xvm-type (Unity Pro).For .fef and .scy-types of PL7, consistency is checked at device startup.* OFS may detect an inconsistency when reading / writing items. In the case of weak frequency of group polling, this option allows to periodically check the consistency at the period defined in the dynamic coherency check in the PLC software folder. ● **Strict level**: A variation with the application signature stops the communication. ● **Debugging level**: If an unimportant inconsistency is detected, the communication continues.
Option	**Simulated option:** No physical connection is made to the device. The variables are simulated directly by the server. See also the simulation folder *(see page 118)*. **Read only option**: All variables relating to the device are **Read Only**.
Modbus/ Concept	*Specific to the devices programmed using Concept.* Enables support of unlocated variables. Active this option when the user wishes to use the access to unlocated variables function. For more information on this subject, go to the concept link *(see page 86)* section.
Comment	Allows to add any comment.

Driver and push data settings

The driver and push data settings display this way:

The table describes the driver and push data fields:

Field	Description
Driver information	Summary of driver settings in read only.
Push data area	Not available for certain networks, in particular the ModbusTCPGateway option. Write orders from the device to the server. For more information on this subject, go to the push data section *(see page 105)*. N.B.: if "No Push Data" is selected, other fields are not significant.

Adjustment Settings

The adjustment settings display this way:

The table describes the adjustment fields:

Field	Description
Max Channels	Number of channels allocated to the device. **Note:** This parameter has a very large impact on performance, as it determines the maximum number of requests that OFS can process in parallel *(see page 348)*.
Max Pending	Maximum number of requests authorized to be pending (awaiting response) for a device. This number does not depend on the number of opened channels. Range: [0...256], 0 is the default value and it indicates that OFScalculates the optimal value.
Frame Timeout	Permissible delay between request and answer. Range: [1000...10900], to the maximum of a third of Device Timeout.
Device Timeout	Delay for the status change of the device (Missing, Unknow or OK). Range: [3000...32767], at least three times the Frame timeout (or 0 to disable the feature). For more information on this subject, see frame and device timeout *(see page 98)*.

Modbus Gateway

The illustration shows an example of a gateway-type alias (Modbus Gateway):

	Device name: Modbus Gateway
Device name	Modbus Gateway
Device address	MBTG:139.175.80.40 [...]
⊟ General	
Comment	My comment
⊟ Communication information	
⊞ Driver information	
⊟ Adjustment information	
Max Channels	1
Max Pending	0

NOTE: here, you must check 'gateway' in the ModbusTCPGateway driver.

Field	Description
Max Channels	The indicated value must be consistent with the technical characteristics of the gateway. If this parameter is greater than the capacity of the gateway, system messages on the management of sockets may be displayed in the trace file. **Note:** This parameter has a very large impact on performance, as it determines the maximum number of requests that OFS can process in parallel *(see page 348)*.
Maximum pending requests	Maximum number of requests authorized to be pending (awaiting response) for a device. This number does not depend on the number of opened channels. Range: [0...256], 0 is the default value and it indicates that OFScalculates the optimal value.

The illustration shows an example of a gateway device-type alias:

Device name	Modbus device	
Device address	MBTG:139.175.80.40;33/Q	...
⊟ General		
Symbol table file		...
PLC Embedded Data	☐ Using Data Dictionary	☐ No Communication Break
Option	☐ Simulated	☐ Read Only
Preload settings	⊙ No Preload ○ Symbol table	○ Device
Comment		
⊟ Communication information		
⊞ Driver information		
⊟ Adjustment information		
Device timeout (ms)	5000	
Frame timeout (ms)	1000	
Max channels	1	
Max Pending	0	

In this example, the alias is associated with a Concept (/Q) project with a **Modbus index** of 33, on a gateway with the IP address 139.175.80.40.

This device used the gateway parameter values **Max channels** and **Max pending**. The parameters can only be modified in the gateway alias properties, not in the alias.

For this type of alias, the Push function is not supported.

XWAY Gateway

The illustration shows an example of a gateway-type alias (XWAY Gateway):

Device name	XWAY Gateway	
Device address	XWAYG:29	...
⊞ General		
⊟ Communication Information		
⊞ Driver information		
⊟ Adjustment information		
Max Pending	10	

Maximum number of authorized requests pending responses for a device, where the server sends several requests in parallel (0 by default).

The illustration shows an example of a gateway device-type alias:

Device name	Device Gateway	
Device address	XIP01:0.2.0;29/U	...
⊟ General		
Symbol table file		...
PLC Embedded Data	☐ Using Data Dictionary ☐ No Communication Break	
Preload settings	⦿ No Preload ○ Symbol table ○ Device	
Option	☐ Simulated ☐ Read Only	
Comment		
⊟ Communication information		
⊞ Driver information		
⊞ Push Data		
⊟ Adjustment information		
Device timeout (ms)	5000	
Frame timeout (ms)	1000	
Max channels	1	
Max Pending	0	

Here, the alias is associated with a Unity Pro(/U) project with a **XWAY index** of 29 on a gateway with XIP address 0.2.0.

This device uses the gateway parameter values **Max Pending**. These parameters can only be modified in the gateway alias properties, not in the alias.

Section 6.4
The Device overview folder

Aim of this section

The aim of this section is to describe the device overview folder.

What Is in This Section?

This section contains the following topics:

Creating a new device

Creating a new device

To create a new device, you have to follow the procedure described below:

Step	Action
1	From the **File menu**, select **New Device Alias.** **Note**: You can use the contextual menu by right mouse clicking on top window to create a new device. ***Result***: A new device is created and the configuration tool gives a default name that can be immediately changed or later by a "rename". **Note**: you cannot have two identical names.
2	Assign a name to your device.
3	Define the device network address *(see page 80)* which includes the network driver, the type and the device address.
4	Set the Device type (optional and only for Unity devices). If your application plans to use Enhanced Diagnostics specific items (refer to the chapter Enhanced Diagnostics Specific Items *(see page 202)*, you have the possibility to set the device type to match with the actual device present in your application. In other cases, let the value be set to Default. When the Device type is set, you can browse or animate only the Enhanced diagnostics specific items related to that type. The Default value allows browsing or animating Enhanced Diagnostics specific items common to all devices types. **Note**: If the device is connected during browse or items adding operations, the actual device type (identified at device connection) overrides the configured type.
5	Provide a Symbols table file name *(see page 84)* (optional).
6	Set the alias properties *(see page 88)* which are related to how the server will behave towards the variables created on that alias. **Note**: It is recommended to associate a single and unique alias for each device. If not, the properties of the first alias will be used to create an item with the second alias.

Additional information

From the **Edit menu**, you can copy, paste or delete a device. You can also use the contextual menu (mouse right-click) for the same operations.

Adjusting timeout item values

Description

The **frame timeout** shows the maximum length of time the Server will wait for an answer from a device after sending a request. This can be defined according to the device in its properties page.

The frame timeout may be configured dynamically, device by device, using the specific *(see page 193)* #TimeOut item.

To avoids item quality flickering and over-long OPC application startup time due to missing devices, a **device timeout** feature has been introduced.

When active, this option has two effects:

- if the device is physically missing, then it will be considered 'missing' for a time equal to the Device Timeout.
- when the communication is interrupted an observation period of a duration equal to the device timeout is engaged. After this period, and if the communication stays inoperational, the device is declared missing and all of its active items are affected (the quality of the elements is set to BAD).

This timeout can be defined device by device in the properties page.

If the value in the device property page is set to 0, then the device timeout takes the default value (5000 ms).

This feature is **incompatible** with **synchronous groups**.

If the multi-channel *(see page 348)* feature is enabled, the frame timeout value is the same for all channels open with a given device.

Adjusting Communication Timeout with a device

Description

Various parameters can be used to set this important communication parameter. They can be global or device *(see page 88)* specific parameters. They can also be static (configured using the Configuration Tool) or dynamic (configured using a write specific *(see page 193)* item *(see page 98)* and method.

Section 6.5
The Default devices folder

Aim of this Section

The aim of this section is to describe template management.

What Is in This Section?

This section contains the following topics:

Topic	Page
The Default devices folder	101
Dynamic Consistency and Consistency Level	102
Push data support	105

The Default devices folder

Description

This folder lists all the alias properties applied by default when an alias is created.

Illustration:

Default device settings			
General			
PLC Embedded Data	☑ Using Data Dictionary		☑ No Communication Break
Preload settings	○ No Preload	○ Symbol table	◉ Device
Dynamic consistency	☑ Dynamic consistency		☑ New Symbol Detection
Consistency level	○ Strict		◉ Debug
Option	☐ Simulated		☐ Read Only
Communication information			
Push Data			
Service support	☐ Available		
Initial value	◉ Zero		○ Read from device
Push Data area	Base address 0	Size	0
Stamp option	☐		
Quality check rate (s)	0		
Adjustment information			
Max channels	4		
Max Pending	0		
Device timeout (ms)	5000		
Frame timeout (ms)	1000		

A full set of default parameters of your choice can be set up so that property adjustment for each new alias created is minimized.

Dynamic Consistency and Consistency Level

Introduction

To access the value of a symbolized object in the PLC memory, the OFS server must know:
- The topological address associated with this object in the case of a located object,
- The address of this object in PLC memory in other case.

The correspondence between the symbols and the topological addresses is done in 3 different ways:
- Through a file exported from UnityPro software (XVM exported file type),
- Through directly by the UnityPro software (STU file type),
- Through directly from the PLC.

The OFS server provides different mechanisms to manage the consistency between the OFS application and the PLC. These mechanisms depend on:
- The PLC type,
- The synchronization type (by XVM file, by UnityPro or directly by the PLC),
- The consistency level (Strict, Debug, 'Dynamic consistency check').

Description

The **dynamic consistency control** function is available on PLCs configured with the Concept link *(see page 86)* or the Unity Pro link *(see page 85)* or the Unity Pro file of exported XVM symbols *(see page 269)* or the direct PLC symbol database *(see page 283)*.

This function allows the server to check at regular intervals the consistency between the application loaded in the PLC and the currently open Concept or Unity Pro symbols database or the direct PLC symbol database.

Thus by using both Concept and OFS or Unity Pro and OFS, uploading several modifications to the PLC using Concept or Unity Pro will cause, after several seconds, closing and reloading of the Concept or Unity Pro database by the OFS server.

The automatic loading function can be deactivated (see PLC Software folder *(see page 122)*). In this case, reloading may be performed manually using the OFS Manager *(see page 143)*, with the reload and update services.

OFS automatically updates its network requests in the event that certain variable locations have changed and, if the OPC browse interface is opened and closed, an updated list of symbols will appear.

With Concept, it is possible to use non-located variables but their value cannot be read as long as the variables are not used. With OFS, all non-located and unused variables will be displayed with the attribute "Quality Bad". If, following automatic updating of the Concept database, OFS discovers that some unlocalized variables that were not used, are in fact used, the attribute "Quality Bad" will be replaced by the attribute "Quality Good" and the updated value will be displayed.

To use this function:
- Configure the device with a Concept or Unity Pro project file ,
- Confirm the dynamic consistency check option in the devices properties page.

NOTE: For proper operation with Concept or Unity Pro, the automatic save option must be enabled (in Concept, in the menu Options->Preference->Common, the option **Save after download** is checked). If you do not wish to use this option, you must save manually.

The OFS server checks the consistency between the application loaded in the device and the symbols file linked to its alias if the option **Dynamic consistency check** is checked. When there is a variation, it applies the selected consistency policy. If the dynamic control option is not checked, consistency is checked only on each access to the device.

Consistency Policy

The consistency policy defines the procedure to follow if there is any variation between the application of the PLC and that of Unity Pro. This behavior is defined in the configuration of the alias.

Detailed Description Unity

The policy for a direct link with Unity Pro is as follows:
- **Strict level**: If there is any variation, all Items of the device are positioned with the Quality field set to "Bad".
- **Debugging level**: In the event of a minor variation (not affecting the existing application variables), animation of all variables is maintained. If the modification affects the memory location of the variables, the animation is suspended for all variables.

NOTE: In debugging mode, because of the permissiveness of the algorithm, some "destruction"-type operations may not be detected by the server.

Concept Detailed Description

The policy for a direct link with Concept is as follows:
- **Strict level**: OFS checks that the application in the device is strictly identical to the one in the symbols file. If there is any variation, all the Symbolic Items of the device have their Quality field set to Bad.
- **Debugging level**: In the event of inconsistency, the animation of non-located symbols is stopped, the located symbols remain accessible for reading and writing.

Direct PLC Symbol Database Detailed Description

The policy for a direct PLC Symbol Database, combined with or without a symbol file, is as follows:

By default the **Dynamic consistency** is always available with a **Debug** consistency level.

This profile cannot be modified in the configuration tool when the**Using Data Dictionary** option is set.

In the event of a modification that:
- Is minor (not affecting the existing application variables), animation of all variables is maintained
- Affects the memory location of the variables, the server re-synchronizes the animated variables directly from the embedded data dictionary in the PLC

NOTE: The Reload Database option in the PLC symbol folder does not impact the Direct PLC Symbol Database policy.

Push data support

Description

As a general rule, to update OPC items automatically, the server sends some network requests to the PLC, then waits for the PLC responses to update its internal data tables. This is called *polling* the device.

In contrast, the *Push data* feature corresponds to the spontaneous transmission of data by the PLC to the active server, without any request having come from the server.

The data is regarded as being *pushed* by the PLC. This feature is particularly worthwhile when the values of the data being monitored do not change very frequently. This does, however, mean that specific processes have to be included in the PLC application for sending data.

This function is supported on TCP/IP (except ETY 120), Fipway and Ethway networks.

This feature can be enabled and configured PLC by PLC using the device *(see page 88)* property page.

Data sent by the device to the server should fit within the Push data range that has been defined for this device. Only one range can be defined for each device (using the device *(see page 88)* properties page).

Any number of OPC items can be defined within this range. They are seen as ordinary OPC items.

The device has an option which can be used to send a timestamp *(see page 88)* with the data, used by the server to update the timestamp property of all items associated with Push data.

Procedure and Example

The Push data should be sent to the server using a request code 37 h for X-Way (generally via the function WRITE_VAR PL7) and the function code 16 for Modbus (generally via the Concept EFB function WRITE_REG).

Concept example of using WRITE_REG to test the PUSH DATA function:
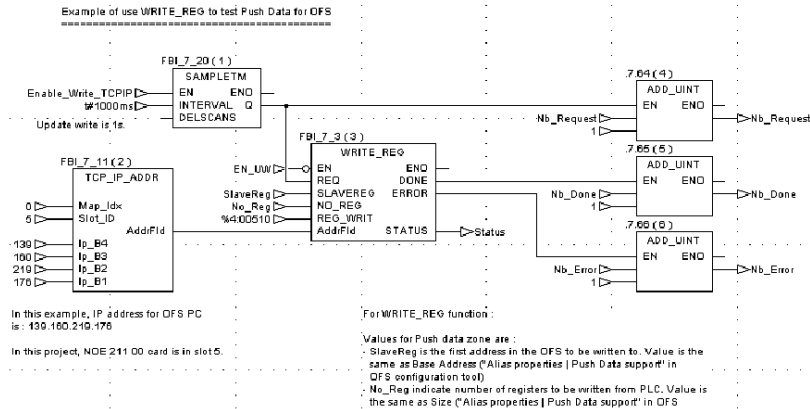
Some sample applications are provided on the DVD to show how a PLC application can send Push Data to the server.

In both cases, the functions and the behavior of the server are perfectly identical.

To use these functions, you must adhere to the following procedure:

1	Creating an alias for the device with the Configuration Tool,
2	Open the properties page of the device.
3	Define the range of Push data for the device (Base and Size). Example: range %MW1000..%MW1500 : base = 1000, size = 500 Example: range 401000..401200 : base = 1000, size = 200
4	Define the initialization mode for the Push data area: Values to 0 or values read from the device.
5	Close the properties page and the Configuration Tool and save the parameters.
6	Create an application or use a sample application provided on the DVD capable of sending Push data to the server (check the consistency with the Push data range shown below for the device). Load it into the PLC.
7	Launch the OPC test client, then connect it to the OFS server.
8	Create an item associated with the equipment in order to establish the connection and initialize the Push data range.
9	In the server diagnostics window, a message should appear and indicate that the Push data are being received from the device.
10	Create an item in the Push data range using the OPC test client.

11	Launch the write operation from the application.
12	The value of the item should have been updated.
13	You can verify the procedure using the server diagnostics interface(Network window), then read the counters in the transaction area: Slave Request and Slave Answers.

The number of OPC items that can be created in the Push data range is unlimited (single variables and tables) but it is not possible to create variables that straddle the limits of the area.

In addition to its value, each OPC item must include important attributes:

● Quality,
● Timestamp.

For the items included in the Push data range, the Quality attribute is identical for all items and may be:

● Always defined as Good (if the *Quality Check Rate* value defined in the device properties page is equal to 0),
● Defined according to the communication status and the operation mode of the device (if the *Quality Check Rate* value defined in the properties page of the device is defined as NN and not as 0). Every NN seconds, the server will attempt to read the operating mode of the device:
  ○ When the communication is interrupted, the quality is defined as Bad,
  ○ If the communication is established and if the operating mode is defined as RUN, the quality is defined as Good,
  ○ If the communication is established and if the operating mode is other than RUN (generally this means STOP), the quality is defined as Uncertain,

  The Quality Check Rate option is only available for PLCs of types Concept, PL7 on X-Way and Unity Pro.

For items included within the Push data range, when the Timestamp option is used, the date/time is set as follows:

● The current time and date of the server when a read operation is requested by the OPC client,
● The time and date of the PLC when the server receives the new values of this latter,
● The current time and date of the server when the Push data area is initialized (whether it be with the value of 0 or read from the device).

The Timestamp option may be enabled individually for each alias, from the properties page.

To send the time/date to the server, the PLC must include it in the header of the data sent.

Place the GMT time on the PLC in compliance with the OPC standard.

To facilitate data formatting according to the PLC, two header formats may be used.

Illustration of the 2 formats:
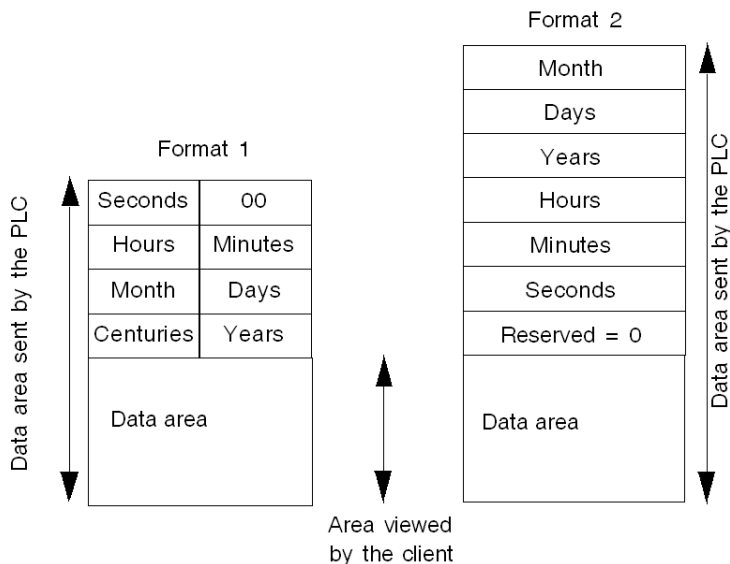
NOTE: The difference between the two formats is made by OFS by checking the least significant byte of the first word that contains 0 in format 1 and a value from 1 to 12 in the second case.

NOTE: On a Premium, the date/time may be easily be inserted using the RRTC function.

NOTE: Some sample PLC applications are provided on the DVD.

In order to allow creation of the Push data range and the reception of the associated data prior to creating an item, configure the device (in the device properties page) so that it is preloaded when the server is started.

All OPC write operations are performed directly on the device. The Push data area is not affected at all.

All OPC read operations of the device are performed directly (unless for a cache read), the Push data area is simultaneously updated.

For X-Way devices, only variables %MW and %MD may be associated with the Push data area. The others (%MB, %MF) are managed as though the area was not defined.

For Concept-type devices, the Push data area is always located in 4x. Only variables of type INT, DINT or FLOAT may be created there.

NOTE:
- If you use the Push function on a Premium via TCPIP in direct addressing mode, and the XIP driver is also enabled, make sure the IP of the Premium is not declared in the XIP driver (the same TCP/IP port 502 is shared).
- Only one Push data area may be created per device. However, if the device is accessible by several network addresses, it is then possible to define one area per address,
- The Push Data function is not supported for I/O objects. It is however possible to send them to the OFS server by copying the I/O objects to standard objects .

NOTE: The size of the data range configured should be at least equal to the quantity of data sent by the device.

NOTE: On TCPIP, OFS listens to port 502 (Schneider TCP Port). Some Schneider tools also use this port (especially the PLC simulator). They should thus not be started on the same machine as OFS.

NOTE:
- On a Premium using TCP/IP the PC extension number must be equal or greater than 100 in the configuration of the Ethernet module in order to specify the use of the Modbus/TCP protocol.
- On a Quantum, the communication function blocks use only located variables.

NOTE: The Push Data feature is not compatible with the Ethernet I/O scanning service.

Section 6.6
The Devices without Aliases folder

Devices without Aliases folder

Presentation

The Devices without Aliases folder displays as follows:

⌐	**Devices without Aliases**		
⊟	**General**		
	Option	☐ Simulated	☐ Read Only
	Symbol Access Mode	☐ Located Only	
	Unlocated support	☐ Unlocated support	
⊟	**Communication information**		
	⊟ **Adjustment information**		
	Max channels	1	
	Device timeout (ms)	5000	
	Frame timeout (ms)	1000	

Description

The options to be chosen here are the same as in the device overview *(see page 96)*. The selections made here are applied only to the devices created without aliases, or aliases created dynamically with OFS Manager, at server run time.

Section 6.7
The Deadband folder

Aim of this Section

The aim of this section is to describe deadband management of an analog value.

What Is in This Section?

This section contains the following topics:

The Deadband Folder

Description

Deadband is a percentage of the range of values that an analog variable can take. If the value exceeds the range, the cached value is updated and the server sends a notification. It is part of the Group attribute, and so is applied to every variable of that group and considered as the notification criterion when the value changes.

This folder defines the settings for deadband feature:
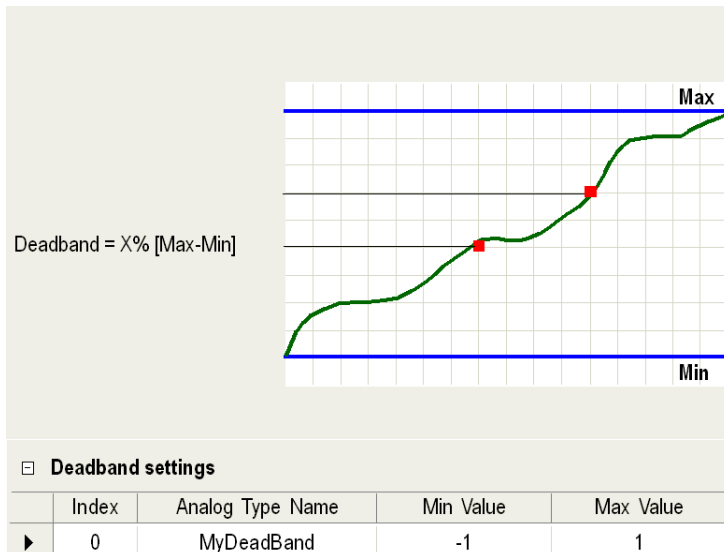
The range can be adjusted here for each variable, whether floating or integer, with minimum and maximum values.

You can add or delete entries by using the contextual menu (mouse right-click) over the table.

NOTE: The Configuration tool will not allow a value less than that entered in the **Min value** field to be entered in the **Max value** field.

Definition

Deadbanding is associated with the cyclic reading of a user group, and is a method of filtering notifications of changes in the value of items: it avoids waking up the client application when the variable changes within a dead range around the last value received.

Note: The deadbanding mechanism does not reduce the flow of requests between the server and the PLC. It is used to reduce the number of notifications sent by the server, and thus processed by the client application: this will reduce the load on the processor.

Note: The deadbanding mechanism has no effect when the client requests a synchronous or asynchronous read or refresh.

Description of the Deadband mechanism

Description

The OFS server uses deadbanding as specified by OPC standards:

In general, deadbanding only concerns real variables: *%MF*, called *ANALOG* variables by the OPC standard. As an extension of this standard, it is possible to use this feature for integer variables provided that the configuration steps described below have been followed.

NOTE: The OFS server uses this OPC term to refer to *floating point* type PLC variables, even though this term does not correspond to the idea of analog variables usually used in the control system field.

Deadbanding is based on the following notions:

- The analog type, defined with min. and max. limits which represent the range of values (interval) of the variables being handled. This notion has been introduced because the OFS server has no way to obtain these maximum and minimum values directly from the programming tool (PL7, Concept, Unity Pro, XTEL or ORPHEE).

Example:

AnalogType = [-1.0, 1.0]

The max. limit of an analog type (1.0 in the above example) is called Engineering Unit high bound (Eng. high bound unit). The min. limit (-1.0) is called EU low bound (Eng. low bound unit).

- notion of a usual notification range, which corresponds to the difference between the max limit and the min limit defined for an analog type.
  In the previous example:
  The usual range of variation of the *AnalogType* is: 2 = (1.0 - (-1.0)),
- notion of notification threshold, which conditions the transmission of a notification to the client application: notification is transmitted if, and only if, the difference (in absolute value) between the value read and the last value sent is above this threshold.
  The threshold value of an analog type is calculated by applying the deadbanding value defined for the group to the usual variation range of this type.
  Deadbanding is a percentage variation between 0 (i.e.: 0%) and 1.0 (i.e: 100%).
  To summarize, for an analog type, the notification condition is as follows:
  ABS (Value read - Last value sent) > Deadbanding * (Max. limit - Min. limit).

NOTE: All notifications are sent if the deadbanding is 0% (default value). In the previous example: If the deadbanding value assigned to the group is 10%, the notification threshold for the *AnalogType* is:
0.2 = 0.10 (deadbanding) * 2 (usual variation range).
This means that only those variables in the group whose value varies by a difference of more than 0.2 (in absolute value) will be notified to the client application.

Installation of the Deadband in a client application

Description

Declaration of analog types: use of the Configuration tool.

- **Note**:
  "AnalogType" is the name given to the analog type by the user.

NOTE: 1. A maximum of 100 analog types may be defined.
2. You cannot modify the limits for an analog type after the server has started. Changes you make will not go into affect until after you stop and then restart the OFS server.

- Defining the value of the Deadband:
  The dead-banding Percentage associated with a user group can be set when the group is created (AddGroup primitive) or dynamically adjusted during the server session (PercentDeadBand property).
- attaching an item to an analog type:
  The general syntax of an item has an optional parameter which states the analog type to which it belongs, and so informs the OFS server of its notification threshold.
  The syntax of an item with analog type is as follows:
  <item> ::= <driver name>:<API address>!<variable name>[@<analog type name>]
  Note:
  The space before the @ character is optional.
  Example of an item definition: "FIP01:0.31.0!%MF330 @AnalogType".

Comments:

- it is possible to have the same item twice in the same group (e.g. "%MF330") with and without the analog type suffix (" @AnalogType"), so as to be able to compare the effect of dead-banding for notification filtering.
- It is possible to have items of different analog types in the same group (i.e.: several analog types referenced in the same group).

Section 6.8
The Diagnostic folder

The Diagnostic Folder

Description

This folder defines the settings for diagnostic feature. Here is an illustration:

The table describes the fields of the OFS configuration tool:

Field	Description
OverWriteLog file	0verwrite the log files every time the server is started.
Diagnostic	Activates the corresponding log trace file. When this option is selected, a browse window allows to select a log file. You can set the maximum size of the Diagnostic log file (in Mb) by setting the **Max Size (Mb)** value. The default value is 1000. When the Diagnostic log file size exceeds the value, the file is discarded.
Symbol manager	Adds data on the symbol files in the **Diagnostic** file.
Networks	Activates the corresponding log trace file. When this option is selected, a browse window allows to select a log file. You can set the maximum size of the network log file (in Mb) by setting the **Max Size (Mb)** value. The default value is 1000. When the networks log file size exceeds the value, the file is discarded.

Field	Description
Query generator	Adds data on the generated queries in the networks file.
Server mode	**Hidden**: The server is invisible on screen. **Control**: The server is indicated by an icon in the task notification bar, only the reduced menu (About and Exit) is accessible by right clicking. **Diagnostic**: A complete set of diagnostic windows is displayed when the server is running, including a plotting window displaying messages. **Verbose**: The plotting window displays additional information messages. The rest is identical to "Diag." mode. This data can be used to solve a temporary anomaly for the support or an experienced user. It is not recommended to use it in the operating phase, as there may be a large number of messages.

Section 6.9
The Simulator folder

Aim of this Section

The aim of this chapter is to introduce the simulator of the OFS product.

What Is in This Section?

This section contains the following topics:

The Simulator folder

Presentation
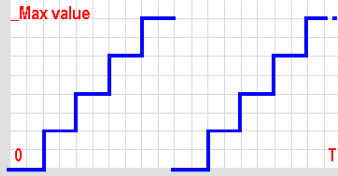
If the alias has been configured with the simulation property *(see page 88)*, any variable created on these devices is simulated by the server.

Description

This folder defines the value variation to be applied to all simulated variables by the server.

Field	Description
Simulator mode	Allows you to select the simulation mode.
Initial values	**Random**: the variables are initialized to random values. **Zero**: all variables are initialized to zero.
Max value	Maximum value for the simulated variable. Range: [0..32767]. The variable is increased at each cycle, and returns to 0 when the maximum value is reached (cyclically). Boolean variables are inverted, floating point variables are increased by 0.3.
Notification adjust	N=1: simulated variables are updated at the same update rate as the group and each time a reading is performed (sync or async) $1 < N \leq 10$: at each period there is one chance in N that the simulated variable will be modified. There is no correlation between different declared variables; their values change individually. The decrease in the probability value (increase of N) reduces the number of notifications, and thus reduces the CPU load on the machine.

Individual simulation of a device

Description

This feature allows the server to simulate a missing device.

The choice between accessing a real device and device simulation is made, device by device, in the properties page of the device *(see page 88)*.

The conditions for use are the same as for simulator mode (setting the parameters for animation of the variables from the simulation folder of the configuration tool).

Section 6.10
The Symbols folder

The Symbols folder

Description

This folder provides a list of file name extensions associated with the symbols tables. This list may be completed by new extensions (up to a maximum of 12 suffixes). You must attach the new suffix to one of the existing 8 types of symbol files.

In addition to the 10 predefined and no modifiable first symbol extensions, you can add or delete entries by using the contextual menu (right-click) over the table.

The extensions are saved to the memory, even when the server is uninstalled and/or reinstalled. However, for this to happen, one condition must be observed: they must be entered when the grid contains extensions which have already been set by the server. If you are starting out with an empty grid (server never installed), they risk being overwritten during installation of the server.

Illustration:

Index	Suffix	Symbol Type
☐ Symbol table file		
📄 New entry		PL7 exported symbols table file
		PL7 exported symbols table file
✖ Deleted entry		ModSoft exported symbols table file
3	PRJ	Concept project file
4	CCN	Concept exported symbols table file
5	CSV	Comma Separated Values file
6	FIS	Comma Separated Values file
7	STU	UnityPro project file
8	XVM	UnityPro exported symbols table file
9	XSY	Located exported symbols table file

Define or edit an extension

To define or edit an extension, you have to follow the procedure described below:

Step	Action
1	Create a new entry.
2	Enter an extension, then press **Return**.
3	Double click on the corresponding field in the **Symbol Type** column. *Result*: A list appears.
4	Select a file type.

Section 6.11
The PLC Software folder

The PLC Software folder

Description

This folder allows you to define the settings in relation with PLC Software:

⊟ Dynamic consistency		
Cyclic consistency check rate (s)	10	
⊟ Project files options		
Unity project files (*.stu)	☐ Allow multiple connections to the project	
Symbol table	☐ Reload Database	

The following table describes the available options:

Option	Function
Dynamic consistency	**Cyclic consistency check rate (s)**: If this option is checked in device properties, the database reloading and period parameters are used to define the action on the database and the frequency of the consistency check.
Project tiles options	**Unity project files (*stu)**: With this option, the OFS server will free the Unity project (STU file) when it has accessed the database to look for variable properties. A remote Unity client can then open and modify this project, which is impossible when this option is not checked. If this modification is downloaded to the PLC, the cyclic consistency check of the server detects the modification and re-learns the project. In the case of this utilization, it is thus strongly recommended to select the option available in Unity Pro V2.0 **Tools → Options → General → Project autosaving on download**. Without this, the user must manually save the project on downloading in order for the application loaded in the PLC to be consistent with the project (STU file). **Symbol table**: If the 'Reload Database' option is checked, the server reloads automatically the new database in case of inconsistency detection.

Section 6.12
The Communication folder

The Communication folder

Presentation

The Communication folder page gives access to general synchronization parameters for data exchange with the devices and polling frequency on reception.

Description

This folder defines the settings for communication server feature:

─ Communication settings		
Communication overrun behavior	⦿ Items to Bad Quality	○ Rate adapt
Advanced feature	☑ Quick Item validation	☑ Quick SetActive State
Specific Items	☑ Enable OPC extensions	☑ Change PLC Status
Modbus Serial Driver	☑ Disable driver configuration on time out	
Modbus Request Optimization	☐ Enable discontinuous memory access	
XWAY for Push Data Gate	0	

The table below describes the fields relative to the communication folder:

Field	Description
Communication overrun behavior	Saturation occurs when a group's update period is too low and the server is unable to update all the items in the programmed period. **Items to Bad Quality** option: The items must be read with the group's update period (default parameters); otherwise, the quality of the unrefreshed items is declared bad. **Rate adapt** option: The items are updated even if a saturation appears. Here, the update period is not guaranteed and the client is notified as to the server polling period. The frame time-out for assigned devices must be sized in order to allow a complete update of the items. The group update period remains the same but enables the group items to be read in the frame time-out. After this time, the quality level switches to bad. In any case, the quality depends on the time-out period. If you want to read as many items as possible, the time-out period must be longer than the group period.

Advanced feature	If the option **Quick Item validation** is checked, item validation makes no physical access to the corresponding device (request emission). If the device is not accessible, this avoids the server having to await communication time out. If the option **Quick SetActive State** is checked, the interface IOPCItemMgt: **Quick SetActive State** () performs a Synchronous Read and an immediate notification to speed up the first values acquisition.
Specific Items	**Enable OPC extensions**: Enables/disables the specific items *(see page 193)*. **Change PLC Status**: If the **Enable OPC extensions** option only is checked, it allows the server to change the PLC operating mode (RUN/STOP)
Modbus Serial Driver	When a device is in time out on a Serial Modbus Network due to device missing, or frame time-out is greater than configured frame time-out, OFS server sends commands to the Serial Modbus Driver in order to adapt the baud rate that generates unexpected workload. When checking **Disable driver reconfiguration on time out** option, the reconfiguration is skipped.
Modbus Request Optimization	When multiple registers with nonadjacent addresses are accessed on the same equipment, OFS performs optimization: a single request is generated to read the entire zone that includes all registers, and then only the requested values are extracted. If the option **Enable discontinuous memory access** is checked, OFS does not perform any optimization, and instead accesses each register individually. In this way it can avoid accessing restricted zones of certain Power devices such as SEPAM.
X-Way for push data	**Gate**: value of the reception gate on which the server will receive data from the remote device. Range: [0..255]

Note: If the **Change PLC status** option is not checked, any client application attempting to write on the #PLCStatus item receives an system message.
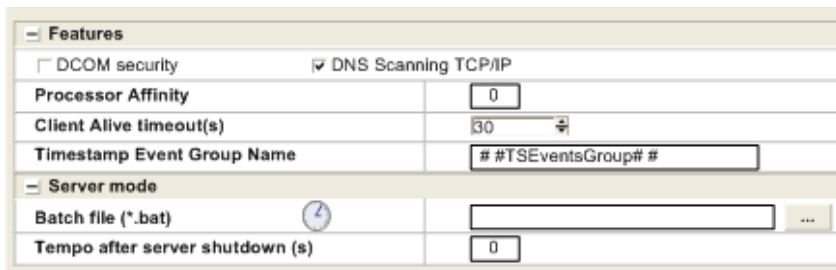
Section 6.13
The Options folder

The Options folder

Description

This folder allows the optional functions of the OFS server to be activated:

Field	Description
DCOM security	Enables/disables **DCOM security** *(see page 51)*.
DNS Scanning TCP/IP	Authorizes the server to use DNS to identify the PLC.
Processor Affinity	This setting gives to the OFS server which processor is used to run all OFS server process tasks.
Client Alive timeout(s)	Refer to Client-alive Service *(see page 381)*.
Timestamp Event Group Name	Allows to configure the OPC group name reserved for Time stamping function *(see page 140)*. Default name is ##TSEventsGroup##.

Shutdown batch file	If a *.BAT* file is indicated, it will be run at the time of the shutdown request from the OFS server. Shutdown of the OFS server will occur after running the *.BAT* file. If the batch process still has not finished after 10 seconds, the batch will be interrupted, and the server will be effectively stopped.
Tempo after shutdown request	If a timeout value is given, the effective shutdown of the OFS server will be delayed by the value of the timeout. Range: [0..32767] seconds. If moreover, a *.BAT* file is indicated, the timeout will be armed after running the batch process (limited to 10 seconds).

Section 6.14
Configuration database management

Database Management

Description

Certain third party software programs (OFS Manager for example) may attempt to modify the configuration database at the same time as, and by taking priority over the configuration tool.

The two following scenarios may arise:

- the configuration tool is launched when the configuration database is in the process of being modified by a third party program,
  In this case, an information message is displayed when the configuration tool is launched indicating that the tool will run in "Read only". The configuration tool will then not be able to modify the configuration database.
- the configuration database is modified by a third party program when the configuration tool is running,
  In this case, when the configuration tool first attempts to modify the configuration database (user clicks "OK" for example) an information message is displayed indicating that the configuration tool is going to switch to "Read only". The configuration tool will then no longer be able to modify the configuration database.

In both cases, in order to be able to modify the configuration database using the configuration tool, you must close the latter and relaunch it until the configuration database becomes unprotected.

Section 6.15
Compatibility with Previous Versions of Configuration Tool

Compatibility with the Previous Version of the Configuration Tool

Description

If an older version of the configuration tool has previously been installed, it will be detected automatically and the configuration parameters will be restored.

The first time the Configuration tool is executed, a dialog box will appear inviting you to restore these configuration parameters:

Illustration:
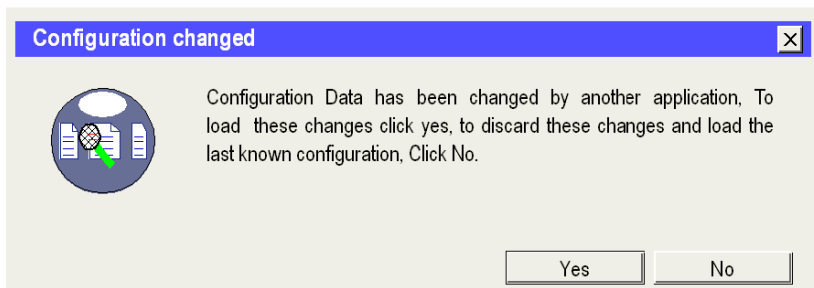
If you answer:

● YES: the previous configuration parameters are restored,
● NO: the previous configuration parameters are lost.

Section 6.16
Time Stamped Events Configuration

Aim of this Section

The aim of this section is to describe the uses of the time stamped events system.

For more details about time stamping, please refer to the **System Time Stamping User Guide** provided inside OFS DVD *(see page 13)* (directory: **\Extras\System Time Stamping User Guide**).

What Is in This Section?

This section contains the following topics:

Time Stamped Events System

Generalities

Some applications manage events which occur quickly. Processing these events via the regular flow of data (standard acquisition by polling, then associating with an alarm in the SCADA and displaying in the alarm viewer) induces a lost of events.

When used to associate with the different IOs (sources) mentioned here below, OFS V3.60 is able to retrieve the events (value and time stamp) which are buffered in the source, and to notify the client with the memorized value and their time stamp provided by the source. This mechanism allows to manage events which occur faster than the polling rate from OFS, in order not to lose any events.

Required Products

The different products involved in the source time stamping system are:
- the sources of the TS events: BMX-IOs (BMX\BME-CRA, BMX-ERT) and the PLC
  NOTE: BMX-ERT can be in a local M580\M340 rack or in an M580\M340 drop.

- Client: Vijeo Citect SCADA (including IO server, alarm server, and Operator Clients) or any OPC client acting as a SOE viewer
- OFS server
- PLC station configuration tool: Unity Pro (version 7.0 or later)

NOTE: To get the minimum versions of the concerned products, allowing to manage the time stamping features, refer to **System Time Stamping User Guide** on the DVD.

Main Use

This illustration shows a typical configuration:
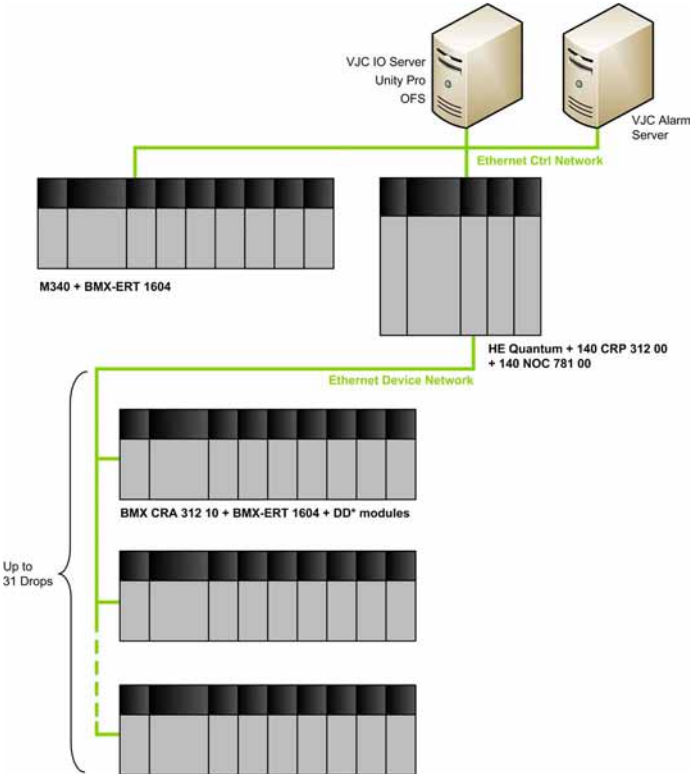
The 140 NOC 781 00 can be replaced by an external router. The VJC alarm server can run on the same machine as OFS and VJC IO server.

For BMX-ERT time synchronization, a GPS receiver is required.

For time synchronization (VJC, OFS and event sources other than BMX-ERT) a NTP server is required.

The NTP server must be synchronized on a GPS.

The PC running OFS and VJC I/O server must be also synchronized by NTP server.

NOTE: Only one viewer of sequence of events can be active at a time.

System Runtime Sequence Overview
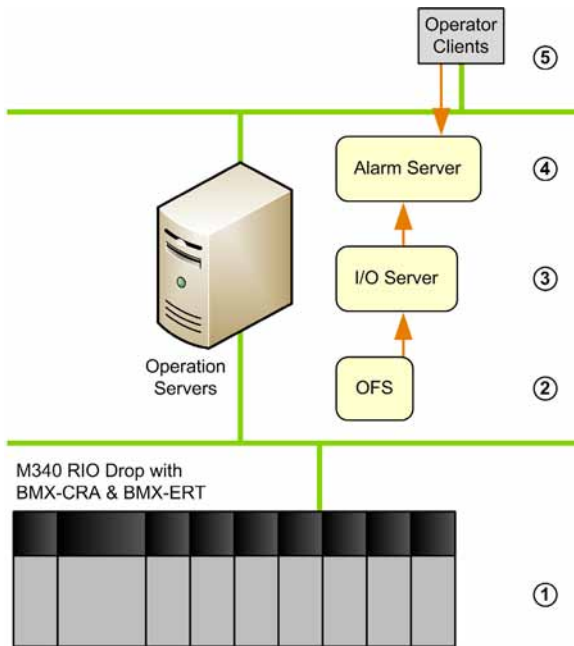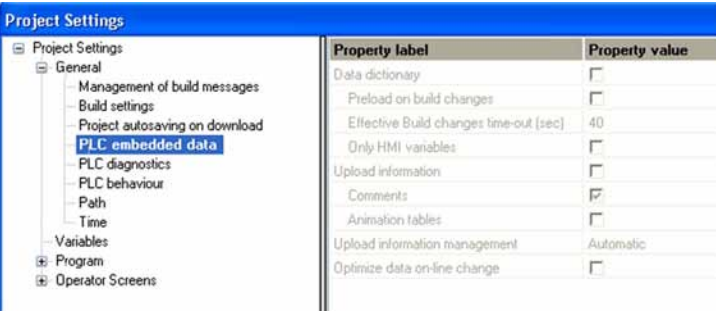
This following table shows the process of the system runtime sequence:

Step	Action
1	Detect and time stamp local I/Os changes. Store TS I/O changes in local event buffer.
2	OFS reads the source event buffers. Send the variables to the I/O server.
3	Send the variables to the alarm server (value, source time, and quality).
4	Process the received variables. Use the received variables to evaluate the alarms.
5	Get the alarms from the alarm server. Display the alarms in the alarm viewer.

NOTE: OFS sends the TS event in the standard OPC-DA notification format.

Unity Pro Configuration

This following table shows the Unity Pro procedure:

Step	Action
1	In Unity Pro version 7.0 or later, select **Tools → Project Settings**. Result: **Project Settings** window appears:
2	Expand the general menu in the **Projects Settings** window, and select the **PLC embedded data** setting.
3	Check the **Data dictionary** box.
4	Click **Apply** to save changes, or click **OK** to save changes and close the window.

This illustration shows the Unity Pro procedure:

NOTE: To configure the different source events, refer to the proper documentation or to the **System Time Stamping User Guide** on the DVD.

OFS Configuration

Configure the TS events *(see page 135)*.

Time Stamped Features

TS Events Features

The custom OPC property is linked to OPC items with the following definition:
- **Description**: time stamped event support
- **PropertyId**: 5012
- **Comment**: true if the variable is configured as time stamped event
- **Type**: VT_BOOL
- **Values**: 1 if the item is a time stamped event and 0 if not
  NOTE: For Unity Pro variables configured as TS events, the property value is set to 1 and used by the OPC client to determine which items can be added in the event group.

TS Events Configuration

The only devices that can be configured with an alias are PLC devices. Event sources other than PLC are not configurable. They are discovered at runtime by reading a **devices description** table embedded in the PLC.

A time stamped events appears in the device panel of the current alias.

This figure shows the configuration of the time stamped events:
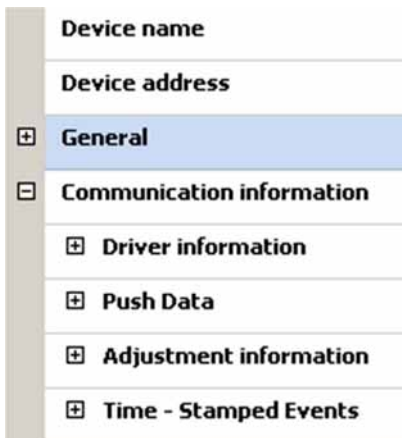
When the **Time - Stamped Events** topic is deployed, you can access different parameters described in the following table:

Property	Values	Default Values	Description
Service support	Checked or unchecked	Unchecked	When checked, the time stamped service is enabled: • time stamped items can be added to the event group • access to event sources buffer is performed at the per-family **Polling Rate** (all event sources linked to the current PLC are accessed)
BMX\BME-CRA Polling Rate (ms)	250...4000 ms by step of 50 ms^2	1000	Polling rate for event read request sent to the BMX\BME-CRA.
BMX-ERT Polling Rate (ms)	250...4000 ms by step of 50 ms^2	1000	Polling rate for event read request sent to the BMX-ERT.
(1) The value 0 is an acceptable value (no PLC event source polling). (2) The value 0 is an acceptable value (no BMX-ERT or no BMX\BME-CRA event source polling).			

When checked, Available property set the **TS Event Service** of the current device as available. In that case the following properties are enabled:
● **BMX\BME-CRA Polling Rate (ms)**: Rate at which `ReadEvent` requests are sent on a BMX\BME-CRA event channel. A unique event channel is opened to a BMX\BME-CRA for all TS variables of the BMX\BME-CRA sub module (except BMX-ERT):
   ○ If value is 0, then no BMX\BME-CRA Event buffer read is performed (it can be used to disable temporarily the BMX\BME-CRA Event sources when tuning the configuration or definitively if the TS Event function is not required for BMX\BME-CRA family).

● **BMX-ERT Polling Rate (ms)**: Rate at which `ReadEvent` requests are sent on a BMX-ERT event channel:
   ○ For in-rack BMX-ERT, a TDA channel is opened for each ERT, in addition to the M340 device Max Channel it belongs to.

   ○ For remote BMX-ERT, a TDA channel is opened for each ERT, in addition to the BMX\BME-CRA event channel the ERT belongs to.

   ○ If value is 0, then no BMX_ERT Event buffer read is performed (it can be used to disable temporarily the BMX_ERT Event sources when tuning the configuration or definitively if the function is not required for BMX-ERT family).

OFS Runtime

When a time stamped item is added to the event group *(see page 191)*, and if **Available Service Support** property is checked for the alias, OFS builds the TS events source list attached to the PLC and sends an identification request to each event source device.

The event service availability status and event source devices attached to a PLC are displayed in the **Network Info** window, in the **Devices** tree-view, under the correct device.

Detailed configuration:

```
Devices
  MBT : 192.168.2.110 = MBP1
    UNITY device
    Device Identity : 140 CPU 651 60
    Device Version : 3.10
    Device Status : GOOD
    Frame TO : 1000 ms
    Device TO: 5000 ms
    Time-stamped events Unity service support : Yes
    Time-stamped events configured service support : Yes
      PLC Polling rate : 1000 ms
      BMX-CRA Polling rate : 1000 ms
      BMX-ERT Polling rate : 1000 ms
    Event Sources Nb : 2 (State is Active)
      MBT:192.168.2.116 =
        BMX-CRA device
        Device Identity : BMX CRA31210 MC
        Device Version : 0.8
        Device Status : GOOD
        Frame TO : 1000 ms
        Device TO: 5000 ms
        Max Pending Req Used =1 UserDefined = 0
        Max Channel: Configured=1 Used=1
        Max Waiting Req : 200
        Request Length: 1021
        Req Sent+Rcv : 4
        Bytes Sent : 1077
        Bytes Rcv : 2116
        Ref Count : 1
        Nb of Error : 1
        Last Error Code: 0
        State Cnt : Good = 1 Uncertain = 0 Bad = 0
        Nb of Waiting Req: 0 (max reached 0)
        Best Access Time : 15 ms (1)
        Worse Access Time : 31 ms (2)
        Last Access Time : 15 ms (4)
        Average Access Time : 0 ms (4)
        Event service available :No
        Event source reservation :Yes (Reservation Id : 0)
        Event source polling rate : 1000 ms
        Nb Events Read : 82
        Nb Events Acked : 0
        Nb Events Discarded : 0
        Event diagnostics flag : 0
        Events buffer fill : 6 %
      MBT:192.168.2.117\0.4.0 =
        BMX-ERT device
```

The following table describes the properties displayed in the **Network Info** window above, for the MBP1 device:

Field	Value	Description
Time Stamped Events Unity Pro Service Support	Yes/No	Yes: ● If **Project Settings → General → PLC Embedded Data → data dictionary** is checked in Unity Pro application. ● If **Project Settings → General → Time set Time Stamping Mode** property value is set to SYSTEM.
Time Stamped Events Configured Service Support	Yes/No	Yes: if the time stamped events service support is checked in the configuration tool.
BMX-CRA Polling Rate[1]	0...4000 ms	Value configured in the **BMX-CRA Polling Rate** box).
BMX-ERT Polling Rate[1]	0...4000 ms	Value configured in the **BMX-ERT Polling Rate** box).
(1) If the time stamped events configured service support value is No, the values are not displayed.		

The table describes the properties displayed in the **Network Info** window above, in the **Event Source Nb** leaf:

Field	Value	Description
Common Device Properties		
State	**Active/Inactive**	Active: each event source buffer is currently polled
Source	**BMX-CRA/BMX_ERT**	Source type name
Device Identity	BMX-CRA31210MC	
Device Version	0.8	
Device Status	**GOOD / BAD / UNCERTAIN / UNKNOWN / MISSING**	
Frame To	1000 ms	Configured frame timeout
Device To	5000 ms	Configured device timeout
Max Pending Req	Used=1/User Defined=0	Configured maximum pending request
Max Channel	Configured=1/Used=1	Configured maximum channel
Max Waiting Req	200 ms	Configured maximum waiting request
Request Length	1021	Request length
Req Sent + Rcv	4	Total number of sent and received requests
Bytes Sent	1077	Total number of bytes sent
Bytes Rcv	2116	Total number of bytes received

Field	Value	Description
Ref Count	1	Device reference counter
Nb of Error	1	Number of detected communication errors
Last Error Code	0	Last detected communication error code
State Cnt	Good=1/Uncertain=0 Bad=0...100%	Gives the number of good or uncertain or bad states
Nb of Waiting Req	0	Number of waiting requests or maximum number reached
Best Access Time	15 ms	
Worse Access Time	31 ms	
Last Access Time	15 ms	
Average Access Time	0 ms	
Event Source Specific Properties		
Event service available	No	No: the device version is not compatible with the TS event function
Event source reservation	Yes	Yes: the Event source is currently reserved
Event source polling rate	1000 ms	Event source buffer polling rate: If service available is No, then the value is 0 ms
Nb Events Read	82	Total number of read events: If service available is No, then the value is 0
Nb Events Acked	0	Total number of acknowledged events: If service available is No, then the value is 0
Nb Events Discarded	0	Total number of discarded events (not notified due to inactive item): If service available is No, then the value is 0
Event diagnostics flag	0	Diagnostics value returned by the last Event buffer read: If service available is No, then the value is 0
Events buffer fill	6%	Percentage of buffer fill returned by the last Event buffer read: If service available is No, then the value is 0

NOTE: If time stamped events configured service support value is No or if time stamped events Unity Pro service support value is No, then the **Event Sources Nb** leaf is not displayed.

Events Sources Monitoring

At Runtime, it is possible to control the status of the connection with the different events sources using the specific items #PLCQualStatus or #PLCQualStatus2.

Event Group

Description

A reserved OPC group name is to be used at group creation for a TS event process. Each OPC client can create at most one OPC events group.

NOTE: The reserved OPC group name can be configured in the OFS Server Settings\Option\ Timestamp Event Group Name field of the Configuration tool *(see page 125)*
Default name is `##TSEventsGroup##`

Items adding:

- If the **Available Service Support** property for TS events is not set for a given item alias, that item is not added. An `E_FAIL` detected error is returned if **Available Service Support** property is not set and an explicit error message is displayed in OFS diagnostics window.
- It is not possible to add specific items. An `E_FAIL` detected error is returned in that case and an explicit error message is displayed in OFS diagnostics window.
- It is not possible to add the same item more than one time. An `OPC_E_DUPLICATE_NAME` detected error is returned if the same item is added and an explicit error message is displayed in OFS diagnostics window.
- It is not possible to add items associated to a same Alias (defined in the OFS configuration tool) in several events groups. An `E_FAIL` detected error is returned in that case and an explicit error message is displayed in OFS diagnostics window.
- OFS connects to all the TS events sources linked to the current item alias and, if the group is already active, starts reading the event buffer of the entire event sources attached to each item PLC.

Group activation:

On group activation, OFS starts reading the event buffer of the entire event sources attached to each active PLC item.

NOTE: Any events that should be notified to inactive items, will be lost (and not forwarded to a client).

Item deactivation:

On item deactivation, the items of the group are deactivated in sequence. When the notification task encounters an inactive item, it discards the value that was to have been notified.

When a PLC event active reference counter is 0 (an event active reference counter gives the number of active items linked to a given device, it is incremented on each activation and decremented on each item deactivation), OFS stops reading the event buffer of the entire event sources attached to it.

Group deactivation:

When the group is deactivated, all items in the group are deactivated. The item deactivation rule described above is applied for all the items of the group.

Items removal:

If items to be removed are active in an active group, the item deactivation rule described above is applied for all the items of the group and then the items are removed.

Group removal:

If the event group is active, the items removal rule described above is applied for all the items of the group and then the group is removed.

Not supported operations:

The following operations performed on the event group return an `E_FAIL` detected error immediately:
- `IOPCGroupStateMgt::SetName()`
- `IOPCGroupStateMgt::CloneGroup()`
- any **IOPCSyncIO** method
- any **IOPCASyncIO2** method

OPC-DA notification:

Reading the event source buffer may return a series of data quite different from standard real-time data access. That is, a series of value changes can refer to a same OPC item.

The OPC client must process the `OnDataChange()` returned arrays (handles, values, quality and time stamp) in the array order. Identical handles can be returned.

Configuration Change On The Fly (CCOTF):

OFS monitors an event modification (OMC) to detect:
- any TS event property modification (add \ remove) of a variable
- any TS variable renaming
- any event source topology modification (add \ remove of event source)

Trouble Shooting Time Stamped Events Application

To diagnose a time stamped application, it is possible to use the OPC test client, delivered with OFS product, according to the following command line: `Ofsclient.exe –l[logfilepath]`

`Logfilepath`: full path of the logfile dedicated to time stamped events. The log file will contain the logs of all source time stamped events which occurred in the system.

NOTE: In the above command line, OFS client assumes that the Timestamp Event group is `##TSEventsGroup##` (default value).
To set the actual configured Timestamp Event group , the following command line is required:
`Ofsclient.exe –l[logfilepath] -g[TSEventGroupName]`
`TSEventGroupName`: name of the actual configured Timestamp Event group

Chapter 7
The OFS Manager Tool

The OFS Manager

The OFS Manager is a tool for troubleshooting and adjustment that works ONLY with OPC Factory Server (locally or remotely) or the OFS simulator. Do not use it when the application is operating.

OFS Manager can be used to get informations on status of server when OFS operates in hidden mode or on a remote PC. All informations usually provided by the debugging interface of the server are then available through OFS manager interface.

The OFS Manager adjustment functions are only accessible using the OFS Manager.

Most of the modifications are memorized and are final. However some changes (debugging mode) are valid only for the current server instance. By closing and opening the server, the changes made are lost.

Connection to the server: **Server** -> Connect menu

Managing aliases: Select an alias and then use the **Alias** menu or the **right mouse button**:

- Changing an alias: Used to change the network address or the symbols table filename
- Deleting an alias: Used to remove an alias from the list

Symbols table:

It is possible to ask the server to close a symbols table file that is already open and to reopen it. No change will be made to the already existing items. On the other hand, the list of symbols will be updated (e.g.: for the OFS navigation interface). Only the new symbols shall be included, but if a symbol has changed address, it will keep the old address.

For that, first select the symbols table filename from among the filenames present in the Symbols Table window. Next use the **Symbols Table** menu or the **right mouse button**. The addition of a symbols file will be accounted for. However, if a filename is changed it will only be accounted for in the next OFS session.

Managing the debugging mode:

The OFS server has 3 debugging options, which are the following:

- Verbose mode: This is a full display mode, messages are shown in the Server Diagnostics window.
- Symbol mode : This is used to display additional information messages about the symbols table.
- Request: This is used to display information about network requests generation in the Server Network window.

Managing log files:

The OFS server can save messages in two different log files (one for the main Diagnostics window and one for the Network window).

With the OFS Manager, when the server is running it is possible to open or close any one of these files.

To open or close the file, select the file in question in the Log Files window and use the **Log** menu or the **right mouse button**.

Viewing information:

The information display is static by default (no refreshing).

To refresh the view, use the **View** -> **Refresh** menu.

To automatically refresh the view, use the **View** -> **Auto Refresh** menu. By default the screen is refreshed every 1 second. This frequency may be modified with the **View** -> **Options** menu.

If the **Status** window is selected, the OFS Manager displays general information (contents identical to that of the Status window of the debugging interface).

If the **Protocols** window is selected and then a protocol (OFS NET MANAGER), the OFS Manager displays the statistical information relating to the chosen protocol. This information is exactly the same as that appearing in the Network windows of the server debugging interface.

For each device connected to the server, certain debugging information may be viewed. Select the equipment below its protocol, in the left hand side of the OFS Manager. The information displayed is the same as that appearing for each device with the server debugging interface.

Reloading function:

For each device associated with a Concept or Unity Pro project, the symbols table may be manually reloaded using the Device->Reload and Update menu. The menu is activated by selecting the device in the list of equipment.

View detected errors and diagnostics messages :

As long as you are connected to the OFS server, all detected error messages are displayed in the Errors text area.

If you are interested in all messages, you may enable the Diagnostics text area from the **View->Debug Messages** menu.

Saving information:

From the File -> Save As menu, you may, at any time, save all information in a .txt file stored by the OFS Manager (Aliases list, Messages, Counter values,). This is the only way to save this kind of information (impossible with the server debugging interface).

Viewing server information:

The **Server Info** section is used to view the server name, the type of product, the version and its operating mode (normal or simulated). This is especially useful when the server is operating in hidden mode and/or in service mode (without interface).

NOTE: To avoid any conflict on the Alias creation, it is recommended to close the configuration tool before using the OFS manager.

Chapter 8
The OFS Test Clients

Aim of this Chapter

The aim of this chapter is to present the test clients provided with the OFS server.

What Is in This Chapter?

This chapter contains the following topics:

OFS C++ OPC DA Client

Description

The OFS client is an OPC client supplied with the OFS server as a **test tool**. It is an OPC client in compliance with the OPC DA V2.0 standard.

Installation

To install it on your machine, select the "Sample OPC client" option during installation of the full extension or the option "OFS server test client" during the installation of the remote extension.

Main uses

The C++ OPC DA client is mainly used to verify the configuration and communication of the whole system: OPC Client / OFS Server / PLCs.

Connect as Client to Server

The following table describes connecting as Client to Server:

Step	Action
1	The following window appears the test client opens:
2	Select the Schneider-Aut.OFS server for Prog ID.
3	Enter the name of the computer of the OPC Server (leave empty if the OFS client is on the server).
4	Click on OK to connect to the server.

The .NET OPC DA/OPC XML-DA Client

Description

The .NET OPC DA/OPC XML-DA client is an OPC client that may used as a **test tool**. It is an .NET OPC client that lets you connect to the OFS server via OPC DA or via the SOAP/XML protocol compliant with the OPC XML-DA V1.01 standard. To install it on the machine, select the .Net station option when installing an extension.

The main uses and characteristics of this client are identical to those of the C++ OPC DA *(see page 148)* client.

The .NET OPC DA/OPC XML-DA client lets you connect to the OFS server via OPC DA or via HTTP OPC XML.

The test clients provided by OFS must not be used for critical functions.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION
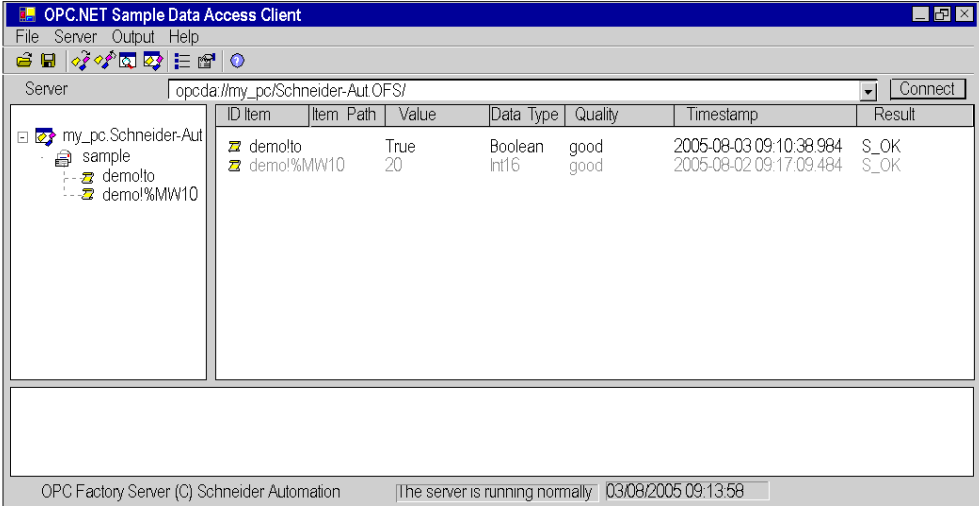
Limit or restrict access to authorized persons by providing appropriate and independent protection via your application or infrastructure.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Connecting the Client to the OPC Server

The following table shows how to connect the .NET OPC DA/OPC XML-DA client to the OFS server via OPC DA:

Step	Action
1	Launch the test client from **Start → Programs → Schneider Electric → OFS → OFS test client → Sample Net - OPC XML client**.
2	When the .NET OPC DA/OPC XML-DA client is started, the following window appears:
3	Click on <Browse> for the list the servers that may be reached. The following window appears on top of the opening window:

Step	Action
4	When the .NET OPC DA/OPC XML-DA client is started, the following window appears:

Connecting the Client to the Site Server

The following window is the result of a connection to the OFS server. It allows you to access certain items via OPC XML-DA:
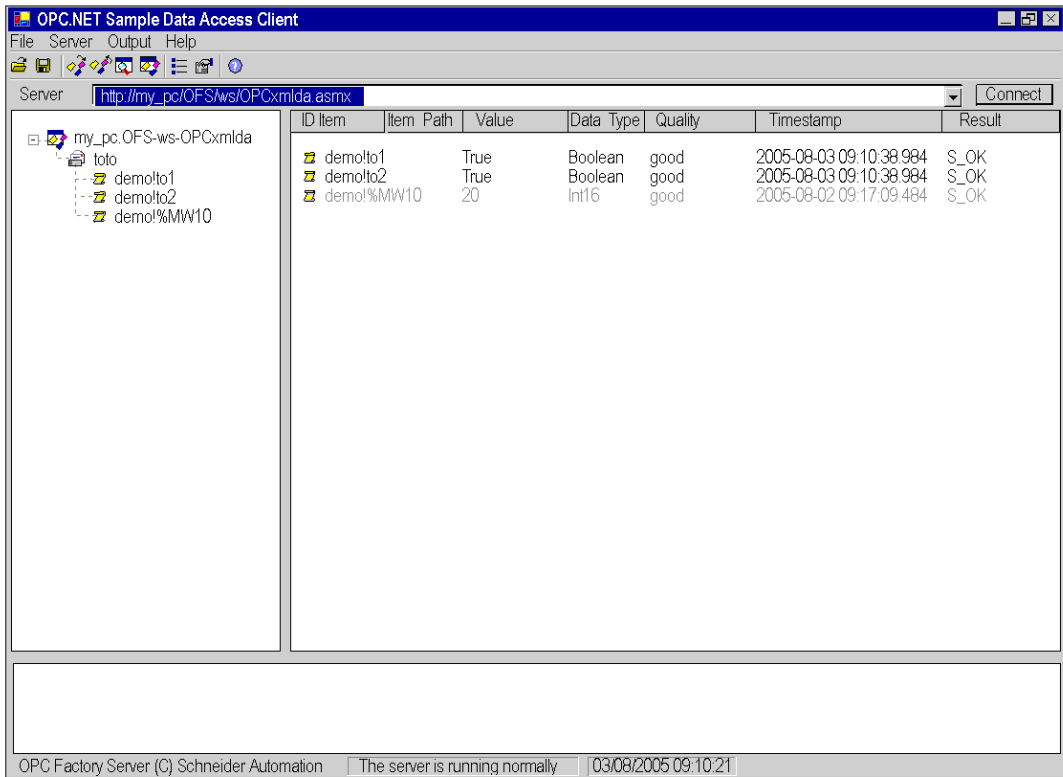
In this case, choose a connection of type http://stationname/Website/OFS/ws/OPCXMLDa.asmx to connect directly to the server of the OFS site.

Chapter 9
The Diagnostics Screens of OPC Factory Server

OPC Factory Server

Description

The OPC Factory server screens are used to view:

- the server's communication status (Server Status),
- the server's diagnostic window (Windows Diagnostic),
- the information screen about the variables configured on the server (Varman Window),
- the information screen about the server networks (Network Window).
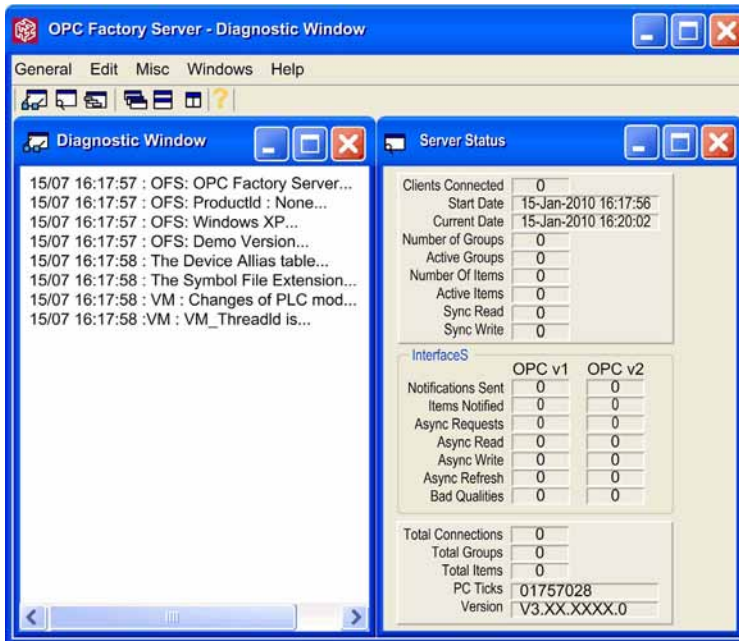
NOTE: The characters "XX.XXXX" correspond to your own version.

Network Window

In diagnostic mode or extended diagnostic mode, the server provides the list of active frequencies corresponding to the various frequencies of the declared groups. For each group, it provides the number of declared items and the number of network queries generated.

To do this, open the **Network windows** window from the **General** menu.
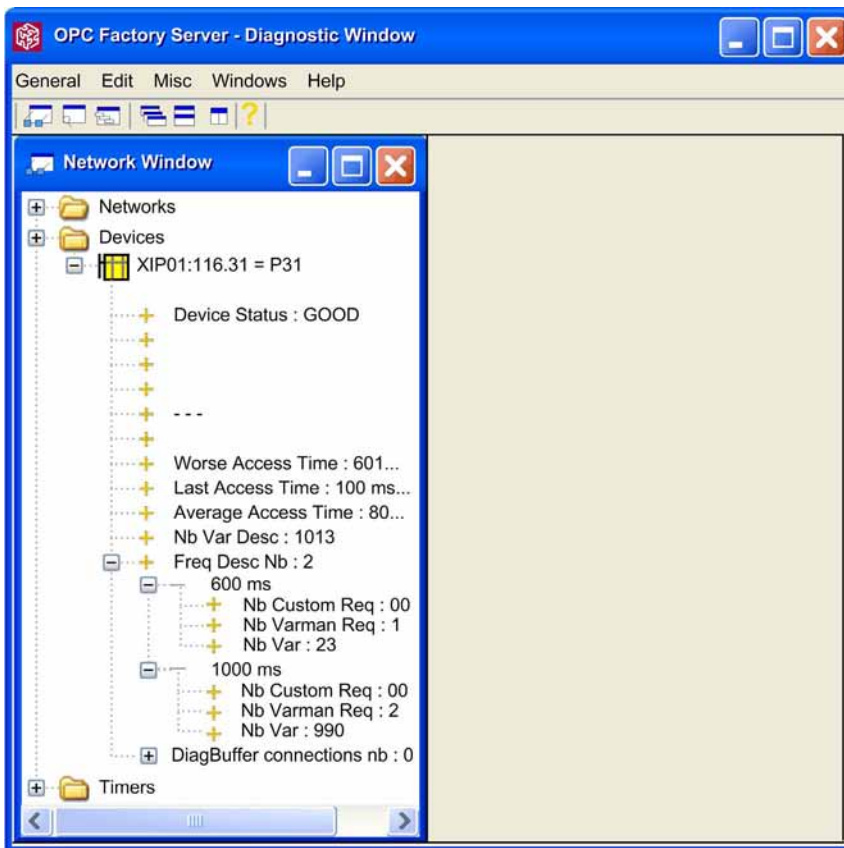
Chapter 10
The OFS Simulator

Simulator mode

Description

Simulator mode enables the user to test the client OPC application without any PLC being present. It provides a simple animation of all created variables and is identical to the actual Server.

The server can be started in simulator mode in two ways:

- by selecting the "OFS Factory Server Simulator" shortcut created during installation. It launches OFS.exe with the "-simu" parameter,
- by checking the "simulator mode" option in the "options" folder of the configuration tool.

When the server is started in simulator mode, no license code is required.

The animation of simulated variables can be set in the configuration tool, under the Simulation folder.

NOTE:
- As all the variables are simulated, there is no link between an item which is actually related to a table of elements (bits, words), and the items related to the individual components of this table,
- in simulation mode, there is no way to determine the maximum supported frame length for a given device,
- It is possible that when using the real device an item that was READ_WRITE in simulation mode becomes READ_ONLY in real mode,
- Concept Boolean variables that are located in State Ram in the register areas (3x or 4x) are actually simulated as bytes and not as Boolean values.

Chapter 11
The OFS Server WEB Site

Aim of this Chapter

The aim of this chapter is to provide an introduction to Web site of the OFS product.

What Is in This Chapter?

This chapter contains the following topics:

Home Page of the OFS Web Site

At a Glance

The home page is used to access the service pages on the site:

- **Diagnostics**,
- **Monitoring** (edit, read/write data).

Accessing the Home Page

Step	Action
1	Open your usual browser.
2	Enter the name of the Web Site machine using the following syntax: http://'name of the Web service machine'/OFS. **Result**: The OFS server home page looks like this: The Documentation link allows you to access and open Web services documentation.

Data Editor Page

At a Glance

This page is used to create animation tables containing lists of PLC or device variables to be viewed or modified.

> ## ⚠ WARNING
>
> **UNINTENDED EQUIPMENT OPERATION**
>
> Limit or restrict access to authorized persons by providing appropriate and independent protection via your application or infrastructure.
>
> **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Illustration

View of the Data Editor page (data editor) from an OFS server:
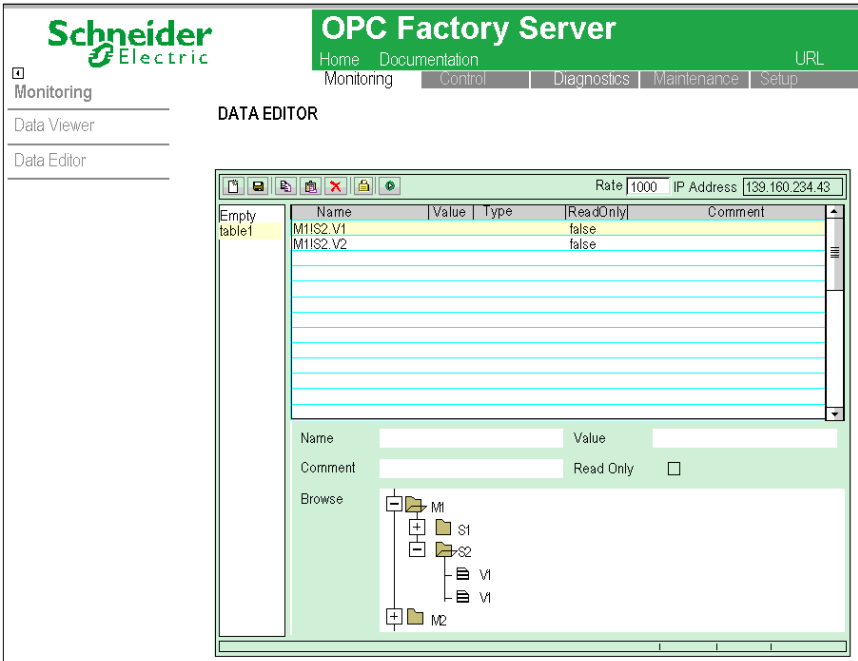
Description of the data editor buttons:

In sequential order:

- create a new table of variables,
- save a table, password protected (this function is not supported),
- copy the selected table or the selected variable,
- paste the copied table or the copied variable,
- delete a table or a variable,
- change the password (this function is not supported),
- start or stop the animation.

NOTE: write operations are not supported.

Double-click in the table to show or hide the pane for editing variables. To add a new variable, fill in the **Name** field or select the variable in the browse section by expanding the **OPC XML DA** tree. Then click **OK**.

OFS Diagnostics Home Page

Diagnostics Page

This page displays the status of the OFS server.

Illustration

The Diagnostics home page is the following:

NOTE: The characters "XX.XXXX" correspond to your own version.

NOTE: you can stop or start the animation by clicking the 'Stop Animation' or 'Start Animation' button.

Chapter 12
The OPC UA Tools

Overview

This chapter introduces the tools for using the OPC UA wrapper.

What Is in This Chapter?

This chapter contains the following topics:
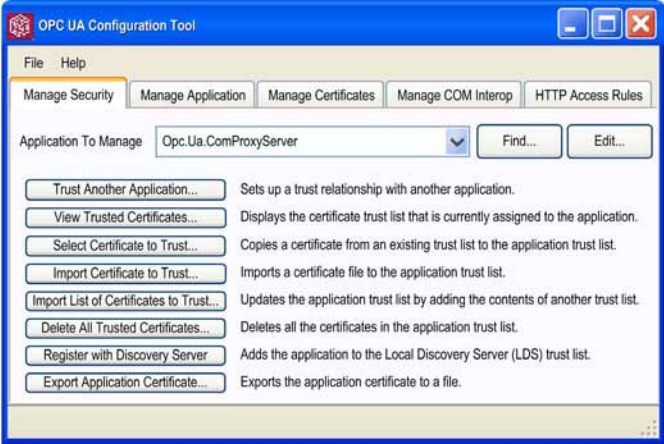
OPC UA Configuration Tool

General

OPC UA configuration tool is used to manage OPC UA applications, security settings, certificates, communications and HTTP access.

Installation

Follow this procedure to install the OPC UA configuration tool product:

Step	Action
1	Select the OPC UA wrapper checkbox in the options at the time of OFS installation. It installs the OPC UA configuration tool. *(see page 43)*
2	Select the OPC UA configuration tool option at location **start → All programs → Schneider Electric → SoCollaborative → Ofs → OPC UA Tools → OPC UA Configuration Tool**, the following screen appears:

OPC UA Wrapper

Configuration

Select the OPC UA wrapper option at location **Start → All programs → Schneider Electric → SoCollaborative → Ofs → OPC UA Tools → OPC UA Wrapper**.

Result: The **OPC UA Wrapper Server** screen appears:
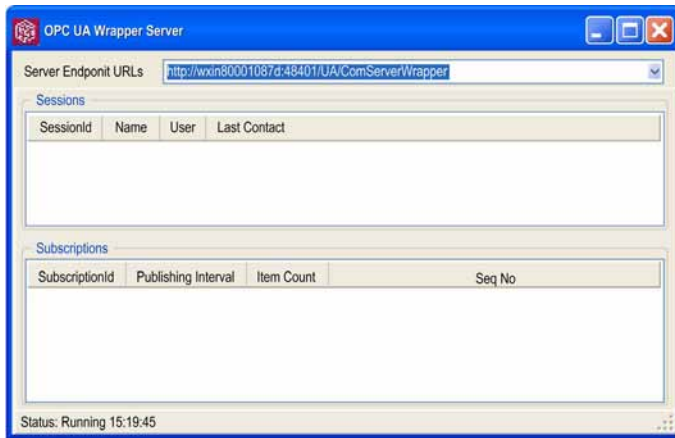
Different Access Modes for Server Connection

There are two modes of connection between OPC UA wrapper and OPC UA clients:
● local access, running on same PC
● remote access, running on different PC

Local Access

The OPC UA wrapper and OPC UA client are running on same PC. The OPC UA wrapper communicates with OFS version V3.60 server and provides data to OPC UA client.

This figure shows the OPC UA wrapper and OPC UA client on same PC:
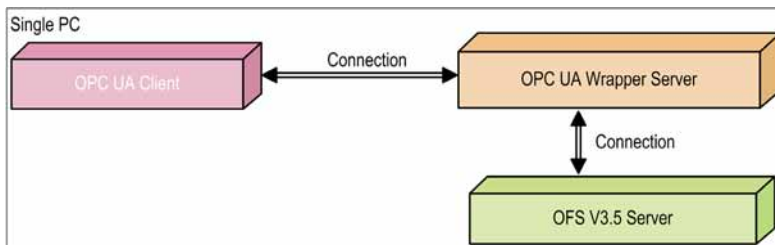
Remote Access

The OPC UA wrapper and OPC UA client are running on two different PC. But OFS version V3.60 server and OPC UA wrapper, both applications must run on a same windows PC. OPC UA wrapper communicates with OFS version V3.60 server and provides data to OPC UA client.

The figure shows the OPC UA wrapper and OPC UA client on different PC:
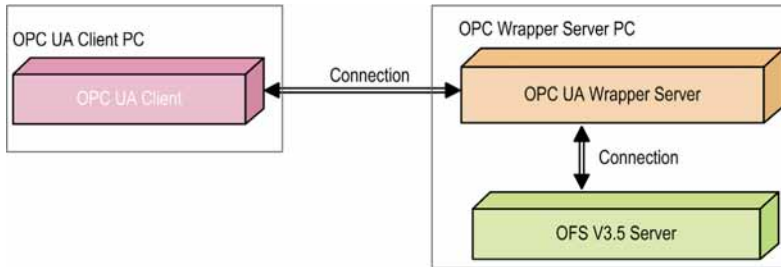
OPC UA Sample Client

General

The OFS UA sample client is an OPC client supplied with the OFS server as a test tool. It is an OPC client in compliance with the OPC UA version 1.1 standard.
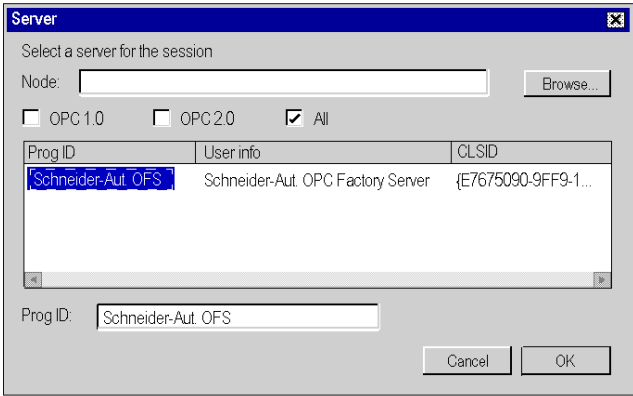
Installation
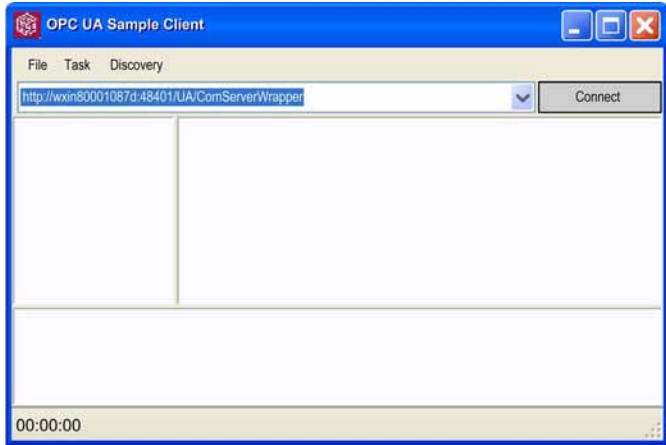
Select the OPC-UA wrapper checkbox in the options at the time of OFS installation. It installs the OPC UA sample client. *(see page 43)*

Main Uses

The OPC UA client is mainly used to verify the configuration and communication of the overall system: OPC UA client, OFS server and PLCs.

Connect as Client to OPC UA Server

Follow this procedure to install the OPC UA sample client product:

Step	Action
1	Select the OPC UA sample client option at location **start → All programs → Schneider Electric → SoCollaborative → Ofs → OPC UA Tools → OPC UA Client**, the following screen appears of the OPC UA sample client:
2	Enter the OPC UA wrapper **End Point URL**.
3	Click **Connect** to connect to the OPC UA wrapper.

Part V
User Example

Chapter 13
Example of using OFS

Aim of this Chapter

This chapter shows an example of using OFS with the OPC client supplied.

This section describes the procedure for reading and writing a word on a UNITY-type PLC.

What Is in This Chapter?

This chapter contains the following topics:

Introduction to Server Installation

Introduction

Before using the OFS server, it must be installed *(see page 37)* and configured *(see page 71)*. Once these two phases are complete, the OFS server is ready for use.

NOTE: When using ready-to run supervisory software, you may not be able to use all features listed in the following chapter (see the documentation of the OPC interface of your supervisory software to verify that point).

- Configuration:
  The OFS Configuration Tool is used to perform the following operations:
  - Configuration of symbols tables,
  - configuration of aliases and addresses,
  - configuration of the device's options with its properties page,
  - configuration of general server options.
- Operation:
  The client must launch the server and initialize communication. The user can then:
  - create groups,
  - create items,
  - execute synchronous read,
  - execute synchronous write,
  - activate group notification,
  - activate group.

At the same time, the server automatically sends notification of value changes.

Example of an OFS application with a Unity Pro PLC on TCP IP

General

In this example, you will see how to use OFS to read and write a word in a Unity Pro PLC (Quantum or Premium) using TCP IP. You need to completely install OFS (client + server).

Step 1, launching the OFS configuration tool

The following steps show how to create and configure an alias. The alias will be used by OFS to read and write a word.

- Launching the OFS configuration tool:
    - click "Start", then "Programs", "Schneider Electric", "OFS" and run "OFS Configuration Tool".

The following screen shows the main window of the OFS configuration tool:
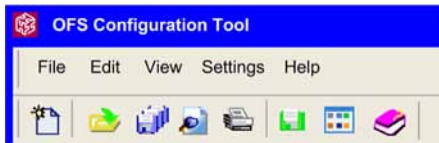
Step 2, creating an alias:

In this step, we are going to create and configure an alias:

- in the file menu, select **New Device Alias**,
    Result: A new alias is created.
- Enter the new Device Alias name.

Step 3, selecting the driver

To access the driver's configuration screen, click on the cell in the column "<driver>: <Device address>":

The following screen shows the main window for configuring the driver of the OFS configuration tool:
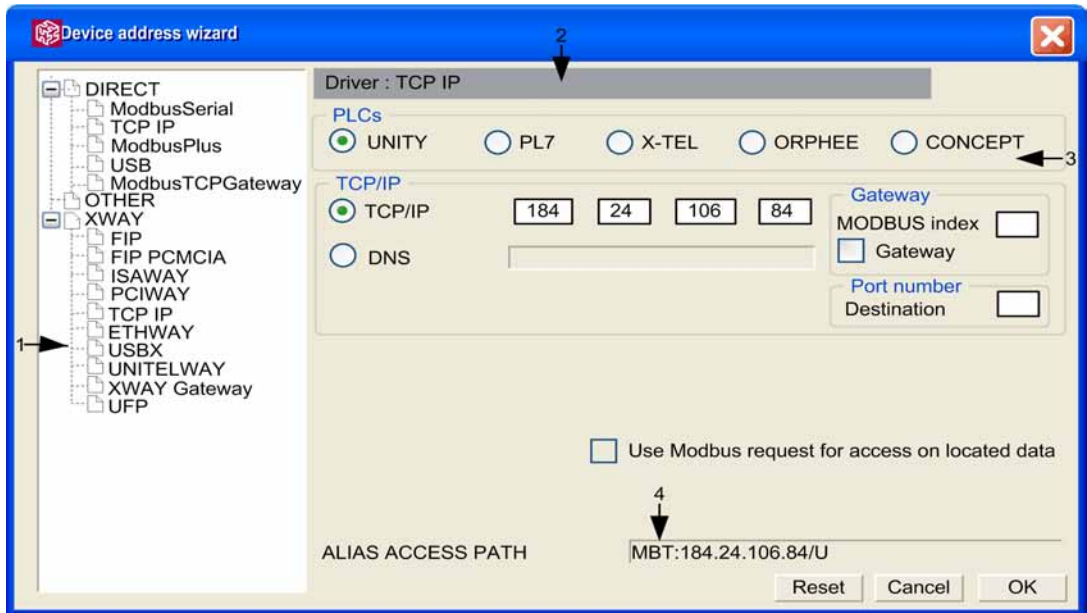
This screen will help you configure the alias. It is divided into 4 zones:
- zone 1 shows all the available drivers,
- zone 2 shows data for each driver,
- zone 3 enables you to identify the type of PLC in use,
- zone 4 displays the final data of the alias' address.

In this example, we will be communicating using the TCP IP protocol:
- in zone 1, click "+" in front of Direct, then TCP IP,
- in zone 3, define the type of PLC (in this case Unity Pro),
- validate the parameters by clicking "OK". You should see the address of the alias MBT:182.24.106.84/U.

Step 4, properties of the alias

Once the alias is created, you can set up other properties in "Alias properties" on the "OFS configuration tool".

The following screen shows the main window of the alias properties:

Device name	DevExample_1	
Device address	XIP01:116.10/U	...
⊟ **General**		
Symbol table file		...
PLC Embedded Data	☐ Using Data Dictionary ☐ No Communication Break	
Preload settings	⦿ No Preload ◯ Symbol table ◯ Device	
Option	☐ Simulated ☐ Read Only	
Comment		
⊟ **Communication Information**		
⊞ Driver information		
⊞ Push Data		
⊞ Adjustment information		
⊞ Time - Stamped Events		

Once you have entered all the parameters, select **Save configuration** in the file menu.

The alias creation phase is now complete. You can quit the OFS configuration tool selecting **Exit** (file menu).

Executing OFS and using the OPC client

General

The following steps show how to start up the OFS server with an OPC client using the aliases previously created.

NOTE: When you make a change in the OFS configuration tool, you must re-start the server for the modifications to be taken into account.

Step 1: Starting the server and using the OPC client

The OFS server is launched automatically when an OPC client is started.

OPC clients are started from the menu "Start", then "Programs", "Schneider Electric", "OFS", "OFS Test Clients" and "OPC Client"

The following window appears:
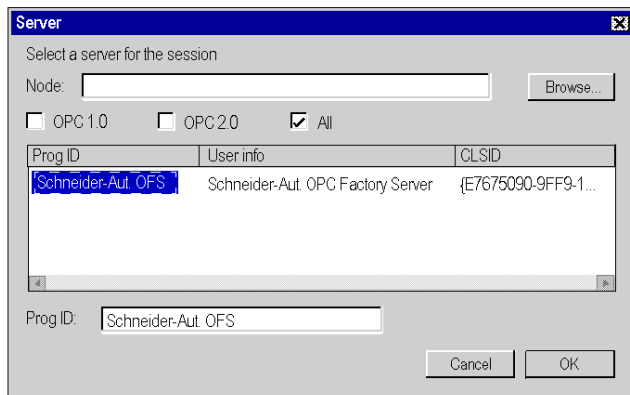
- Select the Schneider-Aut. OFS server in **Prog ID**,
- Enter the name of the computer of the OFS server in **Node**,
- and click on **OK** to connect to the OFS server.

Step 2: Creating an Active Group

In the OFSClient window, click on File then New. The option "Initially active" is checked by default. Give this group a name then click OK.

Step 3: Adding an item

Click "Item" then "New" in the toolbar. The following window appears.

This screen shows all the aliases created in the OFS configuration tool. Click on your alias "Device_TCPIP" which appears in the field Item ID.

In our example we want to read and write a word. Let us take for example the word 10, or %MW10.

The OFS Client syntax to enter in order to read or write a word is !%MW10 after the name of the alias "Device_TCPIP" (see screen above).

Step 4: Reading/writing an item

This step describes the procedure for reading or writing a value on a word.

The figure below shows the result of adding an item:
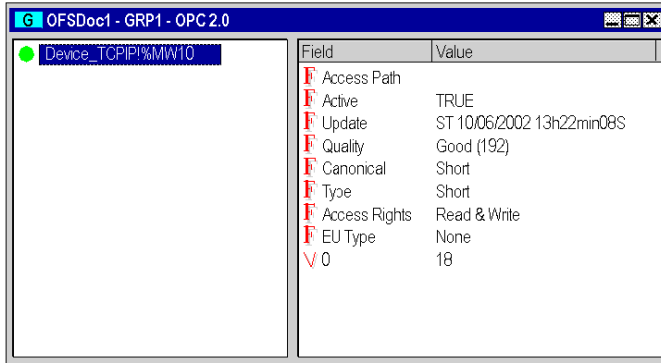
The field "Active" means the item is refreshed periodically by a value change in the PLC.

The item is active when the indicator (next to the item) is green.

The field "Good(192)" means the value displayed is the current value in the PLC.

To write a value in this item, click "Item" then "Write". Enter a number in the "Value" field and click OK.

Part VI
Advanced User Guide

Aim of this section

The aim of this section is to guide you in the product's advanced features.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
14	Concepts	181
15	Items	187
16	Variables	229
17	Symbols	259
18	The Diag Buffer	289
19	Communication	339
20	Performance	361
21	Client-alive Service	381

Chapter 14
Concepts

Aim of this Chapter

The aim of this chapter is to describe certain important features of the product.

What Is in This Chapter?

This chapter contains the following topics:

Synchronous Services

Description

- these services are used for **partial** or **complete** reading and writing of a group of items,
- the periodic scanning of variables (read polling) must be handled by the client application,
- the term "**synchronous**" means that the client application which calls up these read or write services is blocked for the time it takes to obtain a result. The instruction which follows a synchronous read or write call in the code of the client application will only be executed when all the communication requests corresponding to that call have been processed. During a synchronous read operation, the OFS server **does not guarantee** that all the variables in a group **will be accessed in the same PLC scan** if this group is **transcribed on several** communication requests. The OFS server provides a mechanism for ascertaining the number of requests necessary to **access the whole** of a group of items (for synchronous groups only).

A synchronous read or write service is consistent if it is either:

- Done in single request user group (with prefix '$')
- The number of communication requests does not exceed 1 (check the value of the #NbrRequest specific item)

NOTE: For more details refer to the section on consistencyReading *(see page 262)* and writing *(see page 263)*

⚠ **WARNING**
UNINTENDED EQUIPMENT OPERATION
To ensure that a synchronous read or write is performed in the same PLC scan, make sure that the client application read or write operation is consistent.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Asynchronous Services

Description

- These services are used for partial or complete reading and writing of a group of items,
- the periodic scanning of the evolution of variables (read polling) must be handled by the client application,
- the client application is not blocked during the time it takes to obtain the data,
- a notification mechanism (which must be activated) notifies the client of the results,
- the process for synchronization with the PLC is exactly the same as the one outlined for synchronous *(see page 182)* services.

Notification service

Description

The periodic scanning of variables: read polling and the notification of changes in variable values are performed by the OFS server.

The client application should program a "Wake up" function, which is called up by the OFS when value changes have occurred on items of all periodically examined groups.

This means that the "Wake Up" function is unique in the client application: it receives all the notifications from the OFS server, then it must redistribute them to the processing functions specific to each periodically scanned group.

NOTE: For ready-to-run supervisory software, the "Wake Up" function should be pre-programmed. If this is not the case, the notification mechanism may not be used.

The name of this "Wake Up" function is set by the OPC standard **OnDataChange**.

NOTE: In the "Wake up" function, processes that take up a significant amount of CPU time (e.g.: over complex display) should not be performed, as this may adversely affect the Operating System's performance.

The OFS server notifies by group, not individually by item. That means that, for a given group, the OFS server sends the list of items whose value has changed to the client application "Wake Up" function. In the case of a table type item, the OFS server transmits the whole table even if only a subset of the elements has changed in value.

The following notions are associated with the notification service:

● Assignment of a scanning period ("RATE") to a group: this enables you to scan the PLC variables with different periods.
Example: display the PLC time every second, and display a temperature every minute.

● Allocation of a deadband to a group (dead banding): filtering of notifications when group variable values change. Notification occurs if, after the group scanning period, variables have changed by more than a certain percentage with respect to their old value (see chapter on Deadband *(see page 111)*).
Example: inform the client application only if temperatures have changed by more than 10%.

NOTE: Deadbanding is only applied to floating point or integer variables. The aim of these two concepts is to enable the user to control (limit) the flow of notifications sent to the client application, to avoid overloading the system.

Symbol consultation

Description

The OPC Browse Interface is supported by the OFS product. This enables users to browse symbols available for a given PLC, provided that the OPC Client used supports the Browse interface. This is an easy way to determine which variables can be created for a given device. Browsing structures and tables are available when the programming language includes these object types (e.g.: Concept programming tool).

NOTE: Only devices declared with the Configuration Tool and associated with a Symbol Table can be browsed.

NOTE: When browsing Unity Pro symbols of ANY_ARRAY type, only the first element of the table is visible.

Chapter 15
Items

Aim of this Chapter

The aim of this chapter is to introduce operations on OFS variables.

What Is in This Chapter?

This chapter contains the following sections:

Section 15.1
Items under OFS

Aim of this Section

The aim of this section is to describe OPC items.

What Is in This Section?

This section contains the following topics:

General information on OPC items

General

Before reading or writing values, an OPC item should be created for each device variable.

The general syntax for an OPC item is:

<item>::=**<driver name>**:**<address device>**/<Device Type>!**<definition variable>**[:**<length table>**|<number of extracted bit>][;**<postfix>**]

<nom driver>:**<device address>** /<Device Type>The part may be replaced by an alias *(see page 88)* created by the Configuration Tool.

If an alias is not used, the **driver name** must be one of the names given in the following list and the **device address** is the address of the device on the communication medium:

Driver name	Example of device address	Communication medium
UNTLW01 *	0.254.0	Uni-Telway
FIP01 *, FIP02 *	1.31	Fipway adapter 01 or 02
FPP2001 *	1.31	Fipway PCMCIA adapter 01
ISAway01, ISAway02	0.254.0	ISAway adapter 01 or 02
Ethway01 *, Ethway02 *	1.31	Ethway adapter 01 or 02
XIP01 -> XIP09	1.31	X-Way TCP-IP adapter 01 to 09
MBPLUS01,MBPLUS02,MBPLUS03, MBPLUS04	PM.12 or DM.15.3	Modbus Plus adapter 0 or 1 or 2 or 3
MBT	139.160.218.102	Modbus TCP-IP
MBTG	139.160.218.102	Modbus TCP (gateway)
MODBUS01,MODBUS02,MODBUS03, MODBUS04	6	Modbus Serial adapter 1 or 2 or 3 or 4
Pciway01, Pciway02	0.254.0	PCIway adapter 01 or 02
USB	-	USB
USBX	0.254.0	USB with X-Way address
UFP01, UFP02	1.31	Fipway USB adapter 01 or 02

The variable definition part can either be a variable address (see the "Syntax" column in the other tables in this chapter) or a symbol *(see page 87)*.

Modbus Plus users planning to use Concept and OFS simultaneously, or the multi-channel feature, should use DM mode. Otherwise you may not be able either to connect to the PLC with Concept or download your application.

For variables that support this feature, the table length enables the user to create items that are actually tables and gives the number of elements making up this table.

The postfix can be R (read only) and is a way of creating an item that will always be considered as read only.

NOTE: The **driver name**, **device address**, device type and **variable definition** parameters are mandatory.
 The **table length** and **suffix** parameters are optional.
Examples:
- UNTLW01:0.254.0/S!%MW3
- MODBUS01:12/Q!400003
- FIP01:0.31.0/U!%MW5
- MBPLUS01:DM.5/Q!400005
- XIP01:0.5/T!%MW100
- MBT:1.2.3.4/U!%MW100
- TSX1!%MW100
- QTM1!400100
- TSX2!toto
- QTM2!toto

The **device address** field for MBT and MBTG, uses the ";xx" suffix to designate the index of the Modbus node connected to the TCP/IP gateway. For example, "139.160.218.103;50".

NOTE: To define an item, the OPC DA test clients accept the two following syntaxes:
- The Path field provides the Alias and the Name field defines the variable,
- The Path field is empty (or indicates the host server) and the Name field completely defines the item according to this syntax: Alias!Variable.

Contrary to OPC DA clients, OPC XML clients may use the Path field only to qualify the host server. If the field is empty, the default server is then addressed.

Definition of a group of items

Definition

All the OFS product's services are based on the concept of a group of items:

- Several groups can be defined.
- A group may concern several devices: each item of a group can have a different device address.
- A group concerns various communication devices and media: each item may refer to a different communication driver. If a device can be accessed via several communication media, it is possible to mix variables addressed via different media within one group.
- The items comprising a group can be different: it is possible to mix all types of objects managed by the OFS server.
  For example: mixing words, double words and floating points within one group.
- All the items in the same group have the same update rate and deadbanding percentage.

An item is a variable of any PLC which can be accessed either by their address or by their symbol.

OPC Item Properties

Properties

The IOPCItemProperties interface is supported by OFS server.

The following properties are supported:

- canonical data type,
- value,
- quality,
- timestamp,
- access rights,
- description (only if a comment has been given in the Workshop),
- the forcing state of a bit (only for input bits and output bits, see *I/O module objects, page 240*).

For **Concept variables** and **Unity Pro variables** only:

- InitialValue (the initial value of a variable),
- VariableKind (the type of variable: elementary, structured, function block, section),
- VariableTypeId (The Type id as known by the Concept tool),
- MemoryArea (areas: 0x,1x,3x,4x, unlocated, unused, etc.),
- AreaIndex (the index inside of the memory area),
- VariableSize (the size, useful for non elementary variables),
- RelativeOffset (the offset inside of a structured variable).

For a given variable, some of them may not be supported if it doesn't make sense (e.g.: no Description if the variable does not have a comment, no InitialValue if the variable has no initial value, etc.)

For **Unity Pro variables** only:

- CustomString (free character string buffer),

To test the use of OPC item properties, you can use OFS client (see *OFS C++ OPC DA Client, page 148*).

Example of use:

You want to know when the link between the PC and the PLC is broken. When this is the case, you want to display something special in your OPC client application:

The quality of an item is the specification to use: in general, it is not possible to use the item quality to display something, only its value can be used.

The solution is to create an item, which has a value directly linked to the quality of another item.

When the communication works well, the quality value is always 192 (QUALITY_Good). On the contrary, the quality value is 24 or 28 (QUALITY_Bad + reason).

Using OFS client, create a group and an item. When you have done this, reopen the browse interface, reselect the same symbol and click on the Properties button. Select ID 3 (Item Quality) then double-click on OK. The value of the new item is the quality of the previous item.

Specific Items

Description

A specific item is an OPC item which is not related to any PLC variable but is a way to view/modify some internal parameters (internal to the OPC server or internal to the PLC). These items can be used with the test client given with the product, thus avoiding any modifications to your OPC application that may not be reusable with any another OPC server *(see page 148)*.

- a specific item has a path like any other item,
- the definition of a specific item always starts with a '#' character,
- specific item can be created in any group (except synchronous groups),
- certain specific items can have an active status in an active group. Thus, the server can automatically detect changes,
- specific items can be read/written within any subset of the group (including both ordinary and specific items),
- to read or write a specific item either synchronous or asynchronous functions can be used.

Specific items available for a device can be browsed in the **#Specific** subfolder attached to any device. The **Diag Buffer** function has been implemented in the form of a set of specific items. They are introduced in a separate section, in addition to the list provided below.

NOTE: All specific items are disabled if the enable OPC extensions box is not checked in the Communication folder of the configuration tool *(see page 123)*.

Specific items can be divided in 2 categories, basic specific items and enhanced diagnostics-specific items

Basic Specific Items

This table shows the basic specific items:

Name	Type	Access	Can be activated
#AppliName	VT_BSTR	R	No
#AppliVersion	VT_BSTR	R	No
#DisableDevice	VT_I2	R/W	Yes
#MaxChannel	VT_I2	R/W	Yes
#NbrMaxPendingReq	VT_I2	R/W	Yes
#NbrRequest	VT_I2	R	No
#PlcStatus	VT_I2	R/W or R	Yes
#RefreshDevice	VT_I2	R/W	No
#TimeOut	VT_I2	R/W	Yes
#DeviceIdentity	VT_BSTR	R	Yes
#AppliID	VT_I4/VT_ARRAY	R	Yes
#AppliOMC	VT_I4/VT_ARRAY	R no polling	No

Name	Type	Access	Can be activated
#PLCQualStatus	VT_I2	R	Yes
#PLCQualStatus2	VT_I2	R	Yes
#SwitchPrimaryAddress	VT_UI2	R/W	Yes
<<system>>!#ClientAlive	VT_BOOL	R/W	No
#TSEventSynchro	VT_BOOL	R/W	No
#TSEventItemsReady	VT_BOOL	R	Yes

#AppliName

Name	Type	Access	Can be activated
#AppliName	VT_BSTR	R	No

NOTE: The value of `#AppliName` is read only at first variable creation.

It gives you the name (if any) read from the device.

#AppliVersion

Name	Type	Access	Can be activated
#AppliVersion	VT_BSTR	R	No

NOTE: The value of `#AppliVersion` is read only at first variable creation.

It gives you the application version (if any) read from the device. This item is updated following start-up of the server (the initial item provided by item `#AppliVersion` is kept even if the project is reloaded after an application modification).

#DisableDevice

Name	Type	Access	Can be activated
#DisableDevice	VT_I2	R/W	Yes

If communication with the device is enabled, the value read is 0, otherwise the value read is 1.

To modify the state, write either 0 or 1.

This item can be used to temporarily disable communication with a device (for instance before modifying the device that cuts communication) to avoid timeouts or others.

If the value written is 1, the items related to that device become `Bad` immediately since the server stops sending requests to that device. If the value written is 0, the server will once again send all the requests to the device, and the items should become `Good` again within a few seconds.

#MaxChannel

Name	Type	Access	Can be activated
#MaxChannel	VT_I2	R/W	Yes

This item is related to the multi-channel *(see page 348)* feature.

Even if it is possible to create it for any device, it is not practical to do so for all types of device.

You can ascertain the maximum number of channels currently configured for this device by carrying out a read. Its value may be the result of the off-line configuration *(see page 88)*.

You can set the maximum number of channels that can be used to communicate with the device by carrying out a write. You can either decrease or increase the value and it is taken into account rapidly allowing the results of the tuning of the number of channels to be viewed immediately.

This item is an adjustment parameter that allows you to find the best efficiency in order to achieve a permanent configuration using the configuration tool. Modifications (above all for reducing the value) may cause temporary disruption to communication.

#NbrMaxPendingReq

Name	Type	Access	Can be activated
#NbrMaxPendingReq	VT_I2	R/W	Yes

Read/Write of the `#NbrMaxPendingReq` parameter for a given device. This parameter is the number of requests that can be transmitted in parallel to the device before its capacity is exceeded. In general, this parameter is automatically adjusted by OFS to the value which gives the best server performance. However, it can be lowered to avoid communication overloads to the PLC.

This parameter is strongly linked to the previous parameter (`#MaxChannel`) for devices managing multi-channel *(see page 348)* configurations.

#NbrRequest

Name	Type	Access	Can be activated
#NbrRequest	VT_I2	R	No

Its value (a number of requests) is only related to one device (defined by its path). This indicates the number of requests sent to that device, by the server, to refresh its internal cache. This includes all the frequencies that may exist in the server.

This item *(see page 261)* may be created with no path within a synchronous group (name starting with $ or $$). In this case, its value is the number of requests necessary to read the whole group.

If this item is created with no path within an ordinary group, its value is always 0. This is only possible for questions of compatibility.

#PlcStatus

Name	Type	Access	Can be activated
#PlcStatus	VT_I2	R/W or R	Yes

The value returned is the PLC mode (1 if the PLC is running, 0 if it is stopped).

It is possible to write the value to force the operating mode of the PLC. To do this, the Change PLC Status *(see page 223)* option must have been checked in the **Communication** folder of the configuration tool.

NOTE: Using the #PlcStatus is costly. You are advised to insert this item within a group with a large period.

#RefreshDevice

Name	Type	Access	Can be activated
#RefreshDevice	VT_I2	R/W	No

This item is designed to manage the consistency between the symbol table file and the application in the PLC.

If the value 1 is written to this item, the server reads the application name and version from the device.

If this item is read, the server performs a consistency check between the application name and version already read from the device and the same information from the symbol table file open for that device. The value returned can be:

- 0: no check was performed (no symbol table information or device information),
- 1: everything is OK and consistent,
- 2: application names are not consistent,
- 3: application versions are not consistent,
- 6: applications are different but the symbols are consistent.

#TimeOut

Name	Type	Access
#TimeOut	VT_I2	R/W

Its value (expressed in ms) is only related to one device (defined by its path). This value represents the frame timeout which is the time the server will wait for an answer from the device after it has sent a request. Any write will modify the server internal parameter for this device.

#DeviceIdentity

Name	Type	Access	Can be activated	Value read
#DeviceIdentity	VT_BSTR	R	No	PLC product reference

#AppliID

Name	Type	Access	Can be activated	Value read
#AppliID	VT_I4/VT_ARRAY	R	Yes	8 ID (values) maximum can be read 5 ID can be read to date: ● 1: (CID) Creation ● 2: (MID) Overall modification ● 3: (AID) Automatic modification ● 4: (LID) Memory availability ● 5: BLOCKID (DID) Data blocks

#AppliOMC

Name	Type	Access	Can be activated	Value read
#AppliOMC	VT_I4/VT_ARRAY	R no polling	No	16 OMC (counters) maximum can be read 8 OMC can be read to date: ● 1: Overall application ● 2: Program ● 3: Configuration ● 4: Type definition ● 5: Variables ● 6: Animation table ● 7: Communication ● 8: Functional modules

The value of these items is a copy of the Application signature (ID) and the object modification counters (OMC) read in the PLC. This can be used to feed back internal changes made by Unity Pro on generation of the application. Some of these signatures are modified each time you generate an application, others are positioned only when the application is created.

These items are used to detect an application modification, or to check their consistency, for example in a complex use of **Diag Buffer**.

Each signature is written in a double word (DWORD).

Generally, to detect an application modification, #AppliID is enabled, then #AppliOMC is read to precisely determine the modification type.

#AppliOMC cannot be enabled, only read.

#PLCQualStatus

Name	Type	Access	Can be activated	Value read
#PLCQualStatus	VT_I2	R	Yes	● QUAL_BAD and QUAL_COMM_FAILURE: Device is INCONSISTENT (SymbolFile differs from PLC application). ● QUAL_BAD and QUAL_DEVICE_FAILURE: No communication with the device since DEVICE_TO milliseconds. ● QUAL_BAD: Device is MISSING or UNKNOWN. ● QUAL_GOOD: Communication with the device is correct. ● QUAL_UNCERTAIN: No communication with the device since less than DEVICE_TO milliseconds.

This item has to be added as active in a standard active OPC group in the form `<myAlias>!#PLCQualStatus`, where `myAlias` is the PLC alias configured in the OFS Configuration tool.

This item allows monitoring the communication activity of a device.

As event sources other than PLCs are not configured in the OFS configuration tool, reference `#PLCQualStatus` in the following form: `<direct_address>!#PLCQualStatus`:
● For BMX-CRA drop: `direct_address>=MBT:<IP_address>/U` (with `IP_address` is the drop IP address).
● For InRack BMX-ERT: `<direct_address>=<direct_address_plc>\\<r>.<s>.<c>/U`
  ○ r: rack number of the BMX-ERT
  ○ s: slot number of the BMX-ERT in the rack
  ○ c: channel number of BMX-ERT in the slot
  ○ direct_address_plc: PLC address build with the configuration tool
● For BMX-ERT located in a BMX-CRA drop: `<direct_address>= MBT:<IP_address>\\<r>.<s>.<c>/U`

NOTE: When `#PLCQualStatus` is active, a `%S0` active item is implicitly added to the group. That guaranties that a device connection is established for diagnostics (a read request is periodically sent to the device).

NOTE: Adding many `#PLCQualStatus` linked to a same device in several groups with different rate generate a read request at the smaller update rate.

NOTE: For event sources other than PLC (BMX-ERT, BMX-CRA, and so on) `%S0` does not exist. In that case, OFS switches in a transparent way to the mechanism described below (`#PLCQualStatus2` item). From OPC client side, the syntax remains `<direct_address>!#PLCQualStatus`.

#PLCQualStatus2

Name	Type	Access	Can be activated	Value read
#PLCQualStatus2	VT_I2	R	Yes	● QUAL_BAD and QUAL_COMM_FAILURE: Device is INCONSISTENT (SymbolFile differs from PLC application). ● QUAL_BAD and QUAL_DEVICE_FAILURE : No communication with the device since DEVICE_TO milliseconds. ● QUAL_BAD: Device is MISSING or UNKNOWN. ● QUAL_GOOD: Communication with the device is correct. ● QUAL_UNCERTAIN: No communication with the device since less than DEVICE_TO milliseconds.

This item has the same features as `#PLCQualStatus`. The difference is that the device connection is maintained by sending, periodically, a mirror request to the device, and no a read request.

NOTE: Adding many `PLCQualStatus2` linked to a same device in several groups with different rate generates a mirror request for each rate. Thus, it is recommended to activate only one `PLCQualStatus2` by device at the accurate rate.

#SwitchPrimaryAddress

Name	Type	Access	Can be activated	Value read
#SwitchPrimaryAddress	VT_UI2	R/Y	Yes	• 0: indicates that primary communication path is `Device Address A`. • 1: indicates that primary communication path is `Device Address B`. • QUAL_BAD: Communication with the device is interrupted, the primary, and standby communication paths are **OFFLINE**. • QUAL_GOOD: Communication with the device is correct and the primary communication path is **ONLINE**

Writing any value will operate a switch from primary to standby communication path.

`E_FAIL` is returned on a write operation if no standby communication path is configured or if standby communication path is **OFFLINE**

<<system>>!#ClientAlive

Name	Type	Access	Can be activated	Value read
<<system>>!# ClientAlive	VT_BOOL	R/W	No	Refer to Client-alive Service .

#TSEventSynchro

Name	Type	Access	Can be activated	Value read
#TSEventSynchro	VT_BOOL	R/W	No	–

This item allows OFS to send a `Synchro value` to the entire event sources attached to the PLC, each time a write operation is performed.

It can be performed by the SOE viewer when it starts or when it restarts. It allows getting initial values or current values of all the configured events items. Perform it before the event group activation. It is the responsibility of the OPC client to manage the correct order sequence.

NOTE: For VJC I/O server is generic and cannot do anything dedicated to OFS. The job described above is done by OFS OPC driver.

When `#TSEventSynchro` is added to a standard OPC group (that is not in a reserved event group), OFS connects to all the TS events sources linked to the current device alias. On each write, OFS sends a `Synchro value` to all the event sources linked to the current device alias.

#TSEventItemsReady

Name	Type	Access	Can be activated
#TSEventItemsReady	VT_BOOL	R	Yes

This specific item value is set to `FALSE` each time OFS server starts to browse TS Events and set to `TRUE` when the TS Events browse is completed.

Specific Items Supported on PLCs

The table shows the specific items available on the different PLCs:

	Unity PLCs	PL7 PLCs on X-Way networks	PL7 and ORPHEE PLCs on non-X-Way networks	CONCEPT PLCs	XTEL PLCs	ORPHEE PLCs
#AppliName	R	R	Not available	R	Not available	Not available
#AppliVersion	R	R	Not available	R	Not available	Not available
#PlcStatus	R/W	R/W	Not available	R/W	R/W	Not available
#DisableDevice	R/W	R/W	R/W	R/W	R/W	R/W
#TimeOut	R/W	R/W	R/W	R/W	R/W	R/W
#NbrMaxPendingReq	R/W	R/W	R/W	R/W	R/W	R/W
#RefreshDevice	R/W	R/W	Not available	R/W	Not significant	Not significant
#NbrRequest	R	R	R	R	R	R
#MaxChannel	R/W	Not available	R/W	R/W	Not available	Not available
#DeviceIdentity	R	Not available	Not available	Not available	Not available	Not available
#AppliID	R	Not available	Not available	Not available	Not available	Not available
#AppliOMC	R	Not available	Not available	Not available	Not available	Not available
#OFSStatus	R	R	R	R	R	R

	Unity PLCs	PL7 PLCs on X-Way networks	PL7 and ORPHEE PLCs on non-X-Way networks	CONCEPT PLCs	XTEL PLCs	ORPHEE PLCs
#PLCQualStatus	R	R	R	R	R	R
#PLCQualStatus2	R	R	R	R	R	R
#SwitchPrimaryAddress	R/W	Not available				
system → !#ClientAlive	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable
#TSEventSynchro	R/W	Not available	Not available	Not available	Not applicable	Not applicable
#TSEventItemsReady	R	Not available	Not available	Not available	Not available	Not available

Enhanced Diagnostics Specific Items

This table shows the enhanced diagnostics specific items supported by Unity PLCs only:

Name	Browse Path	Type	Implicit Access	Can be activated
#IOStatus	#Specific\Diag\IO	VT_BOOL	%S10	Yes
#WatchDog	#Specific\Diag\CPU	VT_BOOL	%S11	Yes
#PlcRunning	#Specific\Diag\CPU	VT_BOOL	%S12	Yes
#OverRun	#Specific\Diag\CPU	VT_BOOL	%S19	Yes
#RackIOStatus	#Specific\Diag\IO	VT_BOOL | VT_ARRAY (size=8)	%S40…47	Yes
#PCMCIABattStatus0	#Specific\Diag\CPU	VT_BOOL	%S67	Yes
#PlcBattStatus	#Specific\Diag\CPU	VT_BOOL	%S68	Yes
#PCMCIABattStatus1	#Specific\Diag\CPU	VT_BOOL	%S75	Yes
#DiagBuffConf	#Specific\Diag\Application	VT_BOOL	%S76	Yes
#DiagBuffFull	#Specific\Diag\Application	VT_BOOL	%S77	Yes
#BackupProgOk	#Specific\Diag\CPU	VT_BOOL	%S96	No
#IntIOStatus	#Specific\Diag\IO	VT_BOOL	%S117	Yes
#ERIOStatus	#Specific\Diag\IO	VT_BOOL	%S117	Yes
#FIPIOStatus	#Specific\Diag\IO	VT_BOOL	%S118	Yes
#REMIOStatus	#Specific\Diag\IO	VT_BOOL	%S118	Yes
#LocIOStatus	#Specific\Diag\IO	VT_BOOL	%S119	Yes
#MastPeriod	#Specific\Diag\Application	VT_I2	%SW0	No

Name	Browse Path	Type	Implicit Access	Can be activated
#FastPeriod	#Specific\Diag\Application	VT_I2	%SW1	No
#AuxPeriod	#Specific\Diag\Application	VT_I2|VT_ARRAY (size=4)	%SW2...5	No
#WatchDogValue	#Specific\Diag\Application	VT_I2	%SW11	No
#OSVersion	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW14...16	No
#MastReqNb	#Specific\Diag\CPU	VT_I2	%SW26	No
#MastTimes	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW30...32	No
#FastTimes	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW33...35	No
#Aux0Times	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW36...38	No
#Aux1Times	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW39...41	No
#Aux2Times	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW42...44	No
#Aux3Times	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW45...47	No
#CPUStopTime	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=5)	%SW54...58	No
#HSBYStatus	#Specific\Diag\CPU	VT_I2	%SW61	Yes
#CCOTFStatus	#Specific\Diag\CPU	VT_I2	%SW66	Yes
#KeySwitch	#Specific\Diag\CPU	VT_I2	%SW71	No
#ReqCounters	#Specific\Diag\Comm	VT_I2|VT_ARRAY (size=3)	%SW87...89	No
#MaxReqNb	#Specific\Diag\Comm	VT_I2	%SW90	No
#AppSign	#Specific\Diag\Application	VT_I2|VT_ARRAY (size=2)	%SW94 %SW95	Yes
#CardStatus	#Specific\Diag\CPU	VT_I2	%SW97	No
#ForcedObjects	#Specific\Diag\IO	VT_I2|VT_ARRAY (size=2)	%SW108 %SW109	Yes
#CPUErr	#Specific\Diag\CPU	VT_I2	%SW124	No
#CPUErrType	#Specific\Diag\CPU	VT_I2	%SW125	No
#FIPIOCnxPointStatus	#Specific\Diag\IO	VT_I2|VT_ARRAY (size=16)	%SW128...143	Yes
#OpenConnectionsNb	#Specific\Diag\Comm	VT_I2	%SW128	No

Name	Browse Path	Type	Implicit Access	Can be activated
#GlobalDataStatusValue	#Specific\Diag\IO	VT_I2	%SW138	Yes
#EthWorkload	#Specific\Diag\Comm	VT_I2|VT_ARRAY (size=2)	%SW139 %SW140	No
#IPAddress	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=2)	%SW141 %SW142	No
#IPSubnetMask	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=2)	%SW143 %SW144	No
#IPGateway	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=2)	%SW145 %SW146	No
#MacAddress	#Specific\Diag\CPU	VT_I2|VT_ARRAY (size=3)	%SW147...149	No
#ERIODropStatus	#Specific\Diag\IO	VT_I2|VT_ARRAY (size=4)	%SW152...155	Yes
#RackStatus	#Specific\Diag\IO	VT_I2|VT_ARRAY (size=8)	%SW160...167	Yes
#GlobalDataStatus	#Specific\Diag\IO	VT_I2|VT_ARRAY (size=4)	%SW168...171	Yes
#ERIOCnxStatus	#Specific\Diag\IO	VT_I2|VT_ARRAY (size=4)	%SW172...175	Yes

Simple Diagnostics Specific Items

This table shows simple elements of the specific items supported by Unity PLCs only:

Item	Description	Implicit Access
#IOStatus	Supported PLC: Any (Any device type). Normally at 1, this bit is set to 0 when: ● A fault occurs on an input/output module of a basic or extension rack configuration (incorrect configuration, exchange fault, terminal block or module missing, module fault, trip, and so on). The input/output fault diagnostic bits provide information on the nature of the fault. ● A fault occurs on an extension rack (loss or power supply fault, link or extension module fault). This bit is set to 0 when: ● On Premium, whenever a fault occurs on one of the devices connected on FIPIO (incorrect configuration of one of the modules, exchange fault, fault on one of the modules comprising the device, power loss or power supply fault on the device, and incorrect connection point). ● On Premium, whenever a fault occurs on the FIPIO link (link fault, processor connection coding incorrect, channel master conflict on FIPIO). On Quantum, whenever an I/O fault occurs on Local, Distributed I/O CPU, Distributed I/O NOM, Remote I/O or Ethernet Remote I/O bus. On M580, whenever an I/O error occurs on Local or Ethernet Remote I/O bus. The bit is set to 0 by the system on discovering I/O faults and reset to 1 as soon as the fault disappears.	%S10
#WatchDog	Supported PLC: Any. This bit is set to 1 when it detects overrun of the software watchdog of at least one of the tasks. The application goes to Halt state (PLC in Stop). The software watchdog values are defined in the task configuration. This bit is reset to 0 when Init command is sent by the user after halt.	%S11
#PlcRunning	Supported PLC: Any. This bit reflects the running state of the application: ● 0 = Stop, init, or any other state. ● 1 = Application is running.	%S12

Item	Description	Implicit Access
#OverRun	Supported PLC: Any. Normally at 0, this bit is set to 1 when the execution of a periodic task exceeds the period value. This bit is reset to 0 by the user program. This bit is contextual to each task.	%S19
#PCMCIABattStatus0	Supported PLC: Premium/Quantum. This bit is used for preventive maintenance of battery of SRAM PCMCIA card on slot 0. It is set to 1 when the level of battery is below threshold (BVD2 signal). In this case, data retention is still insured, but the battery must be changed. This bit is supported on all SRAM cards irrespective of their size.	%S67
#PlcBattStatus	Supported PLC: Premium/Quantum. You can diagnose the backup battery by using the following results: ● %S68 = 0: Battery is present and functioning. ● %S68 = 1: Battery is missing or no longer functioning.	%S68
#PCMCIABattStatus1	Supported PLC: Premium/Quantum. This bit is used for preventive maintenance of battery of SRAM PCMCIA card on slot 1. It is set to 1 when the level of battery is below threshold (BVD2 signal). In this case, data retention is still insured, but the battery must be changed. This bit is supported on all SRAM cards irrespective of their size.	%S75
#DiagBuffConf	Supported PLC: Any. This bit indicates if the diagnostic buffer is configured: ● 1: Diagnostic buffer is configured. ● 0: Diagnostic buffer is not configured.	%S76
#DiagBuffFull	Supported PLC: Any. This bit indicates if the diagnostic buffer is full: ● 1: Diagnostic buffer is full. ● 0: Diagnostic buffer is not full.	%S77

Item	Description	Implicit Access
#BackupProgOk	Supported PLC: Modicon M340/Modicon M580. This bit is set to 0 by the system when the card is missing or not usable (bad format, unrecognized type, and so on) or card content inconsistent with internal application RAM. This bit is set to 1 when the card is correct and application is consistent with CPU internal application RAM.	%S96
#IntIOStatus	Supported PLC: Modicon M580. Normally at 1, this bit is set to 0 after detection of fault in a device on the CPU integrated network.	%S117
#ERIOStatus	Supported PLC: Quantum. Normally at 1, this bit is set to 0 after detection of fault in a device on the Ethernet Remote I/O network.	%S117
#FIPIOStatus	Supported PLC: Premium. Normally at 1, this bit is set to 0 after detection of fault in a device on FIPIO; the causes are listed in the definition of system bit %S10.	%S118
#REMIOStatus	Supported PLC: Modicon M340/Unity Momentum/Quantum . On Quantum: normally at 1, this bit is set to 0 whenever an I/O fault occurs on S908 Remote I/O bus. On Modicon M340: normally at 1, this bit is set to 0 whenever an I/O fault occurs on CANOPEN bus. On Unity Momentum: normally at 1, this bit is set to 0 whenever an I/O fault occurs on I/O bus	%S118
#LocIOStatus	Supported PLC: Any. Normally at 1, this bit is set to 0 whenever an I/O fault occurs on a I/O module placed in one of the racks; the causes are listed in the definition of system bit %S10.	%S119
#MastPeriod	Supported PLC: Any. Contains the period of this task. If this value is 0 (%SW0 = 0), the functioning of the MAST task is cyclical. The period is expressed in ms (1...255 ms).	%SW0
#FastPeriod	Supported PLC: Premium/Quantum/Modicon M340/ Modicon M580. Contains the period of this task. The period is expressed in ms (1...255 ms).	%SW1

Item	Description	Implicit Access
#WatchDogValue	Supported PLC: Any. The cycle of the MAST task is monitored by a software watchdog. This detects certain application errors (endless loops and so on) and ensures maximum duration for refreshing outputs. Triggering the watchdog results in a software fault (the application changes to Halt mode). You can define the value of the MAST task watchdog during configuration and the value is displayed in this word. Expressed in ms (10...500 ms), depending on the configuration. The other tasks are controlled by a watchdog whose value can be configured.	%SW11
#MastReqNb	Supported PLC: Modicon M340/Modicon M580/Unity Momentum. Number of requests processed per second by the MAST server irrespective of communication link used. It includes requests sent by Unity Pro, I/O scanner, communication EF, remote PLC, HMI, SCADA, and so on.	%SW26

Item	Description	Implicit Access
#HSBYStatus	Supported PLC: Premium/Quantum. HSBY status register: ● Bit 0...1: PLC mode: ○ This PLC is in Offline mode = 0 1 ○ This PLC is running in Primary mode = 1 0 ○ This PLC is running in Standby mode = 1 1 ● Bit 2...3: OtherPLCMode: ○ Other PLC in Offline mode = 0 1 ○ Other PLC running in Primary mode = 1 0 ○ Other PLC running in Standby mode = 1 1 ○ Other PLC is not accessible (switched off, no communication) = 0 0 ● Bit 4: Logic mismatch: ○ 0: PLCs have matching logic ○ 1: PLCs do not have matching logic ● Bit 5: PLC set: ○ 0: This PLC is set as unit A ○ 1: This PLC is set as unit B ● Bit 6: CPU-sync link status: ○ 0: The CPU-sync link is operating properly. The contents of bit 5 are significant. ○ 1: The CPU-sync link is not valid. In this case, the contents of the bit 5 is not significant because the comparison of the two MAC addresses cannot be performed. ● Bit 7: Main processor OS version matching status between primary and standby: ○ 0: OS versions match. ○ 1: OS versions mismatch. If OS version mismatch is not allowed in the command register (bit 4 = 0), the system does not work as redundant as soon as the fault is signaled.	%SW61

Item	Description	Implicit Access
#HSBYStatus	● Bit 8 (for Quantum only): This bit indicates if there is a Copro version mismatch between Primary and Standby: ○ 0: Copro OS versions match. ○ 1: Copro OS versions mismatch. ● Bit 9 (for Premium only): This bit indicates if at least one ETY module does not have the minimum version: ○ 0: All the ETY modules have the minimum version. ○ 1: Atleast one ETY module does not have the minimum version. In this case, no Primary PLC can start. ● Bit 10 (for Premium only): This bit indicates if there is a Monitored ETY OS version mismatch between Primary and Standby: ○ 0: No Monitored ETY OS version mismatch. ○ 1: Monitored ETY OS version mismatch. If OS version mismatch is not allowed in the command register (bit 4 = 0), the system does not work as redundant as soon as the fault is signaled. ● Bit 11: Reserved. ● Bit 12 (for Quantum only): This indicates the bit 13 relevance: ○ 0: Information given by bit 13 is not relevant. ○ 1: Information given by bit 13 is valid. ● Bit 13: IPaddress: ○ 0: This PLC has IP@. ○ 1: This PLC has IP@+1. ● Bit 14: Reserved. ● Bit15: Healthy: ○ 0: The CoPro interface is healthy. ○ 1: The CoPro interface is not healthy.	%SW61

Item	Description	Implicit Access
#CCOTFStatus	Supported PLC: Quantum/Modicon 580.	%SW66
	Status of an Ethernet I/O configuration change. Can be set either by CRP or by CPU. Low byte detailed status: ● 00: Idle ● 01: Inprogress ● 02: Completed ● 03: Failed, recoverable error ● 04: Failed, fatal error ● 05: Failed, CCOTF is rejected by the drop High byte: detailed status (set by CRP unless CPU is mentioned). ● 00: Idle ● 01: Request length invalid ● 02: Request header invalid ● 03: Request descriptor invalid ● 04: Request signature invalid ● 05: FDR server invalid ● 06: EIP scanner invalid ● 07: Header request ID invalid ● 08: Header drop ID invalid ● 09: Header device name invalid ● 0 A: Descriptor length invalid ● 0B: Descriptor RTE invalid ● 0C: Descriptor offset invalid ● 0D: Signature length invalid ● 0E: Signature data invalid ● 0F: Signature count invalid ● 10: FDR IP invalid ● 11: FDR subnet mask invalid ● 12: FDR gateway invalid ● 13: EIP CID invalid ● 14: EIP device number invalid ● 15: EIP IP invalid. ● 16: EIP vendor ID invalid. ● 17: EIP product type invalid ● 18: EIP product code invalid ● 19: EIP timeout invalid ● 1 A: EIP OT RPI invalid ● 1B: EIP TO RPI	

Item	Description	Implicit Access
#CCOTFStatus	• 1C: EIP path invalid • 1D: Process succeed • 1E: Process busy • 1F: Drop not exist • 20: Drop already exist • 21: Drop not reachable • 22: Process device manager error • 23: Process FDR builder error • 24: Process FDR server error • 25: Process EIP scanner error • 26: Process EIP signature mismatch • 27: Process EIP connection rejected • 28: Process unknown error • 29...3F: Undefined • 4C: Max CCOTF retries reached • 4D: Bad signature detected by the CPU • 4E CPU: Communication error with CRP • 4F CPU: IOPL error (build or swap) • 50: CRA received wrong communication parameter error • 51: FDR server did not respond • 52: Error when downloading PRM file from the server • 53 CRA: downloaded 0 size file from FDR server • 54: Bad configuration in PRM (bad CRC, invalid confirmation, Signature mismatch: managed by MC) • 55: PRM download timeout • 56: All other (example CCOTF count difference between new and old configuration more than 1) • 57...FE: Undefined • FF: Unknown error	%SW66

Item	Description	Implicit Access
#KeySwitch	Supported PLC: Quantum. This word provides the image of the switches on the Quantum CPU front panel. It is updated automatically by the system. ● X0 = 1: If the key switch is in memory protect (legacy Quantum) or lock (HE Quantum) position ● X1 = 1: If the key switch is in stop position (legacy Quantum only). ● X2 = 1: If the key switch is in start position (legacy Quantum only). ● X3...X7: Unused. ● X8 = 1: If the slide switch is in MEM position (legacy Quantum only). ● X9 = 1: If the slide switch is in ASCII position (legacy Quantum only). ● X10 = 1: If the slide switch is in RTU position (legacy Quantum only). ● X11...X15: Unused.	%SW71
#MaxReqNb	Supported PLC: Any. Maximum request number managed by the MAST server (UMAS, Modbus, UNI-TE). This word is initialized by the system with a value (NB) which depends on the PLC model. The possible values of the maximum request number are: 2 ← NB ← (NB+4). NB depends on PLC model. Example: On 572xxx PLC, NB = 8. The initial value is 8, the possible values are: 2, 3...12.	%SW90
#CardStatus	Supported PLC: Modicon M340/Modicon M580. Indicates the status of the card. ● 0000 = No error. ● 0001 = Application backup or file write sent to a write-protected card. ● 0002 = Card not recognized or application backup is damaged. ● 0003 = Backup of the application requested, but no card is available. ● 0004 = Card access error, for example, after a card has not removed properly. ● 0005 = No file system present in the card or file system is not compatible. When the error is fixed, the system sets the card status to 0.	%SW97

Item	Description	Implicit Access
#CPUErr	Supported PLC: Any. The last type of system fault encountered. ● 16#30 = System code fault. ● 16#53 = Time-out fault during I/O exchanges. ● 16#60...64 = Stack overrun. ● 16#65 = Fast task period of execution is too low. ● 16#81 = Detection of backplane error. ● 16#90 = System switch fault: Unforeseen IT.	%SW124
#CPUErrType	Supported PLC: Any. Contains the type of the last CPU execution fault. ● 16#2258 = Execution of HALT instruction. ● 16#DE87 = Calculation error on floating point numbers (%S18, these errors are listed in the word (%SW17). ● 16#DEB0 = Watchdog overflow (%S11). ● 16#DEF0 = Division by 0 (%S18). ● 16#DEF1 = Character string transfer error (%S15). ● 16#DEF2 = Arithmetic error (%S18). ● 16#DEF3 = Index overflow (%S20).	%SW125
#OpenConnectionsNb	Supported PLC: Quantum. The low byte of this word is unused. The high byte indicates the number of incoming TCP connections open on the Ethernet link TCP/IP port 502.	%SW128
#GlobalDataStatusValue	Supported PLC: Quantum. Global data status value.	%SW138

Array Diagnostics Specific Items

This table shows array elements of the specific items supported by Unity PLCs only:

Item	Description	Implicit Access
#RackIOStatus	Supported PLC: Premium/Modicon M580/Modicon M340.	%S40...47
`#RackIOStatus [0]`	Normally at 1 (state OK), the bit is set to 0 if the faults occur on the inputs/outputs of the rack 0.	%S40
...	...	...
`#RackIOStatus [7]`	Normally at 1 (state OK), the bit is set to 0 if the faults occur on the inputs/outputs of the rack 7.	%S47
#AuxPeriod	Supported PLC: Premium/Quantum/Modicon M580.	%SW2...5
`#AuxPeriod[0]`	AUX0 task period expressed in units of 10 ms.	%SW2
`#AuxPeriod[1]`	AUX1 task period expressed in units of 10 ms.	%SW3
`#AuxPeriod[2]`	AUX2 task period expressed in units of 10 ms.	%SW4 on Premium/ Quantum
`#AuxPeriod[3]`	AUX3 task period expressed in units of 10 ms.	%SW5 on Premium/ Quantum
#OSVersion	Supported PLC: Any	%SW14...16
`#OSVersion[0]`	The word contains the commercial version of the PLC. For instance, the meaning of the value 16#0135 is 01.35.	%SW14
`#OSVersion[1]`	The word contains the patch information for the commercial version. High Word: 00 (for openness). Low Word: Patch number corresponding to a letter. ● 0: No patch ● 1: A ● 2: B ... For instance, the meaning of the value 16#0003 is Patch C.	%SW15
`#OSVersion[2]`	The word contains the firmware internal version. For instance, the value 16#0043 is for ir 43.	%SW16
#MastTimes	Supported PLC: Any.	%SW30...32
`#MastTimes[0]`	Execution time of the last cycle of the MAST task in ms. This time corresponds to the time elapsed between the start (input acquisition) and end (output updating) of the task cycle. Interruption occurs due to higher priority tasks and processing of terminal requests.	%SW30
`#MastTimes[1]`	Maximum execution time of the MAST task (the maximum value of %SW30) measured since the last cold start, expressed in ms.	%SW31
`#MastTimes[2]`	Minimum execution time of the MAST task (the minimum value of %SW30) measured since the last cold start, expressed in ms.	%SW32

Item	Description	Implicit Access
#FastTimes	Supported PLC: Premium/Quantum/Modicon M580/M340. Modicon	%SW33...35
#FastTimes[0]	Same role as %SW30 for Fast task.	%SW33
#FastTimes[1]	Same role as %SW31 for Fast task.	%SW34
#FastTimes[2]	Same role as %SW32 for Fast task.	%SW35

Item	Description	Implicit Access
#Aux0Times	Supported PLC: Premium/Quantum/Modicon M580.	%SW36...38
#Aux0Times[0]	Same role as %SW30 for Aux0 task.	%SW36
#Aux0Times[1]	Same role as %SW31 for Aux0 task.	%SW37
#Aux0Times[2]	Same role as %SW32 for Aux0 task.	%SW38
#Aux1Times	Supported PLC: Premium/Quantum/Modicon M580.	%SW39...41
#Aux1Times[0]	Same role as %SW30 for Aux1 task.	%SW39
#Aux1Times[1]	Same role as %SW31 for Aux1 task.	%SW40
#Aux1Times[2]	Same role as %SW32 for Aux1 task.	%SW41
#Aux2Times	Supported PLC: Premium/Quantum.	%SW42...44
#Aux2Times[0]	Same role as %SW30 for Aux2 task.	%SW42
#Aux2Times[1]	Same role as %SW31 for Aux2 task.	%SW43
#Aux2Times[2]	Same role as %SW32 for Aux2 task.	%SW44
#Aux3Times	Supported PLC: Premium/Quantum.	%SW45...47
#Aux3Times[0]	Same role as %SW30 for Aux3 task.	%SW45
#Aux3Times[1]	Same role as %SW31 for Aux3 task.	%SW46
#Aux3Times[2]	Same role as %SW32 for Aux3 task.	%SW47
#CPUStopTime	Supported PLC: Any.	%SW54...58
#CPUStopTime[0]	Part of the time of the last power failure or PLC stoppage (in BCD), it contains the seconds and day of the week.	%SW54
#CPUStopTime[1]	Part of the time of the last power failure or PLC stoppage (in BCD), it contains the hour and minutes.	%SW55
#CPUStopTime[2]	Part of the time of the last power failure or PLC stoppage (in BCD), it contains the month and day.	%SW56
#CPUStopTime[3]	Part of the time of the last power failure or PLC stoppage (in BCD), it contains the century and year.	%SW57
#CPUStopTime[4]	The most significant byte contains the day of the week (1...7) of the last stop. The least significant byte contains a stop code: ● 1 = Change over from RUN to STOP by the terminal or dedicated input. ● 2 = Task watchdog overflow (%S11). ● 4 = Power loss or cartridge lock operation. **NOTE:** On Unity CPU, the stop date is not updated in case of cold start (reset button, card extract, and power on). ● 5 = Hardware fault. ● 6 = Software fault: Halt instruction, %S15, %S18, %S20, EF/EFB errors, SFC errors, application CRC check fail, undefined system function call, and so on.	%SW58

Item	Description	Implicit Access
#ReqCounters	Supported PLC: Any.	%SW87...89
#ReqCounters[0]	Number of requests processed per master (MAST) task cycle irrespective of communication link used.	%SW87
#ReqCounters[1]	For Premium: Number of requests processed by asynchronous server per master (MAST) task cycle. For other platforms: Number of HTTP requests received by the web server of the processor per second.	%SW88 on Premium/ Modicon M580/ Modicon M340/Unity Momentum
#ReqCounters[2]	For Premium: Number of requests processed by server functions (immediately) per master (MAST) task cycle. For other platforms: Number of FTP requests received by the FTP server per second.	%SW89 on Premium/ Modicon M580/ Modicon M340
#AppSign	Supported PLC: Modicon M340/Modicon M580/Unity Momentum. Contains a 32 bit value (%SW94 low word, %SW95 high word) that changes at every application modification except: ● Updating upload information. ● Replacing initial value by current value. ● Save parameter command.	%SW94...95
#AppSign[0]	Value low word	%SW94
#AppSign[1]	Value high word	%SW95
#ForcedObjects	Supported PLC: Any.	%SW108...109
#ForcedObjects[0]	Counts the number of forced discrete bits (%I, %Q, or %M). It is incremented each time a discrete bit is forced and decremented each time a discrete bit is unforced.	%SW108
#ForcedObjects[1]	Counts the number of forced analog channels. It is incremented each time an analog channel is forced and decremented each time an analog channel is unforced.	%SW109 on Premium/ Modicon M580/ Modicon M340

Item	Description	Implicit Access
#FIPIOCnxPointStatus	Supported PLC: Premium.	%SW128...143
`#FIPIOCnxPointStatus[0]`	State of a connected device on FIPIO bus. Normally set to 1, if one of those bits is reset to 0, it means that an error has occurred at the connection point level. %SW128 addresses from 0...15. %SW128:X0 → @0,%SW128:X1 →@1,..., %SW128:X15 →@15, when the error disappears, the bit is set to 1 by the operating system.	%SW128
...	...	...
`#FIPIOCnxPointStatus[15]`	State of a connected device on FIPIO bus. Normally set to 1, if one of those bits is reset to 0, it means that an error has occurred at the connection point level. %SW143 addresses from 240...255. %SW143:X0 → @240,%SW143:X1 → @241,..... , %SW143:X15 → @255, when the error disappears, the bit is set to 1 by the operating system.	%SW143
#EthWorkload	Supported PLC: Quantum.	%SW139...140
`#EthWorkload[0]`	● Lbyte: Percentage of load for IOScanning. ● Hbyte: Percentage of load for processing global data.	%SW139
`#EthWorkload[1]`	● Lbyte: Percentage of load for messaging services. ● Hbyte: Percentage of load for the other services.	%SW140
#IPAddress	Supported PLC: Quantum.	%SW141...142
`#IPAddress[0]`	IP address → 4 bytes in the following order: ● Lbyte: First byte. ● Hbyte: Second byte.	%SW141
`#IPAddress[1]`	● Lbyte: Third byte. ● Hbyte: Fourth byte.	%SW142
#IPSubNetMask	Supported PLC: Quantum.	%SW143...144
`#IPSubNetMask[0]`	IP subnet mask → 4 bytes in the following order: ● Lbyte: First byte. ● Hbyte: Second byte.	%SW143
`#IPSubNetMask[1]`	● Lbyte: Third byte. ● Hbyte: Fourth byte.	%SW144
#IPGateWay	Supported PLC: Quantum.	%SW145...146
`#IPGateWay[0]`	IP default gateway → 4 bytes in the following order: ● Lbyte: First byte. ● Hbyte: Second byte.	%SW145
`#IPGateWay[1]`	● Lbyte: Third byte. ● Hbyte: Fourth byte.	%SW146

Item	Description	Implicit Access
#MacAddress	Supported PLC: Quantum.	%SW147...149
#MacAddress[0]	Mac address: ● Lbyte: MAC address first byte. ● Hbyte: MAC address second byte.	%SW147
#MacAddress[1]	● Lbyte: MAC address third byte. ● Hbyte: MAC address fourth byte.	%SW148
#MacAddress[2]	● Lbyte: MAC address fifth byte. ● Hbyte: MAC address sixth byte.	%SW149
#ERIODropStatus	Supported PLC: Quantum.	%SW152...155
#ERIODropStatus[0]	Each bit indicates the status of an Ethernet Remote I/O connection point 0...15. The bit is set to 0 if at least one I/O module in the drop has detected error. It is set to 1 if all modules are operating correctly.	%SW152
...	...	...
#ERIODropStatus[3]	Each bit indicates the status of an Ethernet Remote I/O connection point 48...61 (maximum 62 Ethernet Remote I/O drops).	%SW155
#RackStatus	Supported PLC: Premium/Modicon M580/Modicon M340.	%SW160...167
#RackStatus[0]	● %SW160:X0: 0 in case of error on the device in the slot number 0 of the rack number 0. Else = 1. ● %SW160:X1: 0 in case of error on the device in the slot number 1 of the rack number 0. Else = 1. ... ● %SW160:X15: 0 in case of error on the device in the slot number 15 of the rack number 0. Else = 1.	%SW160
...	...	...
#RackStatus[7]	● %SW167:X0: 0 in case of error on the device in the slot number 0 of the rack number 7. Else = 1. ● %SW167:X1: 0 in case of error on the device in the slot number 1 of the rack number 7. Else = 1. ... ● %SW167:X15: 0 in case of error on the device in the slot number 15 of the rack number 7. Else = 1.	%SW167

Item	Description	Implicit Access
#GlobalDataStatus	Supported PLC: Quantum.	%SW168...171
`#GlobalDataStatus [0]`	Global data validity indicator: ● %SW168:X0: 0 in case of error on the device no 1, else = 1 ... ● %SW168:X15: 0 in case of error on the device no 16, else = 1.	%SW168
...	...	...
`#GlobalDataStatus [3]`	Global data validity indicator: ● %SW170:X0: 0 in case of error on the device no 49, else = 1. ... ● %SW170:X15: 0 in case of error on the device no 64, else = 1.	%SW171

Item	Description	Implicit Access
#ERIOCnxStatus	Supported PLC: Quantum.	%SW172...175
#ERIOCnxStatus[0]	Each bit indicates the status of an Ethernet Remote I/O connection point 0...15. The bit is set to 0 if the connection between the PLC and the drop is not operating correctly. It is set to 1 if the connection is operating correctly.	%SW172
...	...	...
#ERIOCnxStatus[3]	Each bit indicates the status of an Ethernet Remote I/O connection point 48...61 (maximum 62 Ethernet Remote I/O Drops). The bit is set to 0 if the connection between the PLC and the drop is not operating correctly. It is set to 1 if the connection is operating correctly.	%SW175

PLC operating mode management

Description

The PLC operating mode can be controlled using the specific item `#PLCStatus`. The modification of any PLC operating mode by the server can be enabled/disabled using the Configuration tool *(see page 123)*.

Changing the operating can change the system behavior.

⚠ **WARNING**

UNINTENDED EQUIPMENT OPERATION

Restrict access to the embedded server by configuring passwords.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The current operating mode of the PLC is viewed by reading the `#PLCStatus` specific item. Since this item can be activated, it is possible to monitor the PLC operating mode using it.

The current operating mode of the PLC is modified by writing the `#PLCStatus` specific item.

The following values are associated with the different operating modes of the PLC:

STOP: 0* RUN: 1* INIT: 2**

(*) Non operational for ORPHEE type PLCs,

(**) Non operational for Unity Pro or PL7 type PLCs.

NOTE: If the programming tool is connected to the device, the exclusive reservation (performed by PL7, Unity Pro or Concept) may stop the modification of the operating mode of the PLC. Modbus Plus devices have Data Master (DM) or Program Master (PM) modes. To modify the PLC operating modes of certain devices, it may be necessary to use PM mode.

Section 15.2
Detected Error management

Aim of this Section

The aim of this section is to introduce you to detected error management.

What Is in This Section?

This section contains the following topics:

Topic	Page
Feedback Mechanism	225
Objects outside Software Configuration	227

Feedback Mechanism

Description

- **the feedback mechanism consists of three parts:**
  - description of the result of the call (execution) of a primitive,
  - the description of the validity of an item: Quality flag,
  - the availability of a GetErrorString primitive used to call up the description label of an event from its code *(see page 393)*.

- **result of calling up a primitive:**
  - All the methods offered return a detected error code. The programming language used to carry out the OPC client can use it as a detected error code or trigger an exception (usually languages using OLE Automation, e.g. Visual Basic).

In particular, this means that an event detected by a "function" type primitive is not indicated to the caller by means of the value that it returns.

- **anomalies that can be returned are the following:**
  - E_xxx: standard detected errors defined by OLE and Win 32,
  - OPC_E_xxx: improperly operating specific to OPC,
  - OFS_E_xxx: improperly operating specific to the OFS server,
  - in addition to the action described above, some of the exposed primitives contain a `pErrors` parameter in their call interface (output parameter).

This `pErrors` parameter is defined for the primitives that can manage several items during the same call (e.g.: AddItems).

- **pErrors allows you:**
  - to log a report for each item (an element in the `pErrors` table),
  - to indicate an anomaly to the caller through a channel other than that for triggering the exceptions. Typically, when `S_FALSE` is returned, there is no triggering of exceptions, as the result from the primitive is of type success with a error code. In order to know on which item the event occurred, the `pErrors` parameter must be consulted.

For example, the `pErrors` parameter allows notification for the AddItems primitive that some of the items mentioned have an invalid syntax.

- **description of the validity of an item:**
  - primitives with `synchronous` and `cyclic` read contain the parameter `pQualities`, which describes the validity of the items concerned. They give a Quality attribute for each item.

For these primitives, this parameter comes in addition to the `pErrors` parameter. The Quality attribute of an item is a value on 8 bits composed of 3 fields : Quality, Substatus and Limit.

B7	B6	B5	B4	B3	B2	B1	B0
Quality		Substatus				Limit	

To obtain the detected error code which corresponds to the field concerned, it is advisable to apply the appropriate extraction mask and to study the value thus obtained.

- the Limit field (2 bits) is not managed,
- the Quality field (2 bits) which designates the validity of an item's value:

B7	B6	Quality	Meaning
0	0	Bad	The value of an item is incorrect for the reasons shown in the Substatus field
1	1	Good	The value of an item is correct
0	1	Uncertain	An anomaly has been detected on the item, but it is still "too early" to set it to Bad. Transitional status.

- the Substatus field (4 bits) which provides details on the Quality field, and whose significance varies according to the value (Bad, Good) of the Quality field.

The **Substatus** field for the **Bad** value of the Quality field:

B5	B4	B3	B2	Substatus	Meaning	Validity value
0	0	0	0	Non-specific	Incorrect value with no specific reason: various causes	0
0	1	1	0	Communication interruption	Incorrect value due to a communication interruption with the PLC	24

The **Substatus** field for the **Good** value of the Quality field:

B5	B4	B3	B2	Substatus	Meaning	Validity value
0	0	0	0	Non-specific	Correct value. No particular conditions	192

The **Substatus** field for the **Uncertain** value of the Quality field:

B5	B4	B3	B2	Substatus	Meaning	Validity value
0	0	0	0	Non-specific	A risk has been detected.	64

NOTE: For all the other values not mentioned in the above tables, please contact technical support.

Objects outside Software Configuration

Description

The OFS server does not have access to the software configuration of the applications it accesses.

If a group contains items that are outside the software configuration, it may not be read on other items that are compatible with the configuration. It is due to the fact that optimization algorithms are used in read requests.

In case of a table, the OFS server can not read the whole table, even if only one sub-element of its elements is outside configuration.

Example 1: Application in which 522 words have been configured: from %MW0 to %MW521. The read or write of a group containing table item %MW520:10 will not be possible for the whole of this item, even though the words %MW520 and %MW521 are in the configuration.

NOTE: The words %MW520 and %MW521 in this example can be accessed individually.

Example 2: Application in which 522 words have been configured: from %MW0 to %MW521.

An active group with the active items %MW0 (quality Good) and %MW500 (quality Good).

If the item %MW530 is added, %MW500 becomes "Bad" and %MW530 is "Bad" but %MW0 remains "Good".

Explanation: reading the whole active group requires 2 requests: one for %MW0 and another for %MW500 and %MW530.

The first request is still OK: %MW0 remains "Good".

However %MW500 and %MW530 are reported as being "Bad quality".

If the item %MW530 is deleted, %MW500 becomes "Good" again.

Chapter 16
Variables

Aim of this Chapter

The aim of this chapter is to introduce the product's various data types.

What Is in This Chapter?

This chapter contains the following sections:

Section 16.1
Data types

Different OPC data types

Description

The OPC data types handled by the OFS client (called "expected") can be different from the native data types of the variables within the device (called "canonical").

By default, at item creation, they are identical. However, you can opt for another type.

More specifically, conversions between canonical type arrays or 16 bit words and expected type character strings are supported, giving the user easy handling of character strings with PLCs (these do not have canonical type character strings):

- The conversion byte array -> string produces an ASCII string.

Section 16.2
Unity Pro Variables on OFS

Section Contents

This section presents the different Unity Pro variables available either directly (direct addressing) or by a symbol table (.XVM, .STU or .XSY).

What Is in This Section?

This section contains the following topics:

Unity Pro Variables Available Using OFS

At a Glance

OFS provides access to the following Unity Pro variable types:

- Elementary data type (EDT),
- Table, Structure,
- Derived data type (DDT),
- I/O derived data type (IODDT) (1),
- Elementary function block (EFB), derived function block (DFB) (1).

NOTE:
- Postfix S is used to read and write a character string-type variable in the form of a byte table (OCP-type VT_UI1 table). For example, %MW100:10;S.
- Postfix C is used to read and write a string-type variable in the form of a string of characters (OCP-type VT_BSTR). For example, %MW110:10;C.

Description

The following table shows the EDTs in Unity Pro:

Unity Pro data type	OPC data type	Variant type	Returned format
BOOL	BOOL	VT_BOOL	True/False
EBOOL	BOOL	VT_BOOL	True/False
INT	INT	VT_I2	16 bits
DINT	DINT	VT_I4	32 bits
UINT	UINT	VT_UI2	16 bits
UDINT	UDINT	VT_UI4	32 bits
REAL	Float	VT_R4	Floating IEEE
TIME	UDINT	VT_UI4	32 bits
DATE	UDINT	VT_UI4	32 bits
TIME_OF_DAY or TOD (1)	UDINT	VT_UI4	32 bits
DATE_AND_TIME (1)	DFLOAT	VT_R8	Double IEEE
STRING	Array of byte	Array of VT_UI1	2048 bytes max
BYTE	BYTE	VT_UI1	8 bits
WORD	UINT	VT_UI2	16 bits
DWORD	UDINT	VT_UI4	32 bits

(1) : Only by .STU-type symbol table.

Direct addressing data instances

At a Glance

Direct addressing data instances have a preset slot in the PLC memory or in an application-specific module. This slot is recognized by the user.

Access syntax

The syntax of a direct addressing data instance is defined by the **%** symbol followed by a **memory location prefix** and in certain cases some additional information.

The memory location prefix can be:

- **M**, for internal variables,
- **K**, for constants,
- **S**, for system variables,
- **I**, for input variables,
- **Q**, for output variables.

%M internal variables

Access syntax:

	Syntax	format	Example	Program access rights
Bit	%M<i> or %MX<i>	8 bits (Ebool)	%M1	R\W
Word	%MW<i>	16 bits (Int)	%MW10	R\W
Word extracted bit	%MW<i>.<j>	1 bits (Bool)	%MW15.5	R\W
double word	%MD<i>	32 bits (Dint)	%MD8	R\W
Real (floating point)	%MF<i>	32 bits (Real)	%MF15	R\W

<i> represents the instance number.

NOTE: The %M<i> or %MX<i> data detect edges and manage forcing.

Memory organization:
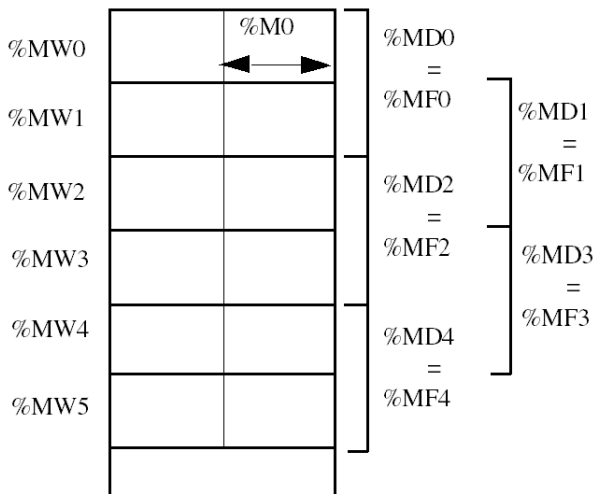
NOTE: Modification of %MW<i> entails modifications to the corresponding %MD<i> and %MF<i>.

%K constants

Access syntax:

	Syntax	format	Program access rights
Word constant	%KW<i>	16 bits (Int)	R
Double word constant	%KD<i>	32 bits (Dint)	R
Real (floating point) constant	%KF<i>	32 bits (Real)	R

<i> represents the instance number.

NOTE: The memory organization is identical to that of internal variables. It should be noted that these variables are not available on Quantum PLCs.

%I constants

Access syntax:

	Syntax	format	Program access rights
Bit constant	%I\<i\>	8 bits (Ebool)	R
Word constant	%IW\<i\>	16 bits (Int)	R

\<i\> represents the instance number.

NOTE: These data are only available on Quantum and Momentum PLCs.

%S system variables

Access syntax:

	Syntax	format	Program access rights
Bit	%S\<i\> or %SX\<i\>	8 bits (Ebool)	R\W or R
Word	%SW\<i\>	32 bits (Int)	R\W or R
Double word	%SD\<i\>	32 bits (Dint)	R\W or R

\<i\> represents the instance number.

NOTE: The memory organization is identical to that of internal variables. The %S\<i\> and %SX\<i\> data are not used for detection of edges and do not manage forcing.

I/O variables

These variables are contained in the application-specific modules.

Access syntax:

	Syntax	Example	Program access rights
I/O structure (IODDT)	%CH\<@mod\>.\<c\>	%CH4.3.2	R
Inputs %I			
Module error bit	%I\<@mod\>.MOD.ERR	%I4.2.MOD.ERR	R
Channel error bit	%I\<@mod\>.\<c\>.ERR	%I4.2.3.ERR	R
Bit	%I\<@mod\>.\<c\>	%I4.2.3	R
	%I\<@mod\>.\<c\>.\<d\>	%I4.2.3.1	R

	Syntax	Example	Program access rights
Word	%IW<@mod>.<c>	%IW4.2.3	R
	%IW<@mod>.<c>.<d>	%IW4.2.3.1	R
Double word	%ID<@mod>.<c>	%ID4.2.3	R
	%ID<@mod>.<c>.<d>	%ID4.2.3.1	R
Real (floating point)	%IF<@mod>.<c>	%IF4.2.3	R
	%IF<@mod>.<c>.<d>	%IF4.2.3.1	R
Outputs %Q			
Bit	%Q<@mod>.<c>	%Q4.2.3	R\W
	%Q<@mod>.<c>.<d>	%Q4.2.3.1	R\W
Word	%QW<@mod>.<c>	%QW4.2.3	R\W
	%QW<@mod>.<c>.<d>	%QW4.2.3.1	R\W
Double word	%QD<@mod>.<c>	%QD4.2.3	R\W
	%QD<@mod>.<c>.<d>	%QD4.2.3.1	R\W
Real (floating point)	%QF<@mod>.<c>	%QF4.2.3	R\W
	%QF<@mod>.<c>.<d>	%QF4.2.3.1	R\W
%M variables			
Word	%MW<@mod>.<c>	%MW4.2.3	R\W
	%MW<@mod>.<c>.<d>	%MW4.2.3.1	R\W
Double word	%MD<@mod>.<c>	%MD4.2.3	R\W
	%MD<@mod>.<c>.<d>	%MD4.2.3.1	R\W
Real (floating point)	%MF<@mod>.<c>	%MF4.2.3	R\W
	%MF<@mod>.<c>.<d>	%MF4.2.3.1	R\W
%K Constants			
Word	%KW<@mod>.<c>	%KW4.2.3	R
	%KW<@mod>.<c>.<d>	%KW4.2.3.1	R
Double word	%KD<@mod>.<c>	%KD4.2.3	R
	%KD<@mod>.<c>.<d>	%KD4.2.3.1	R
Real (floating point)	%KF<@mod>.<c>	%KF4.2.3	R
	%KF<@mod>.<c>.<d>	%KF4.2.3.1	R

<@mod = \<b>.<e>\<r>.<m>

<b> bus number (omitted if station is local).

<e> device connection point number (omitted if station is local).

<r> rack number.

<m> module slot

<c> channel number (0 to 999) or MOD reserved word.

<d> date number (0 to 999) or ERR reserved word.

NOTE: The above syntax is supported only on Premium, M340 local drop, and M580 families. For Unity Quantum and Unity Momentum PLCs, you must use flat addresses such as %I *(see page 235)* and %M *(see page 233)*

Examples: Local station and station on field bus.

%I0.4.1.5 (module 4 channel 1 data item 5)

No. 7

Bus No. 2

%I\2.7\0.5.1.5 (module 5 channel 1 data item 5)
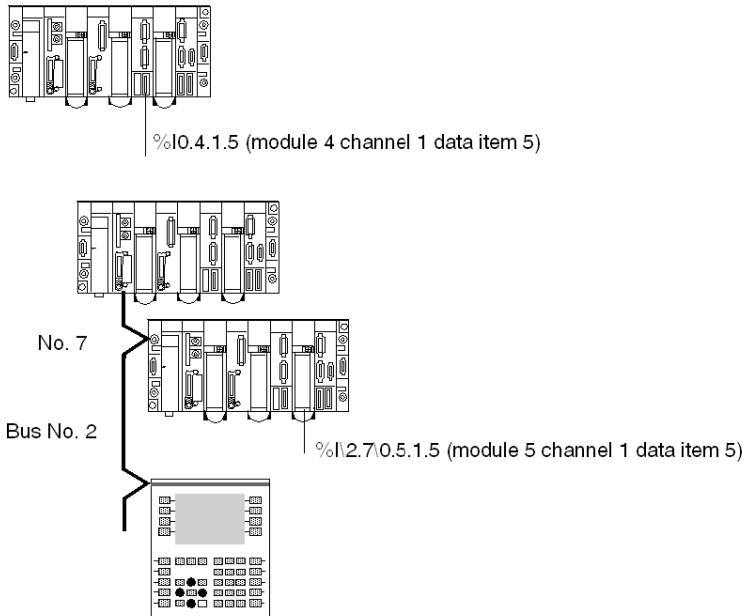
Section 16.3
PL7, XTEL and ORPHEE variables

Aim of this Section

The aim of this section is to introduce you to the different PL7 variables on OFS.

NOTE: Only memory objects of standard objects are accessible for series 7 (XTEL) and series 1000 (ORPHEE) PLCs. The syntax used on these PLC ranges have been recovered and are highlighted in Italics. They are only accessible with these types of PLCs.
For series 7 PLCs, the size of the request is limited to 32 bytes

Meanings of the terms used in the tables:

- - : not available,
- R: read only access,
- W: write access,
- R/W read/write access.

What Is in This Section?

This section contains the following topics:

Topic	Page
Standard Objects	239
Grafcet objects	242
Standard function blocks	243
Table objects	245

Standard Objects

System objects

The table below shows the system objects supported by OFS server:

Object	Syntax	TSX 37 / PCX/TSX 57 on X-Way	TSX 37 / PCX/TSX 57 on non-X-Way networks	TSX Series 7	TSX S1000
System bit	%Si	R/W	-	-	-
System word	%SWi	R/W	-	-	-
System Double word	%SDi	R/W	-	-	-

Memory objects (variables and constants)

The table below shows the memory objects supported by OFS server:

Object	Accepted syntax	TSX 37 / PCX/TSX 57 on X-Way	TSX 37 / PCX/TSX 57 on non-X-Way networks	TSX Series 7	TSX S1000
Internal bit	%Mi *%Bi* *%MXi*	R/W	R/W	R/W	R/W
Word extracted bit	%MWn:Xm	R	R	R	R
Memorized internal bit (S1000 specific)	%Rxi	-	-	-	R/W
Internal byte	%MBi	R	-	-	-
Internal word	%MWi *%Wi*	R/W	R/W	R/W	R/W
Internal double word	%MDi *%DWi*	R/W	R/W	R/W	R/W
Floating point (32 bits)	%MFi *%FDi*	R/W	R/W	R/W	R/W
Constant word	%KWi *%CWi*	R	-	R	-
Constant double word	%KDi *%CDi*	R	-	R	-
Constant floating point (32 bits)	%KFi *%CFi*	R	-	R	-

Object	Accepted syntax	TSX 37 / PCX/TSX 57 on X-Way	TSX 37 / PCX/TSX 57 on non-X-Way networks	TSX Series 7	TSX S1000
Common word on network 0	%NW{j}k j=station no. k=word no.	R/W	-	-	-
Common word on other networks	%NW{i.j}k i=network no. j=station no. k=word no.	R/W	-	-	-

I/O module objects

The table below shows the I/O objects supported by OFS server:

TSX 37 / PCX / TSX 57 on X-Way				
Object	Accepted syntax	I/O object	Extract bit	Table
Discrete input	%Ii.j[.r] %I\p.2.c\m.j[.r]	R	-	-
Discrete output	%Qi.j[.r] %Q\p.2.c\m.j[.r]	R/W	-	-
Input word	%IWi.j[.r] %IW\p.2.c\m.j[.r]	R	R	-
Output word	%QWi.j[.r] %QW\p.2.c\m.j[.r]	R/W	R	-
Double input word	%IDi.j[.r] %ID\p.2.c\m.j[.r]	R	R	-
Double output word	%QDi.j[.r] %QD\p.2.c\m.j[.r]	R/W	R	-
Floating input (32 bits)	%IFi.j[.r] %IF\p.2.c\m.j[.r]	R	R	-
Floating output (32 bits)	%QFi.j[.r] %QF\p.2.c\m.j[.r]	R/W	R	-
Channel error bit	%Ii.j.ERR %I\p.2.c\m.j.ERR	R	-	-
Module error bit	%Ii.MOD.ERR %I\p.2.c\m.j.MOD.ERR	R	-	-

- description for in-rack modules:
  - **i**: rack number*100 + number of position of module in the rack,
  - **j**: channel number,
  - **r** (optional): rank of the object in the channel.

- description for remote FIPIO modules:
  - **p**: 0 or 1: number of the position of the processor in the rack,
  - **2** : channel of embedded FIPIO processor,
  - **c**: connection point number,
  - **m**: 0 : "base" module (manages communication with the processor), 1: "extension" module (can be connected to the base module to double the number of I/Os),
  - **j**: channel number,
  - **r** (optional): rank of the object in the channel.

NOTE: The Fipio I/O objects are only accessible on PLCs programmed using PL7 via X-Way networks.

Grafcet objects

Description

Object	Syntax	TSX 37	PCX/TSX 57
Step state	%Xi	R	R
Activity time of a step	%Xi.T	R	R
Status of step in a macro step	%Xj.i	-	R
Activity time of a step in a macro-step	%Xj.i.T	-	R
State of the IN step of a macro-step	%Xj.IN	-	R
Activity time of the IN step of a macro-step	%Xj.IN.T	-	R
Status of the OUT step of a macro-step	%Xj.OUT	-	R
Activity time of the OUT step of a macro-step	%Xj.OUT.T	-	R

NOTE: The macro-steps are only available on PCX/TSX 57 version 3.0 or above.

Standard function blocks

Definition

See also PL7 *(see page 278)* blocks for R/W property modification.

PL7_3 timer: %Ti

Object	Syntax	TSX 37	PCX/TSX 57
Current value	%Ti.V	R	R
Preset	%Ti.P	R/W	R/W
Done Output	%Ti.D	R	R
Running Output	%Ti.R	R	R

IEC 61131-3 timer: %Tmi

Object	Syntax	TSX 37	PCX/TSX 57
Current value	%TMi.V	R	R
Preset	%TMi.P	R/W	R/W
Working Output	%TMi.Q	R	R

Monostable: %Mni

Object	Syntax	TSX 37	PCX/TSX 57
Current value	%MNi.V	R	R
Preset	%MNi.P	R/W	R/W
Running Output	%MNi.R	R	R

Up/Down Counter : %Ci

Object	Syntax	TSX 37	PCX/TSX 57
Current value	%Ci.V	R	R
Preset	%Ci.P	R/W	R/W
Empty Output	%Ci.E	R	R
Done Output	%Ci.D	R	R
Full Output	%Ci.F	R	R

Register: %Ri

Object	Syntax	TSX 37	PCX/TSX 57
Input word	%Ri.I	R/W	R/W
Output word	%Ri.O	R	R
Full Output	%Ri.F	R	R
Empty Output	%Ri.E	R	R

Drum: %Dri

Object	Syntax	TSX 37	PCX/TSX 57
Full Output	%DRi.F	R	R
Number of current step	%DRi.S	R	R
Activity time	%DRi.V	R	R

Table objects

Definition

Reminders:

The size of the tables is unlimited, excepting (system and memory) bit tables, which are limited to 450 elements.

System object tables

The table below shows the system object tables supported by OFS server:

Element type	Syntax	TSX 37 / PCX/TSX 57 on X-Way	TSX 37 / PCX/TSX 57 on non-X-Way networks	TSX Series 7	TSX S1000
System bit	%Si:L	R	-	-	-
System word	%SWi:L	R/W	-	-	-
System Double word	%SDi:L	R/W	-	-	-

NOTE: Access to system objects via table syntax is an extension as far as PL7 language is concerned. The system objects defined in the Micro and PCX Premium ranges are not all consecutive. This can limit access via table syntax in certain cases.

Memory object tables

The table below shows the memory object tables supported by OFS server:

Element type	Accepted syntax	TSX 37 / PCX/TSX 57 on X-Way	TSX 37 / PCX/TSX 57 on non-X-Way networks	TSX Series 7	TSX S1000
Internal bit	%Mi:L *%Bi:L* *%Mxi:L*	R/W	R/W	R W if module length is 8	R W if module length is 8
Internal word	%MWi:L *%Wi:L*	R/W	R/W	R/W	R/W
Double word	%MDi:L *%DWi:L*	R/W	R/W	R/W	R/W
Floating point (32 bits)	%MFi:L *%FDi:L*	R/W	R/W	R/W	R/W
Constant word	%KWi:L *%CWi:L*	R	-	R	-
Constant double word	%KDi:L *%CDi:L*	R	-	R	-

Element type	Accepted syntax	TSX 37 / PCX/TSX 57 on X-Way	TSX 37 / PCX/TSX 57 on non-X-Way networks	TSX Series 7	TSX S1000
Constant floating point (32 bits)	%KFi:L *%CFi:L*	R	-	R	-
Common word on network 0	%NW{j}k:L j = station no. k = word no.	R/W	R/W	-	-
Common word on other networks	%NW{i.j}k:L i = network no. j = station no. k = word no.	R/W	R/W	-	-
Character string	%MBi:L *%CHi:L*	R/W*	-	-	R/W**

(*) %MBi :L are R/W only if the address and length are even. If not, they are read-only.

(**) The size must be between 2 and the maximum size permitted by ORPHEE.

NOTE: Access to common words via table syntax is an extension as far as PL7 language is concerned.

NOTE: Limit: For a TSX 17 PLC, OFS cannot read bits whilst bits are being written. Also in this same PLC range, it is possible to read up to 16 words using OFS.

Grafcet object tables

The table below shows the Grafcet object tables supported by OFS server:

Element type	Syntax	TSX 37	PCX/TSX 57
Step state	%Xi:L	R	R
Activity time of a step	%Xi.T:L	R	R
Status of step in a macro step	%Xj.i:L	-	R
Activity time of a step in a macro-step	%Xj.i.T:L	-	R
Status of the IN step of a macro-step	%Xj.IN:L	-	R
Activity time of the IN step of a macro-step	%Xj.IN.T:L	-	R
Status of the OUT step of a macro-step	%Xj.OUT:L	-	R
Activity time of the OUT step of a macro-step	%Xj.OUT.T:L	-	R

NOTE: Beyond the "steps status", access to other Grafcet objects via table syntax is an extension as far as PL7 language is concerned.
Reminder:

Macro-steps are only available on PCX Premium version 3.0 or higher.

Additional information on macro-step tables:

- %Xj.i:L syntax reads several consecutive steps (number L) of the macro-step (j).

Example:

%X1.0:3 corresponds to %X1.0, %X1.1 and %X1.2.

- The syntax of a particular step (IN or OUT) in a macro-step (j) reads this step for several consecutive macro-steps (number L).

Example:

%X1.IN:3 corresponds to %X1.IN, %X2.IN and %X3.IN.

%X1.OUT.T:3 corresponds to %X1.OUT.T, %X2.OUT.T and %X3.OUT.T.

Section 16.4
Concept variables on OFS

Aim of this Section

The aim of this section is to introduce you to the different concept variables on OFS.

What Is in This Section?

This section contains the following topics:

Variables concept

Definition

State Ram Objects	Range	Access
Coils	0x	R/W
Input status	1x	R
Input Reg. as UINT	3x	R
Holding Reg. as UINT	4x	R/W
Holding Reg. as UDINT	4x	R/W
Holding reg. as REAL	4x	R/W

Symbols are supported throughout and all variables are represented by symbols, as there is no address syntax in the Concept language.

Relation between Concept basic data types and OPC data types:

Concept data type	OPC Data Type	Variant type	Returned format
BOOLEAN	BOOL	VT_BOOL	True/False
BYTE	BYTE	VT_UI1	8 bits
WORD	INT	VT_I2	16 Bit
INT	INT	VT_I2	16 Bit
UINT	UINT	VT_UI2	16 Bit
DINT	DINT	VT_I4	32 bits
UDINT	UDINT	VT_UI4	32 bits
FLOAT	FLOAT	VT_R4	Floating IEEE
TIME	DINT	VT_I4	32 bits

Structures are supported. It is possible to access a structure either by a bytes table (the user needs to know the internal fields and their type) or field by field with the following syntax:

<**Structure name**>.<**field name**>

In this case, the server finds out the data type directly from the Concept database.

NOTE:

- Structure access may only take place with a device associated with a Concept project file (*.prj) as a Symbol Table file. Both located and unlocated devices may be accessed.
- For easy manipulation of a structure, the user may create a group, and then one item for each field of the structure within this group.
- Access to unlocated variables and structures is only possible if the IEC runtime has been enabled in the PLC configuration *(see page 86)*,
- In addition, unlocated variables and structures must actually be used in the PLC application to be readable/writable with OFS. In fact, with Concept, any unused, unlocated variable is not recognized by the PLC. This is why OFS accepts the creation of an item linked to an unused, unlocated variable, but immediately defines its quality attribute as "Bad" to show it can no longer be read or written. It is possible to obtain automatic updates using the Concept programming tool and the DCC feature.
- A table item or an unlocated structure has "read only" access if the entire size of the table or of the structure exceeds 200 bytes.
- When an item represents an entire structure, it is considered a table.
- Postfix S is used to read and write a variable in the form of a byte table (OCP-type VT_UI1 table). For example, 400001:10;S.

Relationship between Concept variables and IEC 61131

At a Glance

It is possible to access certain Concept variables using IEC 61131 syntax. This does not concern located variables.

IEC 61131 to Concept:

%Mi	0x
%MWi	4x
%MFi	4x (access to 2 registers)
%MDi	4x (access to 2 registers)

Tables are also accepted.

Example:

The variable "Toto", located on register 400023, can also be accessed with %MW23 (UINT), %MF23 (Real) or %MD23 (UDINT). For %MF23 and %MD23, registers 23 and 24 are actually read. The syntax Toto:5 or %MW23:5 represents an array of five registers starting with Toto (=400023).

Section 16.5
Modsoft variables on OFS

Modsoft variables

Definition

The Modsoft supported syntax is limited to long addresses only (6 digits).

Example: 400001.

The following syntax ARE NOT supported (not to be confused with table syntax):

- 4:00001,
- 40001,
- 4x00001.

Any register located in the 6x range-id is not accessible.

The table syntax <reg. number>:<length> is supported for range-id 0,1,3,4.

It allows one or more registers to be read at once (actually <length> registers).

For Holding registers, it is possible to create a floating item or a Long integer using the postfix F or D. Two consecutive registers will be used. The usual R postfix can be used at the same time.

Example:

400001;S byte table for displaying character strings

400001;F floating point for registers 1 and 2

400012;D long integer (32 bits) for registers 12 and 13

400120;FR floating point considered as read-only for registers 120 and 121

Modsoft syntax

Object	Range	Item syntax	Access	Table	Max. size write
Coils	0	00000i	R/W	00000i:L	800
Input status	1	10000i	R	10000i:L	-
Input register	3	30000i	R	30000i:L	-
Holding register	4	40000i	R/W	40000i:L	100

Reminders: In read mode, the size of the tables is unlimited, except for bit tables (system and memory), which are limited to 2000 elements.

NOTE:

- The S postfix allows reading and writing a variable to a byte array (Array of VT_UI1 OPC type).

Section 16.6
Variables in general

Aim of this Section

The aim of this section is to introduce you to the different variables on OFS.

What Is in This Section?

This section contains the following topics:

Support of extracted bits

At a Glance

Generally speaking, the reading of extracted bits is supported for any variable of simple integer data type (including Concept unlocated variables):

The syntax is: <Variable definition>: Xn or <variable definition>, n for XTEL or <variable definition>.n for Unity Pro.

The bits are numbered from 0 to 7 for 8 bits integers, from 0 to 15 (for 16 bits integers) and from 0 to 31 for 32 bits integers.

Element types, access:

Element type	Accepted syntax	Concept	PL7	Unity Pro	Orphee or Xtel
Extracted byte bit	%MBi:Xj	-	R	see *Direct addressing data instances, page 233*	-
Word extracted bit	%MWi:Xj *%Wi,j*	R	R/W		R
Double word extracted bit	%MDi:Xj *%DWi:Xj*	R	R		R
System word extracted bit	%SWi:Xj	-	R		-
Constant extracted bit	%KWi:Xj	-	R		R (series 7 only)
Symbol extracted bit (single or double word)	Symbol:Xj	R	R		R (series 7 only)
Structure field extracted bit	Struct.member:Xj	R	-		-

Examples:

Unity Pro	PL7	CONCEPT	XTEL	ORPHEE	MODSOFT
see *Direct addressing data instances, page 233*	%MB100:X6 %MW100:X3 %MD200:X25 %SW6:X7 %KW100:X0 pump :X4	pump:X5 struct1.member:X8 tab1[1000]:X4	W100,3 DW200,25 CW100,0 Pump,4	%MW100:X3 %MD200:X25	300500:X11 400100:X12

Writing extracted bits is only possible for %MW variables on PCX/PMX Premium and micro, version 3.0 or later, on XWAY-type network, and is not supported on Modbus networks.

Local variables

Definition

There is a pseudo-protocol (driver name: "LOCAL") which allows for the creation of variables which are only local in relation to the server (unrelated to any hardware device). These local variables are always WORD (VT_12) type variables, created using a name.

Syntax: "LOCAL": ! <name>

Example: "LOCAL:!Bridge"

If two or more clients create the same local variable (the same name), its value is shared. This means that if a client modifies the value, the other client(s) will be notified (if notification has been activated). This function is generally used to exchange data from one client to the other.

Managing Variable Tables

Description

- the OFS server manages tables of variables. This provides easy access to a group of contiguous variables of the same type,
- the OFS server accepts several kinds of syntax, according to the target PLC: <**Origin Element**>:<**Length**>
  The <Origin Element> field represents either the address or the symbol of the first element in the table. The <Length> field represents the number of elements (of the same type as the origin variable) in the table.

Example for PL7 objects: For a variable with the address %MW10 and symbol MYARRAY.

A table of 20 elements starting with this variable may be referenced in the following two (equivalent) ways:

- %MW10:20
- MYARRAY:20

NOTE: This is the only syntax to allow the referencing of a table as a symbol for **PL7** objects, as the tables cannot be symbolized in PL7 language. The **Concept** and **Unity Pro** languages accept symbolic references to a table. This syntax can always be used with **Concept** and **Modsoft** variables.

- There are no limits to the size of the tables. However, they must not exceed the zones configured through the workshop,
- a table of variables corresponds to a single item of a group.

NOTE: During "cyclic" read of a group containing a table item, the OFS server sends the whole table to the client application, regardless of the number of elements in this table whose values have been changed.

Chapter 17
Symbols

Aim of this Chapter

The aim of this section is to describe symbol management within the OFS product.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
17.1	Symbol operation	260
17.2	Symbol management	266
17.3	Symbols and links	279
17.4	Symbols management through Direct PLC link	283

Section 17.1
Symbol operation

Aim of this Section

The aim of this section is to describe several features concerning symbols.

What Is in This Section?

This section contains the following topics:

The Different Groups of Items

Description

The OFS product has 2 types of groups:

- **User Group:**
  - an item may be localized on any device,
  - it is not possible to find out the number of requests needed to read the whole group,
  - Any one part of a group may be read,
  - the group is notifiable,
  - the name of a group may be any string of characters.

- **Synchronous Group:**
  - all items must be localized on the same device,
  - it is possible to find out the number of requests needed to read the whole group (specific items #NbrRequest),
  - even if the user executes the read function on a part of the group, all items are read,
  - the group is notifiable,
  - it is not possible to add any specific items other than #NbrRequest, nor any local variables, to a synchronous group
  - the declaration of items in the Push Data box is prohibited in synchronous groups (it is impossible to guarantee that items in Push Data and items in polling will be updated synchronously)
  - the group name must start with $ or $$,
  - the device timeout for the devices in use in synchronous groups must be set to 0 (this function must not be used).

$: number of requests limited to 1. The creation of items is prohibited when the maximum size of a read request is reached. A write request is refused when the items in the group targeted for reading exceeds the maximum authorized size (it should be noted that a write request, because it contains both the description of the items and the values to be written, is more restrictive as far as number of items is concerned).

$$: any number of requests, all linked to the same device.

A synchronous group may include the specific item "#NbrRequest" which enables the user to find out the number of communication requests needed to read all of the items in the group.

This item is read-only and may be read at any time, without needing to physically read the group (no time used up on network communication).

This item can only be used in a synchronous group.

NOTE: The system group *(see page 392)* function is used for compatibility reasons only. Avoid this as much as possible (it is of no use for an ordinary group).

Read Consistency

Definition

- Consistency of a group of items:
  TAll the items in a group are consistent with each other (i.e. read in the same PLC cycle) if and only if the group is transcribed on a single request. This means that the client application can be sure of the consistency of the items accessed in read mode when the #**NbrRequest** specific item associated with the group or the device equals 1 (synchronous group only)
  For more details refer to the section on performance .
  When the **prefix** '$' is used before the name of a group, the OFS server checks each time an item is added that the request number does not exceed the unit. These are known as **single request** user groups.
  During a write request, if the number of items from a synchronous group exceeds the size of a request, it will be refused in its entirety.

NOTE: The maximum quantity of lodge able items in a write frame is generally less than the lodge able quantity in a read frame. That is the reason why writing all of the items in a synchronous group can not be executed.

The OFS server (AddItems primitive) refuses to add the item and reports message if a single request group cannot be transcribed on a single request.

Write Consistency

Definition

The write primitive displayed by the OFS server allows one or several items to be written in a group at the same time. Obviously the items must be modifiable.

NOTE: During a write request, the OFS server overwrites the old values present in the PLC. The client application must take charge of the preliminary overwrite confirmation, if this is necessary.

If, during a write request concerning several items, there is some overlapping between items, you shouldn't be able to see what the write order will be. Write optimizer focus on performance rather than transmission order.

Example: If the write concerns items `%MW0:5` and `%MW0`, the values provided by the 3rd element of a `%MW0:5` item and by the 2nd item (`%MW2`) are taken into account, but the final value will be one or the other.

Consistency of the variables with each other during a write operation:

Write consistency is obtained when the data to write is lodged in the same network request, i.e. either table type variables or same type variables, whose addresses are contiguous and whose total size does not exceed the maximum size for a request.

Asynchronous Operation

Description

In asynchronous operation mode, a request for any asynchronous operation receives an immediate response. That does not mean that the operation requested has been completed, but that it has either been refused (incorrect code response), or that it is underway (correct code response).

The completion and the outcome of the operation will be announced using the notification mechanism. In order for this to occur, this mechanism must be activated before using asynchronous operation.

The following are the four operations:

- Read,
- Write,
- Refresh,
- Cancel.

Read/Write:

Similar to asynchronous operation with the same name (same functions, same restrictions).

Refresh:

Requests notification of all the values in progress of all the group's active items. The group must be active.

Cancel:

Stops the progress of a current read, write or refresh operation. It is not possible to know if the operation has actually been stopped.

Periodic Read Utility Installation

Description

Installation of the periodic read utility of a group's items consists of 4 stages:

- Subscription of the group to the notification service set up by the OFS server.
- Programming of the OnDataChange "waken" function, called by the OFS server to notify of changes in values that have occurred in the groups.
- Activation of all the items to be examined, if this has not already been done.
- Activation of the group to trigger regular examination of the group's items, which the OFS server is responsible for: ActiveStatus property set to TRUE. In terms of performance, it is advisable to activate the elements within a non-active group first and then activate the group. By doing this, you will avoid having too long a start up time due to numerous network requests.

Reminder:

The OnDataChange primitive receives notifications for **all groups** whose servers ensure read polling.

- **Notification** is done by group, and not individually for each item of a group. So the OnDataChange primitive receives the **list of the group's items** having changed value from one of the read polling function's iterations to another.

Stopping a group's periodic read utility is carried out in 2 phases:

- deactivation of the group: ActiveStatus property set to FALSE,
- stopping the group's subscription to notification service.

NOTE: For user groups: it is possible to activate/deactivate the group's item at any time. For synchronous groups: (name starting with $ or $$) all items are still regarded as active, in other words no partial activation/deactivation is possible.

Section 17.2
Symbol management

Aim of this Section

The aim of this section is to describe symbol management.

What Is in This Section?

This section contains the following topics:

Introduction to symbol management

Introduction

OFS server establishes symbol/address correspondence using a symbol file. This symbol file may have been created by a programming workshop (Concept, Modsoft, PL7, Unity Pro) or with an external tool such as a Text Editor (CSV format).

For devices in the S7 range, access to symbols is only possible by firstly converting corresponding applications to Premium applications.

Supported symbol file formats are the following:

- PL7 exported symbol table file (default extension SCY),
- PL7 exported application file (default extension FEF),
- Concept exported symbol table file (default extension CCN),
- Concept project file (default extension PRJ),
- Unity Pro located exported symbol table file (default extension XSY),
- Unity Pro exported symbol table file (default extension XVM),
- Unity Pro project file (default extension STU),
- Modsoft exported symbol table file (default extension TXT),
- CSV exported symbol table file (default extension CSV),
- Taylor exported symbol table file (default extension FIS).

For each format, only symbols that are associated with enough information for accessing variables are loaded and can be used (see below for details).

Symbol/address correspondence can also use a Concept *(see page 86)* or Unity Pro *(see page 85)* project file.

Several devices or groups can share the same symbol table file.

The link between the symbol file and a group of items is established either:

- by creating a link between a device and a symbol table. The Configuration tool is used for this:
  - ○ Creating an extension for the format you intend to use (e.g. .txt for Modsoft format),
  - ○ creating an alias for the device with the Configuration tool,
  - ○ linking the symbol table and this device.
- when the group is created, by entering the name and path of the symbol table.
  Symbol management is intended for a user group. Syntax of a group name: <Group name>[=<symbol table file path>].
  E.g.: creating group 1= C:\test.csv

OFS server will report a message to the client application if, while establishing this link, it detects that the neutral file does not exist or that it is invalid (its content is syntactically incorrect).

If a symbol file contains "collisions" (multiple declarations of the same symbol or address) the OFS server only retains the 1st occurrence of this identifier, and does not take following occurrences into account:

for example, if a symbol file contains the following associations:

- "PUMP" associated with "%MW0",
- "PUMP" associated with "%MW1",
  the OFS server will therefore consider the "PUMP" symbol to correspond only to %MW0.

NOTE:
- In all cases, the extension should have been configured *(see page 121)* beforehand.
- The use of symbols has no effect on the performance of the read write utilities of variables displayed by the OFS server. The only difference in performance concerns the group creation phase: The creation of a group of symbols is in fact longer, as it includes the translation of the symbols into addresses when creating the items in the group (AddItems primitive).
- Schneider Electric configuration softwares use XSY files to exchange data on variables (symbols based on located variables).

NOTE: The path locating the symbol file cannot contain extended characters or unicode characters.

Unity Pro exported symbols file

Procedure

To create such a file using the Unity Pro workshop:

- open the application using Unity Pro,
- open the application browser,
- open the data editor,
- open any window in this editor (e.g. FB variable and instance),
- use the File->Export menu to create the file.

This exported file authorizes the consistency check between the symbols file and the application in the PLC (see *Setting the Alias Properties, page 88* and *The PLC Software folder, page 122*).

Types of symbols available with the XVM files

For this type of file, access is possible for:

- Simple variables (EDT),
- Derived variables (DDT) if the DDT option has been enabled for exporting the application in Unity Pro,
- The inputs, outputs, inputs/outputs and public elements of derived function block instances (DFB).

I/O derived data types (IODDTs) are not supported.

NOTE: The Input and Output element descriptions of derived function blocks (DFB), elementary function blocks (EFB) and system sequential function charts (SFC) are supported by UnityPro from the V2.3 version. In order to access to these elements, you need to use an XVM symbol file generated by Unity Pro version V2.3 or upper.

Link with the XVM file

For an alias, the link with the XVM file uses the Unity Pro exported symbols table.

Application Consistency

The dynamic coherency check *(see page 102)* defines the procedure to follow if there is any variation between the application of the PLC and that of Unity Pro.

NOTE: When a project modification is transferred to the PLC, the consistency of XVM exported symbols with the Unity Pro file will only be accounted for after the file is manually exported by the user. The export operation may be automated by checking the 'XVM file' option in the menu '**Tools → Options → General → Automatic Backup During Transfer to PLC** from Unity Pro V2.0.2.

PL7 Exported Symbols Table File

Procedure

To create such a file using the PL7 software, proceed as follows:

● Open the application with PL7,
● Open the application browser,
● Open the data editor,
● Open any window in this editor (e.g. Memory Objects),
● Use the File->Export menu to create the file.

The exported file authorizes the consistency check (application name and version) between the symbols table file and the application in the PLC (see *Setting the Alias Properties, page 88* and *The PLC Software folder, page 122*).

For XWAY drivers, "Dynamic consistency" cannot be configured, the option is always disabled.

In "strict" level, the consistency is checked with the application name and the version.

In "Debug" level, no checking is performed and the qualities of the items are always in "good".

PL7 Exported Application File

Procedure

To create such a file using the PL7 software, proceed as follows:

- Open the application with PL7,
- Use the File->Export Application menu to create the file.

The exported file authorizes the consistency check (application name and version) between the symbols table file and the application in the PLC (see Device *(see page 88)* and PLC software folder *(see page 122)*. Consistency is checked only at device startup. If there is any inconsistency, all items of the device are positioned with the Quality field set to **Bad**. The OFS server does not use the configuration data of this file.

CONCEPT exported symbol table file

Procedure

To create such a file using the Concept workshop:

- open the application with Concept,
- use the **File->Export** menu,
- select Variables: text delimited,
- do not select a section,
- create the file with the .CCN *(see page 84)* extension.

The two other choices in the File-Export menu (Variables: Factory Link and Variables: Modlink) should be avoided.

Only access to located variables is possible using this kind of file since it does not contain the necessary information to access unlocated variables. For the same reason, access to structured variables is not possible.

MODSOFT exported symbol table file

Description

To create such a file with the Modsoft workshop:

- open the application with Modsoft,
- from the main menu, select "**Utility"->"Symbol Table**" > to open the symbol table editor,
- use the "**File I/O"->"Export**" menu to create the file.

This exported file does not authorize the consistency check (application name and version) between the symbols table file and the application in the PLC.

Modsoft applications may receive comments, using the comments section of the file. However, the OFS server only uses reference symbols.

Only symbols compliant with IEC format are supported. Symbols defined for the bits extracted from registers are not supported.

CSV symbol table file

Description

This type of file can be used with tools such as text editors (e.g. Notepad) or other tools (e.g. Excel 97 or later).

The format of each line is very simple:

`<Address><Separator><Symbol><Separator><Comments>`

- the `<Address>` should be a valid address for the device associated with that symbol file,
- the `<Separator>` can be a comma, a space or a tab character,
- the `<Symbol>` can be any string of characters without a comma/space/tab/special character.

In cases where certain special features are used (table length, special postfix such as R), add them to the address.

Example: table with 10 Read Only registers,

400001:10;R Table_Status

This file does not authorize the consistency check (application name and version) between the symbols table and the application in the PLC (see *Setting the Alias Properties, page 88* and *The PLC Software folder, page 122*). With Excel 97, use commas as separators.

NOTE: Maximum lengths are 50 characters for the address, 33 characters for symbols and 510 characters for comments.

TAYLOR exported symbol table file

Description

To create the symbols file using the Taylor workshop:

- open the application with the Taylor ProWORX 32 tool,
- select the project ProWORX 32, then right click on the mouse,
- select "Export Documentation",
- select the file type "ProWORX PLUS Symbol .FIS file",
- click on the Save button.

This exported file does not authorize the consistency check (application name and version) between the symbols table file and the application in the PLC (see *Setting the Alias Properties, page 88* and *The PLC Software folder, page 122*).

Browsing of symbols

At a Glance

Symbol browsing is supported by the OPC-Browse interface. It has a multi-level hierarchy:

A node for each device (Alias, Path, Symbols table) whether it is actually connected or not.

For each node:

- a sub-directory named "#Specific" for all specific items that can be created for this device,
- a sub-directory for each structured variable or array (Concept or Unity Pro project file only) which in turn has a sub-directory if the structure contains arrays or sub-structures.
- the complete list of application symbols declared in the symbols table file *(see page 84)* associated with the device *(see page 84)* or nothing (no symbol) if no symbols table has been declared for the device.

Devices which are connected but which have not been configured in the alias table can not be browsed.

Filtering possibilities exist to allow the user to select by type (for example, to ask for all Boolean variables), by name (wildcard '*' accepted), by rights of access, by located or unlocated character (Concept or Unity Pro project file only), by structured or non structured character.

The associated address and comments can also be obtained with each symbol ("&A" filter for the address and "&C" for the comments, or both "&A&C").

It is also possible to filter the variables using criteria based on their addresses.

Summary of the filter syntax (BNF syntax):

`<Symbol filter>[=<Address filter>][&A][&C][&E][&S][+<Filter on Customstring attribute>.`

`<Symbol filter>` any symbol string, including the wildcard '*',

`<Address filter>` any address string, including the wildcard '*',

`&A`: requests the display of the address,

`&C`: requests the display of the comments,

`&E`: only displays simple elements and not structures or arrays (for Concept project only),

`&S`: only displays structures and arrays (Concept or Unity Pro project file only).

Examples of filters:

T*	Requests all symbols beginning with T
B* &C	Requests all symbols beginning with B as well as any possible associated comments
* =%UL	For Concept or Unity Pro project file: requests only unlocated variables

* =%MW1*	Requests all variables with addresses beginning with %MW1
T* =%MX* &A&C	Requests all symbols beginning with T, with addresses beginning with %MX and requests the display of the address and comments

So that the browse interface can go faster (certain types of software require that all the symbols tables are opened when the browse interface is opened), a symbols table can be preloaded when the server is started. This option is selected with the configuration tool when an alias is created in the properties page.

NOTE: When browsing Unity Pro symbols of ANY_ARRAY type, only the first element of the table is visible.

Managing PL7 standard function blocks

Reminder

It is possible to modify the R/W fields of a Standard Function Block (e.g. "Preset" field of a %MNi.P Monostable), only if the Function Block has the "adjustable" property. "Adjustable" or "non adjustable" properties are assigned in the Configuration editor of the PL7 workshop.

During a write request from the R/W field of a standard Function Block, the OFS server does not carry out preliminary checking to make sure that the object has the "adjustable" property.

This means that if the Function Block does not have this property, the OFS server returns the generic detected error code.

Section 17.3
Symbols and links

Overview

This section describes the various links.

What Is in This Section?

This section contains the following topics:

Unity Pro Links

Introduction

In case of Unity PLCs, the symbol consistency is managed by:
- The variable access protocol
- An internal mechanism of OFS

The variable access protocol to Unity PLCs is based on an address in the PLC memory. The PLC responds only to the requests that are consistent with the PLC application. If after a modification of the PLC application, the variable mapping changes, OFS will still be able to communicate with this PLC after a resynchronization.

This mechanism doesn't detect the modifications of the PLC application without affecting the variable mapping.

OFS uses mechanism based on periodic polling of the PLC application stamps to check the consistency between the PLC application and the symbols. To enable this mechanism, check the dynamic consistency option. Once enabled, the mechanism will detect any change even the most minor, in the application. You can use the mechanism to link a SCADA application to a PLC application.

This mechanism is not available when you select direct synchronization by the PLC.

Description

To install the Unity Pro link via the OFS server, select the .stu project file.

This .stu file authorizes the consistency check (application name and version) between the symbol table file and the application in the PLC *(see page 102)*.

The Unity Pro workshop and .stu files can be located on different machines. The OFS server can be located either on the Unity Pro machine (usual case) or on another machine.

The same project can be used simultaneously with the Unity Pro workshop and OFS.

NOTE: In case of direct UnityPro utilisation, do not set the security of UnityPro to ON. Otherwise, the mandatory login will not be activated by the UnityPro server.

Concept Link

Description

Installation of the Concept link is only possible with Concept 2.2 SR2 or higher.

To install the Concept link, select the prj project file (see *Associating a Symbols Table File, page 84* and see *Symbol management, page 266*) as the symbols file for any device or group.

This prj file authorizes the consistency check (application name and version) between the symbol table file and the application in the PLC *(see page 102)*.

The Concept workshop and the prj files should always be located on the same machine. The OFS server can be located either on the Concept machine (usual case) or on another machine (Remote Concept Link feature).

The same project can be used simultaneously with the Concept and OFS Workshop as long as Concept is running in its own memory space (16 bit program).

In order to do this:

● edit the usual Concept shortcut properties,
● in the Shortcut tab, check the "Run in Separate Memory Space" box.

With OFS, more than one Concept project can be used at once, as long as they are from the same version of Concept. In order to do this, create the aliases required, and for each of these select a different project file.

OFS software, when used with the "stripped" Quantum executable file, will not read non-located variables.

If you anticipate using unlocated variables:

● use the full Quantum executable, not a stripped down version,
● activate IEC runtime on the PLC,
● check the **Unlocated media** option on the properties page. Otherwise, you will not be able to access unlocated variables.

CONCEPT Remote link

Description

The remote link offers exactly the same features as the normal Concept link. The only difference is that the Concept machine (where the Concept programming tool and the Concept project files are situated) is not the one on which the OFS server or the simulator is launched.

These machines must be linked by DCOM (usually on TCP/IP). An OFS server (with a license) or an OFS simulator (DEMO mode) must be installed on the Concept machine. An appropriate DCOM configuration must be carried out in order to enable access to this server which is called "proxy server".

On the OFS machine, when specifying a Concept project, open the device properties page in order to check the appropriate remote Concept option (the proxy server is either an OFS server or an OFS simulator) and give the complete access path of the Concept machine.

The path of the Concept project must be the same as that seen by the proxy server on the Concept machine (it must begin with the letter of a drive, followed by the complete path).

Section 17.4
Symbols management through Direct PLC link

Direct Resynchronization of PLC Symbol Database

Introduction

In some network architectures, the PC supporting the OFS server cannot directly access the Unity Pro project files (STU) or Unity Pro variable export file (XVM). As a result, when an on-line modification is performed through Unity Pro connected to the PLC, the OFS server detects an inconsistency but is unable to resynchronize the application variable layout to maintain the animation of the symbolized variable and to add new items.

However, Unity Pro can optionaly build and download the application with the embedded **DataDictionary** which includes the symbolized variable layout. Then the OFS server can automatically retrieve the consistency by accessing this **DataDictionary** in the CPU memory.

In addition, the **DataDictionary** allows OFS to implement a synchronization mechanism between Unity Pro and the CPU in order to avoid the animation break of the symbolized variables during the resynchronizations.

Due to communication limitation, and because the **DataDictionary** is generated from Unity Pro, the automatic resynchronization is available with devices using **TCP IP direct** driver over Ethernet network and **UNITY** PLCs.

The Quantum Safety CPU does not implement the data dictionary.

NOTE: The comments, the IODDT, custom strings and the private members of the DFB are not supported by the Data Dictionary.

Three different operating modes are proposed depending of the Unity Pro and OFS setting and versions:
- partial resynchronization
- full resynchronization
- resynchronization without animation break

Partial Resynchronization

The browse operation of OFS still functions through the Unity Pro project files (STU) or Unity Pro variable export file (XVM). When an inconsistency is detected, OFS resynchronizes the addresses of the animated variables from the **Data Dictionary**. In case of incomplete resynchronization, OFS retrieves the addresses of the variable from Unity Pro through STU or XVM files.
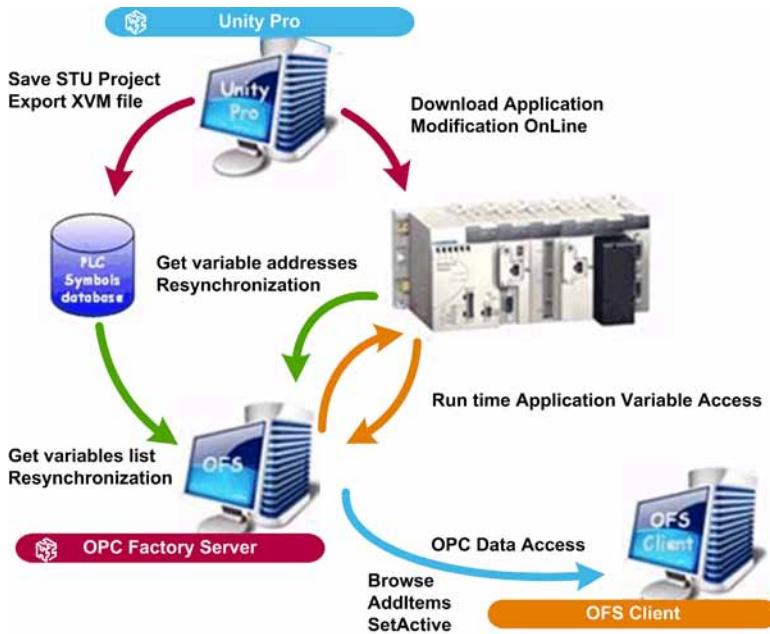
This operating mode is selected by providing the **Symbol table file** and by checking the **Using Data Dictionary** setting in the OFS configuration for the corresponding device:

The **Data Dictionary** setting must be checked in the **PLC Embedded Data** section of the Project Setting in the Unity Pro application. In addition, to get an updated symbol file, the **Project autosaving on download** setting has to be checked.

To take the advantage of this feature, the minimum required versions are:

	Unity Pro	CPU(*)	OFS
M580	V8.0	V1.0	V3.50
M80	V8.0	V1.0	V3.50
M340	V4.1	V2.1	V3.33
Premium	V4.1	V2.7	V3.33
Quantum	V4.1	V2.7	V3.33
(*) not available for Quantum Safety CPU.			

Full Resynchronization

Without any symbol file, the OFS server browses the application variables by using the Data Dictionary embedded in the CPU memory as well as when an inconsistency is detected after an on-line modification.
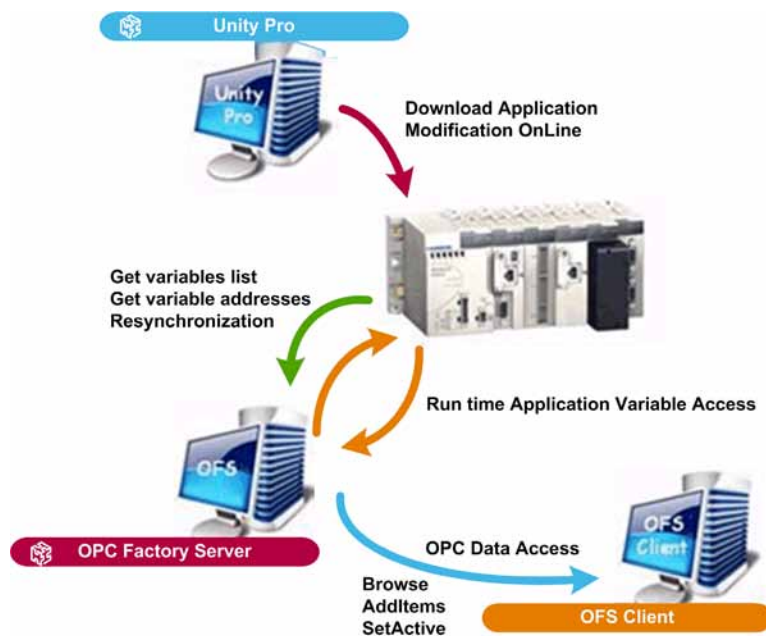
This operating mode is selected by checking the **Using Data Dictionary** setting without any **Symbol table file**:

	General	
	Symbol table file	...
	PLC Embedded Data	☑ Using Data Dictionary ☐ No Communication Break

The **Data Dictionary** setting must be checked in the **PLC Embedded Data** section of the Project Setting in the Unity Pro application.
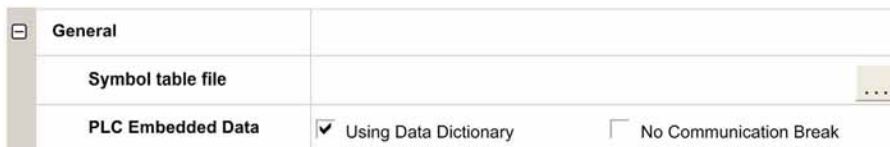
To take the advantage of this feature, the minimum required versions are:

	Unity Pro	CPU(*)	OFS
M580	V8.0	V1.0	V3.50
M80	V8.0	V1.0	V3.50
M340	V4.1	V2.2	V3.34
Premium	V4.1	V2.8	V3.34
Quantum	V4.1	V2.8	V3.34
(*) not available for Quantum Safety CPU.			

NOTE: For a device configuration using a **DataDictionary** without any attached symbol file, if the **DataDictionary** is not reachable for any reason (for example, the data dictionary is not embedded in the CPU because Unity Pro has not built it, or the **Browse** request is not supported because the CPU firmware is not at the right level), then the OFS server can only provide the direct located addresses on this device. In this case, the symbolic accesses are not available.

NOTE: For a device configuration using a **DataDictionary** without any attached symbol file, the following behavior is to be considered when the device is missing at OFS startup. After a certain amount of connection retry failures, the **DataDictionary** preload is cancelled. As a consequence, any IOPCItemMgt::Additems or IOPCItemMgtValidateItems operation will fail until the device is connected. This operating mode allows OFS to not wait for the preconfigured **DataDictionary** timeout (240 seconds) and improves OFS startup duration. The case above will not occur if Partial Resynchronization configuration is used.

NOTE: On configurations using **DataDictionary** and where the device is known to be missing before OFS startup, it is strongly recommended to check the OFS Server Settings \ Communication \ Advanced features \ Quick Item Validation in OFS Configuration tool.

Resynchronization without Animation Break

A build change done through a connected Unity Pro breaks the communication during the symbol database reload and the inconsistency detection. As the result, all the quality of animated items is set to BAD. To avoid this disturbance, a synchronization mechanism is set-up between OFS / Unity Pro and the CPU based on a preload of the Data Dictionary. The synchronization mechanism is limited by a build change time-out.

This operating mode is selected by checking the **No Communication Break** setting in the partial or full resynchronization mode:

General	
Symbol table file	D:\Applications\PLC_Appli_File.XVM ...
PLC Embedded Data	✔ Using Data Dictionary ✔ No Communication Break

General	
Symbol table file	...
PLC Embedded Data	✔ Using Data Dictionary ✔ No Communication Break

The **Preload on build change** setting must be checked in the **PLC embedded data** section of the Project Setting in the Unity Pro application. The **Effective Build change time-out** setting gives a time limit to apply the change after a time-out.

To take the advantage of this feature, the minimum required versions are:

	Unity Pro	CPU[(*)]	OFS
M580	V8.0	V1.0	V3.50
M80	V8.0	V1.0	V3.50
M340	V6.0	V2.3	V3.35
Premium	V6.0	V2.9	V3.35
Quantum	V6.0	V3.0	V3.35
(*) not available for Quantum Safety CPU.			

Chapter 18
The Diag Buffer

Overview

This chapter describes the Diag Buffer detection tool.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
18.1	Description of the Diag Buffer	290
18.2	Diag Buffer for Unity Pro	292
18.3	Diag Buffer for PL7	318

Section 18.1
Description of the Diag Buffer

Definition of the Diag Buffer

General

The **Diag Buffer** detects anomalies on monitored elements and transmits messages to the display systems.

This function is only provided for Premium TSX57/PCX57 PLCs programmed with PL7 and those programmed by Unity Pro that have a minimum software version (see PL7 / Unity Pro documentation for further information).

It lets you view an alarm being triggered in real time and gives all the characteristics of the alarm triggered in a bytes table:

- type of detected error,
- start date and time,
- end date and time,
- trigger zone between 0 and 15 (in case several modules are declared on the same PLC),
- alarm comments...

Illustration

The diagram below shows the functioning of the **Diag Buffer**:
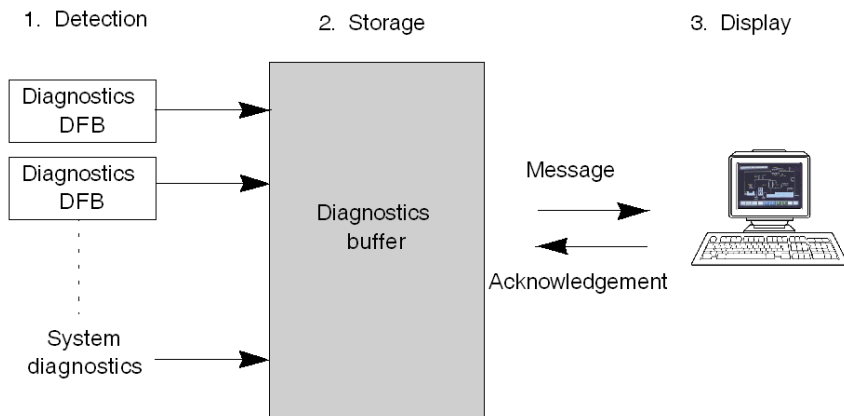
Operation

The table below describes the different operating phases:

Phase	Description
1	The diagnostics DFB integrated in the applications program or system detect when the process is not operational.
2	Detected faults are stored in a buffer memory called **Diag Buffer**.
3	One or more multi-station viewers (15max) allow you to: ● view of one or more areas of a PLC, ● view one or more areas of several PLCs, ● acknowledge messages, ● view the evolution of an item's status.

For more information on the **Diag Buffer**, see the section on **Diag Buffer** installation *(see page 318)*.

Section 18.2
Diag Buffer for Unity Pro

Aim of this Section

This section deals with the installation of the Diag buffer in Unity Pro PLC and its main features. The Diag Buffer is only available on Unity Pro dedicated PLCs.

What Is in This Section?

This section contains the following topics:

Operation from an OPC Client

Reminder about the Diag Buffer

The Diag buffer *(see page 290)* is a feature which detects faults on monitored elements and transmits messages to the visualization system (known as the viewer).

These messages are stored in the PLC's buffer memory.

NOTE: The diagnostic System or Application must be enabled in the application for Diag Buffer functioning.

Description of the Client Interface

Diag Buffer functions authorize access to PLCs using specific items.

The table shows the specific items:

Service	Item	Type	Access	Value read	Value to write
Open connection	#DiagLogon	VT_UI2	READ/WRITE	Viewer or FFFF hex identifier	zone number
Close connection	#DiagLogout	VT_UI2	READ/WRITE	Viewer or FFFF hex identifier	not important
Read next detected error	#DiagReadNextErrorU	VT_UI1+VT_ARRAY	READ	Detected error	–
Message acknowledgment	#DiagAckError	VT_UI2	WRITE	–	Detected error ID number see *Information Retrieved by the Diag Buffer at the Top of the Table, page 311*

Service	Item	Type	Access	Value read	Value to write
Evolution Status	#DiagReadStatusU	VT_UI1+VT_ARRAY	READ/WRITE	The result buffer from the data sent by write	On Write, user sent to OFS the address and the length to read (an ushort for BlockID, an ulong for offset in block and an ushort for size to read)
Delete a detected error in PLC	#DiagResetError	VT_UI2	WRITE		ErrorID
Clear the whole DiagBuffer in PLC	#DiagResetAll	VT_UI2	WRITE		No parameter for this request
Get all the FaultCause	#DiagGetFltCse	VT_UI2	WRITE		ErrorID
Get the name of FaultCause read with the previous specific item	#DiagFltCseResult	VT_UI1+VT_ARRAY	READ	Result of #DiagGetFltCse	
Ask to PLC to rebuild the list of FaultCause	#DiagRetriggError	VT_UI2	WRITE		ErrorID

Type corresponds to OPC standards:

- VT = variable,
- UI1 = unsigned integer on 1 byte,
- UI2 = unsigned integer on 2 bytes,
- UI4 = unsigned integer on 4 bytes,
- ARRAY = table of bytes

#DiagLogon Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	R/W	no	–

This item enables connection to the PLC. Firstly, the number of the zone that you wish to monitor must be indicated on the PLC (between 0 and 15) by carrying out a WRITE function.

Example of a write on `#DiagLogon`:

Value to write:

- bit i = 1: the zone is displayed,
- bit i = 0: the zone cannot be displayed.
  Bit 0 corresponds to zone 0, bit 15 corresponds to zone 15.
  Examples:
  - to monitor zone 6: write 0040 hex
  - to monitor zones 2 and 15: write 8004 hex

Value returned after read:

- the viewer number is displayed if the connection is open, if not the connection is not established and FFFF hex is returned.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_MMI_ALREADY_CONNECTED	The viewer is already connected

HRESULT	Comment
OFS_E_DIAG_BUFFER_FULL	FULL Diag buffer is full
OFS_E_DIAG_TOO_MUCH_MMI	All possible viewers (15) are connected

NOTE: To monitor all zones, write FFFF hex or 0 in `#DiagLogon`.

#DiagLogout Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	R/W	no	–

This item allows PLC disconnection.

Value to write:

● not important,

Value returned after read:

● if the disconnection is successful, FFFF hex is returned, if not the viewer number is returned again.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer is not activated
OFS_E_DIAG_BUFFER_FULL	Diag buffer is full
OFS_E_DIAG_WRONG_MMI_ID	The viewer identifier is not valid (outside range 1 to 15)
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected

NOTE: Destruction of the `#DiagLogon` item will automatically disconnect you from the viewer, without using the `#DiagLogout` item.

#DiagReadNextErrorU Specific Item

Type	Access	Can be activated	Limitation
VT_UI1 + VT_ARRAY	R	yes	–

This item enables you to read the list of detected errors in the diag buffer memory.

Value to write:

- nothing,

Value returned after read:

- detected errors saved in the form of a 550 byte table *(see page 306)*.

Value returned by the item:

HRESULT	Comment
S_OK	Read successful, no modification is recorded in the 120 byte table, or read successful, modifications recorded in the 120 byte table (the anomaly has been acknowledged or has disappeared), or Read successful, a new table has been created (a new anomaly has appeared).
OFS_E_DIAG_NO_BUFFER	Diag buffer is not activated
OFS_E_DIAG_BUFFER_FULL	Diag buffer is full
OFS_E_DIAG_WRONG_MMI_ID	The viewer identifier is not valid (outside range 1 to 15)
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected

#DiagAckError Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	W	no	–

This item allows alarm acknowledgment.

Value to write:

- the value on 2 bytes corresponding to the "Identification number" zone, the second and third byte (from zero) of the table.

Value returned after read:

- nothing.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer is not activated
OFS_E_DIAG_BUFFER_FULL	Diag buffer is full
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected
OFS_E_DIAG_WRONG_ERROR_ID	Non authorized anomaly identifier
OFS_E_DIAG_ERROR_NOT_USED	No element corresponds to this identifier

#DiagReadStatusU Specific Item

Type	Access	Can be activated	Limitation
VT_UI1 + VT_ARRAY	R/W	no	–

This item lets you to know the evolution of the status of a FB anomaly without having to wait to be notified of a change in the bytes table *(see page 310)*.

Value to write:

- value on 8 bytes corresponding to the status address and length.
  E.g. the value returned in the "Length of Status" zone (fourth byte from zero) of the table and the field "Status Address" from FB specific data.
  Var[8] = 98 hex, Var[9] = 01 hex, Var[10] = 76 hex, Var[11] = 25 hex
  The value to write in the #DiagReadStatus item is 25760198 hex or 628490648 dec.

Content	Size	Comment
Copy of Status Address field	6 bytes	Memory address for PLC
Size of Status	2 bytes	Byte length of status byte 00 hex

Value returned after read:

- the dump of memory from specified address (with the specified length).

#DiagReset Error Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	W	no	–

This item allows alarm suppression on PLC buffer.

After deletion, PLC's DiagBuffer updates the alarm state, so user will get a new buffer for this alarm on `#DiagReadNextError` item.

Value to write:

- the value on 2 bytes corresponding to the "Identification number" zone, the second and third byte (from zero) of the table.

Value returned after read:

- nothing.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected
OFS_E_DIAG_WRONG_ERROR_ID	Non authorized anomaly identifier
OFS_E_DIAG_ERROR_NOT_USED	No element corresponds to this identifier

#DiagResetAll Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	W	no	–

This item allows the drain of PLC buffer.

NOTE: Doing so, all viewers are disconnected. Use again `#DiagLogon` to get new alarm.

Value to write:

- not important.

Value returned after read:

- nothing.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected

#DiagGetFltCse Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	W	no	–

This item prepares `FaultCause` identification.

Value to write:

- the value on 2 bytes corresponding to the "Identification number" zone, the second and third byte (from zero) of the table.

Value returned after read:

- nothing.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected
OFS_E_DIAG_WRONG_ERROR_ID	Non authorized anomaly identifier
OFS_E_DIAG_ERROR_NOT_USED	No element corresponds to this identifier

#DiagFltCseResult Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	R	no	–

This item allows read of `FaultCause` name.

Value to write:

- nothing.

Value returned after read:

- an array of byte containing the all the name of `FaultCause`, each name is encoded like:
  - ○ first and second bytes are number of `FaultCause` Name,
  - ○ third and fourth bytes are TotalLen for name part

  For each name, the first byte is length of the name, then one byte for each characters of the name.

Example of result buffer:

Byte N°	Value	Signification
0	1	Word value (Low byte - High Byte) 1 `FaultCause`
1	0	
2	12	Word value (Low byte - High Byte) 12 bytes used after this field
3	0	
4	11	11 characters in the name
5	69	ASCII value for 'E'
6	86	ASCII value for 'V'
7	95	ASCII value for '_'
8	68	ASCII value for 'D'
9	73	ASCII value for 'I'
10	65	ASCII value for 'A'
11	95	ASCII value for '_'
12	84	ASCII value for 'T'
13	114	ASCII value for 'r'
14	117	ASCII value for 'u'
15	101	ASCII value for 'e'

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected
OFS_E_DIAG_WRONG_ERROR_ID	Non authorized anomaly identifier
OFS_E_DIAG_ERROR_NOT_USED	No element corresponds to this identifier

#DiagRetriggError Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	W	no	–

This item allows alarm retrigger (build again the list of `FaultCause`).

After this retrigger, the PLC DiagBuffer updates the alarm, so user gets a new buffer for this alarm on `#DiagReadNextError` item. But this alarm still shows the original number of `FaultCause`, not the updated number.

Value to write:

● the value on 2 bytes corresponding to the "Identification number" zone, the second and third byte (from zero) of the table.

Value returned after read:

● nothing.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected
OFS_E_DIAG_WRONG_ERROR_ID	Non authorized anomaly identifier
OFS_E_DIAG_ERROR_NOT_USED	No element corresponds to this identifier

Description of Client Operation

This diagram shows how an OPC client operates using specific items:
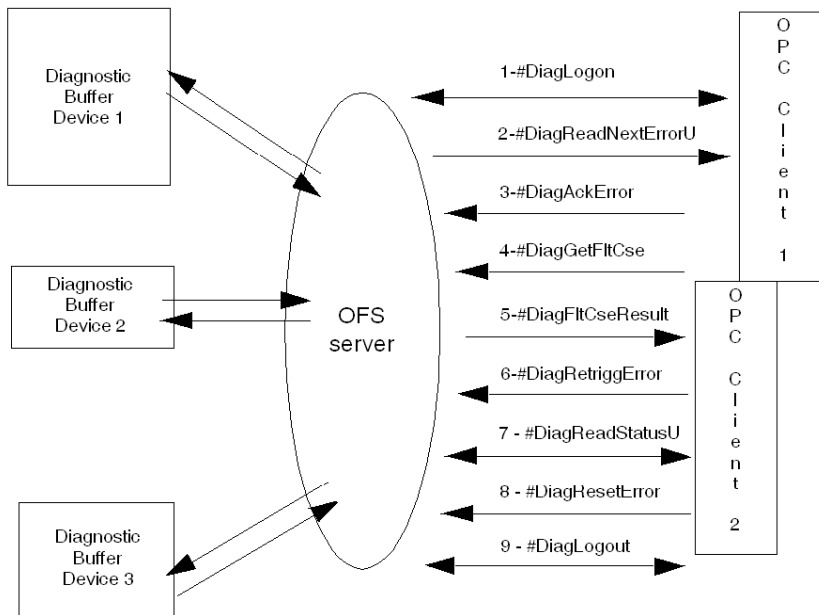
With the OFS server several PLCs can be monitored at the same time, it has a multi-station function (unlike the Unity Pro which can only manage one PLC at a time). To supervise several PLCs at once, simply create other aliases in the configuration tool and add them to another group belonging to the same client (minimum of 1 group per device to monitor).

Diag Buffer Management

Anomalies recorded in the diag buffer memory can have the following statuses:

- active or inactive,
- acknowledgment requested or acknowledgment not requested,
- if acknowledgment is requested, acknowledged or not acknowledged.

NOTE: Only anomalies from the diag buffer can be acknowledged. An anomaly displayed on several viewers is deleted from all viewers once it has been acknowledged on one viewer.
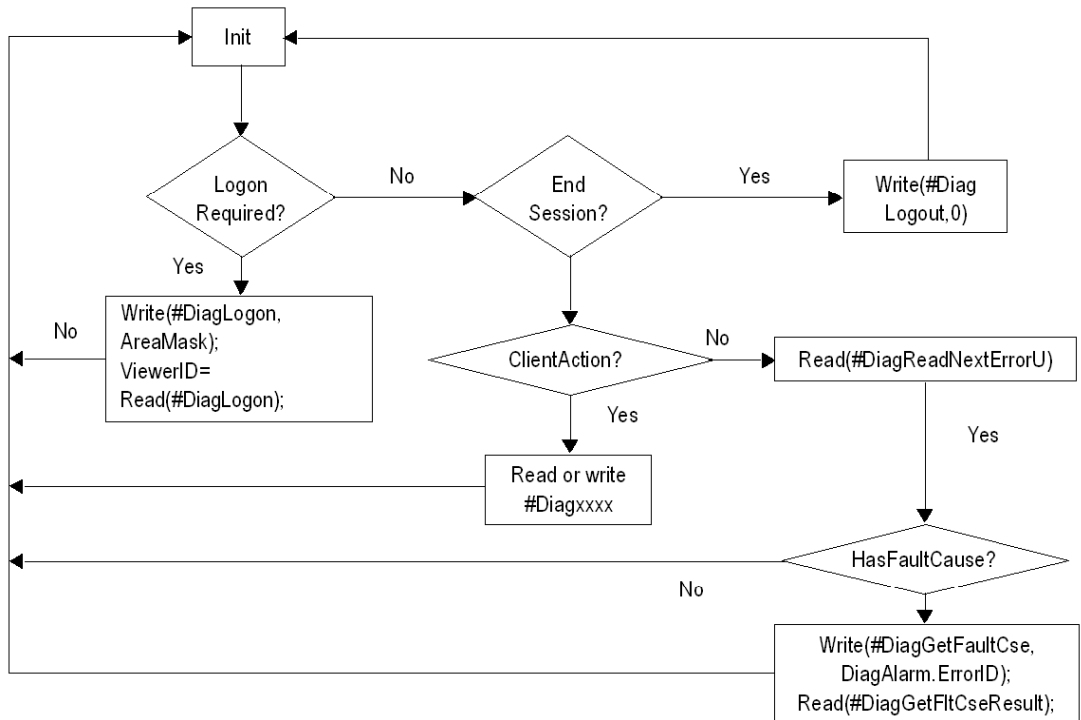An alarm is deleted from the buffer if:

- the alarm no longer exists,
- all viewers have read the alarm,
- the alarm has been acknowledged (after an acknowledgment request).

Description of client sequencing

Description of client sequencing

The graph below shows how and in which order an OPC client uses the specific items of Unity Pro Diag Buffer:

NOTE: Read the causes of an event directly before the next one. It permits to read the causes in any case.

Life Cycle of a Diag buffer alarm in a PLC

The graph below shows the life cycle of an alarm inside the Diag Buffer of Unity's PLC. We can see that a client which does not acknowledge the alarm or which read very slowly the alarm can force the PLC to maintain the alarm inside the Diag Buffer (with risk of overflowing), until the alarm is read (by each connected client) and aknowledged (if necessary). We can also see that an alarm is no longer available if its state is inactive and it has been read. So we can no more call **#DiagReadStatus** or **DiagGetFltCse** on this alarm, for example.
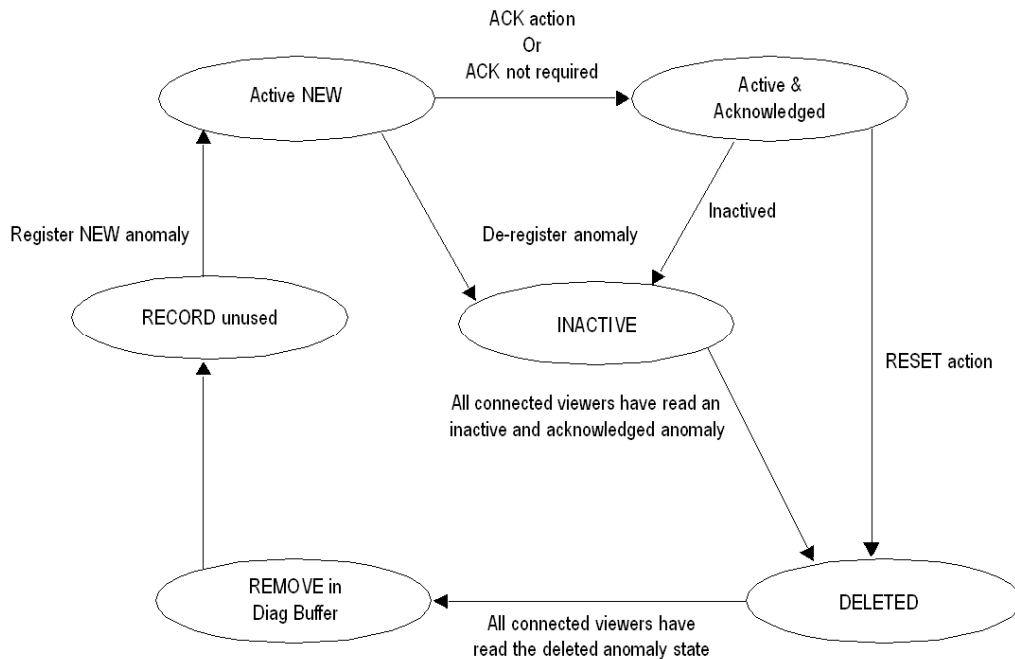
Installation of the Diag Buffer

General

Before starting up an OPC client, it is advisable to create aliases for each of the PLCs to be monitored. In order to make the installation of the **Diag Buffer** easier.

With these aliases it is easier to declare PLC addresses during the creation of an OPC client.

When an OPC client wishes to use the **Diag Buffer**, it must define a handle and use this handle only once in the creation of a group.

To do this, at each call of the **IOPCServer::AddGroup( )** method, the **ClientGroup** parameter (4th parameter) must contain a unique value. This value corresponds to the client's **clientHandle**.

As this value must also be unique amongst all the OPC clients using the **Diag Buffer**, the following procedure must be considered:

- if during the connection, the return code **OFS_E_DIAG_MMI_ALREADY_CONNECTED** is transmitted, it means that the**clientHandle** is already in use. Another value must therefore be used.
  In order to do this, consult the window which can be accessed via the **General->NetManX-WayWindow** menu and extend the branch `Devices<>@Device<>DiagBuffer` connections which gives the list of connected viewers (handle + MMI id).

Possible values for the **clientHandle** are between 0 and 2^{32} - 2 (0 to FFFFFFFE hex). FFFFFFFF hex is reserved.

Example of the settings of the handle with the C++ test client supplied on the OPC Factory Server DVD:

- create a short cut on the executable file OFSClient.exe,
- in the properties of the short cut, add onto the end of the line
  "Target"="C:\ ...\OFSClient.exe" -h10 for example to fix a handle = 10 for this OPC client.

All the examples on the following pages use the test client supplied on the DVD.

For more information on the OPC client see the OFS Client section.

Procedure for Installing the Diag Buffer

As a general rule you need to create two groups per OPC client and then observe the following sequence:

- create an inactive group
- add the specific items (`#DiagLogon`, `#DiagLogout`, `#DiagAckError`, `#DiagReadStatusU`, `#DiagResetError`, `#DiagGetFltCse`, `#DiagFltCseResult`, `#DiadRetriggError`)
- connect to the zone to be monitored (use of `#DiagLogon`)
- create an active group
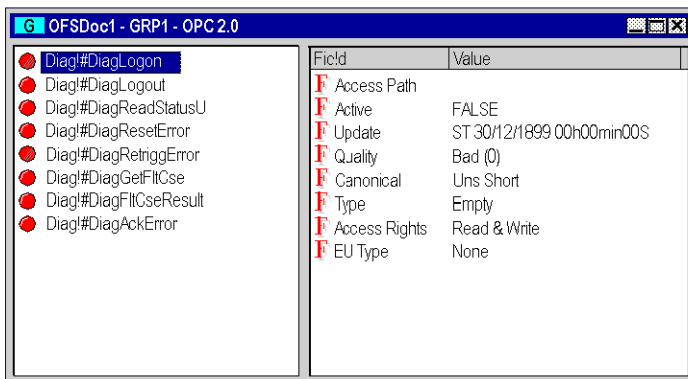- add the item `#DiagReadNextErrorU`

- **an inactive group**:

1- To connect to the **Diag Buffer**, the OPC client must add the #DiagLogon *(see page 295)* specific item to the group. The connection is established when the OPC client writes and validates the zone number of the PLC to be monitored in this item. If the write is successful, the client obtains its **viewer identifier** number by carrying out a read (1 if it is the first connected)

2- To disconnect from the **Diag Buffer**, the OPC client must add the #DiagLogout *(see page 296)* specific item to the group. The disconnection is carried out when the client writes any value in this item.

3- For acknowledgment, the OPC client requires the #DiagAckError *(see page 297)* specific item in the group.

4- To update the **Diag Buffer** status, the OPC client needs to add the #DiagReadStatusU *(see page 298)* specific item.

5- To get the **FaultCause** system message, the OPC client requires the #DiagGetFltCse (to write) and the # DiagFltCseResult *(see page 300)* (to read).

6- To retrigger the list of **FaultCause**, the OPC client requires the #DiagRetriggError *(see page 302)*specific item to the group. The retrigg is done when the client writes a valid ErrorID in this item.

7- For the deletion, the OPC client requires the #DiagResetError *(see page 299)* specific item to the group. The destruction is done when the client writes a valid ErrorID in this item.

The following screen shows the installation of specific items:

● **An active group**: this group must be created or activated after actual connection with the `#DiagLogon` item.
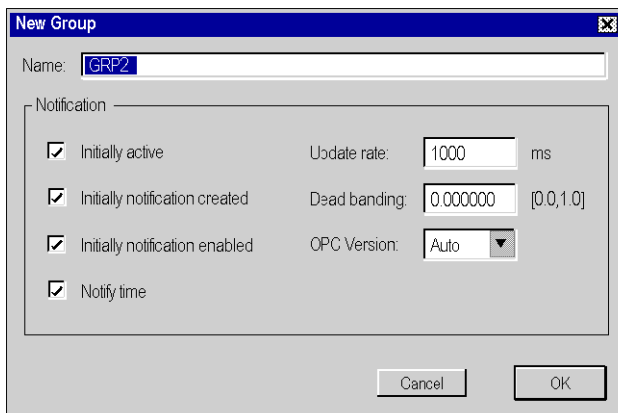
To reset the alarms coming from **Diag Buffer**, the client needs to add the #DiagReadNex-tErrorU *(see page 297)* specific item to the group.

The screen below shows a 550 bytes table *(see page 310)* in which the detected error *(see page 310)* code of the activated alarm can be found. Each byte represents a specific piece of information:

Example of translation, the bytes 2 and 3 represent the identifier of the anomaly.To acknowledge it, the client writes 0604 hex (decimal value is 1540) in #DiagAckErr.

Diag buffer table formats

Description

The 550 byte table *(see page 306)* (alarm reset after a read on #DiagReadNextError) is structured in the following way:

Illustration of the structure of the bytes table:
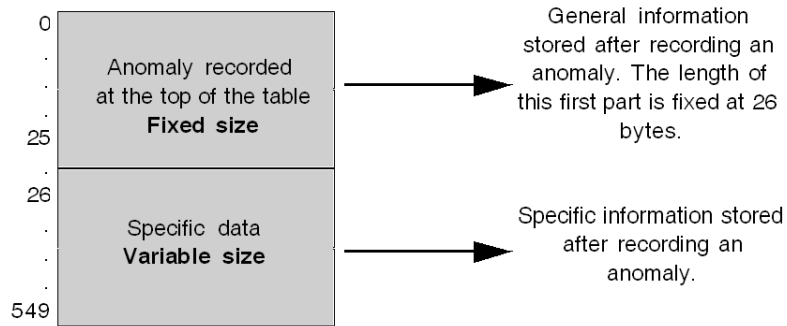
Information Retrieved by the Diag Buffer at the Top of the Table

Description

The illustration details the contents of the first 26 bytes in the table:

Definition of the Contents of the Table

- Internal ID (coded on 2 bytes): a number used inside PLC,
- Identification number (coded on 2 bytes): an identification number which is given for acknowledgment, delete or `GetFaultCause`,
- Class (coded on 1 byte): determines the class of the anomaly.

The table gives the definition of the code retrieved in this byte:

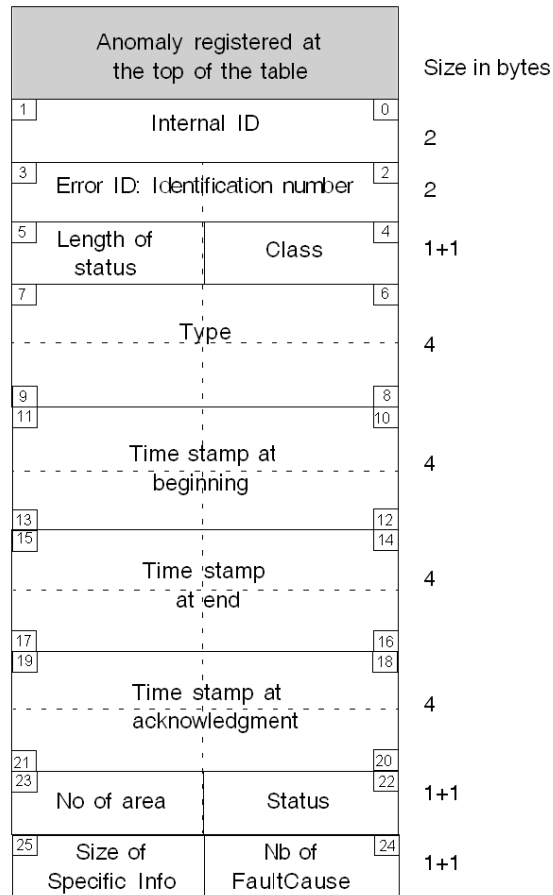| Symbol	Value (hex)	Comment
OFS_DIAGU_CLASS_SFC_MIN_TIME	00	A SFC step exited too early
OFS_DIAGU_CLASS_SFC_MAX_TIME	01	A SFC step exited too late
OFS_DIAGU_CLASS_SFC_SNS_SYNC	02	Simultaneous synchronization error
OFS_DIAGU_CLASS_FB_GEN_FB	40	Generic DFB
OFS_DIAGU_CLASS_FB_STD_EFB	64	Standard diagnostic EFB
OFS_DIAGU_CLASS_FB_USR_EFB	65	User diagnostic EFB
OFS_DIAGU_CLASS_FB_STD_DFB	66	Standard diagnostic DFB
OFS_DIAGU_CLASS_FB_USR_DFB	67	User diagnostic DFB
OFS_DIAGU_CLASS_FB_EXT	68	Extended EF/EFB/DFB
OFS_DIAGU_CLASS_FB_IO	69	IO detected errors reported
OFS_DIAGU_CLASS_GEN_SYS	80	System generic detected error
OFS_DIAGU_CLASS_SYS_CMN	85	Common system IO detected error
OFS_DIAGU_CLASS_SYS_LIO	86	Local IO detected error (not used)
OFS_DIAGU_CLASS_SYS_RIO	87	Remote IO detected error (not used)
OFS_DIAGU_CLASS_SYS_CPU	88	CPU detected error
OFS_DIAGU_CLASS_SYS_IO	89	I/O detected error
OFS_DIAGU_CLASS_SYS_ARITH	94	Arithmetic detected error
OFS_DIAGU_CLASS_SYS_TSK	95	Task detected error
OFS_DIAGU_CLASS_SYS_DGB_FULL	96	Diagnostic buffer full
OFS_DIAGU_CLASS_SYS_FFB	A8	Extended detected error

- Length of status (one byte): size of status if available,
- Type (coded on 4 bytes): copy of the status or activity time for SFC anomalies,
- Time stamp at the beginning of the alarm (coded on 4 bytes): PLC time and date when the alarm was triggered,

● Time stamp at the end of the alarm (coded on 4 bytes): PLC time and date when the alarm disappeared,
● Time stamp at the acknowledgement of the alarm (coded on 4 bytes): PLC time and date when the alarm acknowledgment,

Time stamp format:

Field	Comment	Bits	Value	no. of bits
Sec	seconds	0...5	0...59	6
Min	minutes	6...11	0...59	6
Hour	hours	12...16	0...23	5
Day	days	17...21	1...31	5
Mon	month (January = 1)	22...25	1...12	4
Year	current year - 1997(2001 = 4)	26...31	0...63	6

● Alarm status (alarm): the instantaneous status of the current alarm,
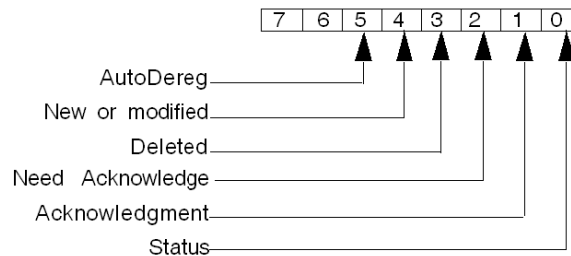
○ bit 0: Status:
   0: disappeared,
   1: active.
○ bit 1: Acknowledgment:
   0: acknowledged,
   1: not acknowledged or acknowledgment not requested.
○ bit 2: Need acknowledge:
   0: acknowledgment is not required,
   1: acknowledgment is required.
○ bit 3: Deleted:
   0: still in DiagBuffer,
   1: deleted from DiagBuffer.
○ bit 4: New or modified:
   0: modified alarm,
   1: new alarm.
○ bit 5: AutoDereg:
   1: detected error has been simultaneously activated and desactivated,

- No of Area: PLC area from where the diag buffer retrieved the anomaly,
- Nb of FaultCause: number of causes available for this alarm,
- Size of Specific Info: number of byte for specific data after this first part.

Specific information sent back by the Diag buffer in the table

Specific Data Types

There are three types of specific data:

- SFC specific data
- FB specific data
- System specific data

SFC Specific Data

The diagram describes the Variable Size Specific data section for SFC:

Specific data for SFC	Size in byte
Length of comment (bytes) + comment	1 + variable
Length of step name (bytes) + step name	1 + variable
Length of transition name (bytes) + transition name	1 + variable
Transition number	1
Reference time in ms	4

Definition of the Contents of the Table

- Length of comments + comments:
  The first part gives the length of the comments and then the comment itself.
- Length of the step name + step name:
  The first part gives the length of the name and then the step name.
- Length of transition name + transition name:
  The first part gives the length of the name and then the transition name.
- Transition number: Internal ID of the transition.
- Reference time in millisecond: configured time for this transition.

FB Specific Data

The diagram describes the Variable Size Specific data section for Function Block:

Specific data for FB	Size in byte
Length of comment (bytes) + comment	1 + variable
Length of instance name (bytes) + instance name	1 + variable
Length of FB type name (bytes) + FB type name	1 + variable
Length of PIN name in error (bytes) + PIN name in error	1 + variable
Status adress	6
Extra data	...

Definition of the Contents of the Table

- Length of comments + comments:
  The first part gives the length of the comments and then the FB comment.
- Length of the instance name + instance name:
  The first part gives the length of the name and then the instance name.
- Length of FB type name + type name:
  The first part gives the length of the name and then the FB type name.
- Length of PIN name in error + PIN name in error:
  The first part gives the length of the name and then the name of the PIN in error.
- Status address: an 6 bytes structures for Status address.
- Extra data: not documented part data.

System Specific Data

The diagram describes the Variable Size Specific data section for System:

Definition of the Contents of the Table

- Length of comments + comments:
  The first part gives the length of the comments and then the FB comment.
- Length of the instance name + instance name:
  The first part gives the length of the name and then the instance name.
- Extra data: not documented part data.

Section 18.3
Diag Buffer for PL7

Aim of this Section

This section deals with the installation of the Diag buffer and its main features. The Diag Buffer is only available on PL7-dedicated Premium PLCs.

What Is in This Section?

This section contains the following topics:

Operation from an OPC client

Reminder about the Diag Buffer

The Diag buffer *(see page 290)* is a feature which detects anomalies on monitored elements and transmits messages to the visualization system (known as the viewer).

These messages are stored in the PLC's buffer memory.

NOTE: The implementation of the diagnostics DFBs in the PLC is necessary for **Diag Buffer** functioning.

Description of the Client Interface

Diag Buffer functions authorize access to PLCs using specific items.

The table shows the specific items:

Service	Item	Type	Access	Value read	Value to write
Open connection	#DiagLogon	VT_UI2	READ/WRITE	Viewer or FFFF hex identifier	zone number
Close connection	#DiagLogout	VT_UI2	READ/WRITE	Viewer or FFFF hex identifier	not important
Read next detected error	#DiagReadNextError	VT_UI1+VT_ARRAY	READ	Error	
Error acknowledgment	#DiagAckError	VT_UI2	WRITE		Detected error ID number see *Information Retrieved by the Diag Buffer at the Top of the Table, page 333*
Evolution Status	#DiagReadStatus	VT_UI4	READ/WRITE	Status0 + Status1	Status handle

Type corresponds to OPC standards:

- VT = variable,
- UI1 = unsigned integer on 1 byte,
- UI2 = unsigned integer on 2 bytes,
- UI4 = unsigned integer on 4 bytes,
- ARRAY = table of bytes

#DiagLogon Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	R/W	no	

This item enables connection to the PLC. Firstly, the number of the zone that you wish to monitor must be indicated on the PLC (between 0 and 15) by carrying out a WRITE function.

Example of a write on #DiagLogon:

Value to write:

- bit i = 1: the zone is displayed,
- bit i = 0: the zone cannot be displayed.
  Bit 0 corresponds to zone 0, bit 15 corresponds to zone 15.
  Examples:
  - to monitor zone 6: write 0040 hex
  - to monitor zones 2 and 15: write 8004 hex

Value returned after read:

- the viewer number is displayed if the connection is open, if not the connection is not established and FFFF hex is returned.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_MMI_ALREADY_CONNECTED	The viewer is already connected

HRESULT	Comment
OFS_E_DIAG_BUFFER_FULL	Diag buffer is full
OFS_E_DIAG_TOO_MUCH_MMI	All possible viewers (15) are connected

NOTE: To monitor all zones, write FFFF hex or 0 in `#DiagLogon`.

`#DiagLogout` Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	R/W	no	

This item allows PLC disconnection.

Value to write:

● not important,

Value returned after read:

● if the disconnection is successful, FFFF hex is returned, if not, the viewer number is returned.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_BUFFER_FULL	Diag buffer is full
OFS_E_DIAG_WRONG_MMI_ID	The viewer identifier is not valid (outside range 1 to 15)
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected

NOTE: Destruction of the `#DiagLogon` item will automatically disconnect you from the viewer, without using the `#DiagLogout` item.

#DiagReadNextError Specific Item

Type	Access	Can be activated	Limitation
VT_UI1 + VT_ARRAY	R	yes	

This items enables you to read errors in the **Diag Buffer** memory.

Value to write:

- nothing,

Value returned after read:

- errors saved in the form of a 120 byte table *(see page 332)*.

Value returned by the item:

HRESULT	Comment
S_OK	Read successful, no modification is recorded in the 120 byte table
S_OK	Read successful, modifications recorded in the 120 byte table (the anomaly has been acknowledged or has disappeared)
S_OK	Read successful, a new table has been created (a new anomaly has appeared)
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_BUFFER_FULL	Diag buffer is full
OFS_E_DIAG_WRONG_MMI_ID	The viewer identifier is not valid (outside range 1 to 15)
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected

#DiagAckError Specific Item

Type	Access	Can be activated	Limitation
VT_UI2	W	no	

This item allows alarm acknowledgment.

Value to write:

- the value on 2 bytes corresponding to the **Identification number** zone starting by a read of the highest ranking bit (the first two bytes in the table).
  E.g. the value returned in the **Identification number** zone of the #DiagReadNextError item table is such that: Var[0] = 04 hex, Var[1] = 05 hex. The value to write in the item #DiagAckError is 0504 hex.

Value returned after read:

- nothing.

Value returned by the item:

HRESULT	Comment
OFS_E_DIAG_OK	OK
OFS_E_DIAG_NO_BUFFER	Diag buffer not activated
OFS_E_DIAG_BUFFER_FULL	Diag buffer is full
OFS_E_DIAG_MMI_NOT_CONNECTED	OPC client not connected
OFS_E_DIAG_WRONG_ERROR_ID	Non authorized anomaly identifier
OFS_E_DIAG_ERROR_NOT_USED	No element corresponds to this identifier

#DiagReadStatus Specific Item

Type	Access	Can be activated	Limitation
VT_UI4	R/W	no	

This item lets you to know the evolution of the status of a DFB anomaly without having to wait to be notified of a change in the 120 bytes error table *(see page 332)*.

Value to write:

- the value on 4 bytes corresponding to the **Status Handle** zone starting by a read of the highest ranking byte.
  E.g. the value returned in the **Status Handle** zone of the #DiagReadNextError item table is such that:
  Var[8] = 98 hex, Var[9] = 01 hex, Var[10] = 76 hex, Var[11] = 25 hex
  The value to write in the #DiagReadStatus item is 25760198 hex or 628490648 dec.

Value returned after read:

- the values of status 0 + status 1, taking into account the value of the words from right to left.
  E.g. the value returned is 0010001D hex; status0 value is 001D hex; status1 value is 0010 hex.

Description of Client Operation

The diagram shows how an OPC client operates using specific items:
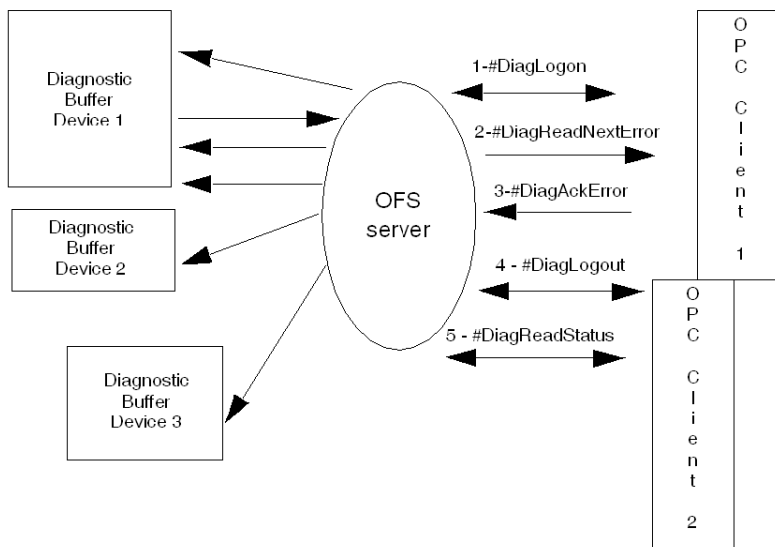
With the OFS server several PLCs can be monitored at the same time, it has a multi-station function (unlike the PL7 which can only manage one PLC at a time). To supervise several PLCs at once, simply create other aliases in the configuration tool and add them to another group belonging to the same client (minimum of 1 group per device to monitor).

Diag Buffer Management

Anomalies recorded in the diag buffer memory can have the following statuses:

- active or inactive,
- acknowledgment requested or acknowledgment not requested,
- if acknowledgment is requested, the error may can be acknowledged or not acknowledged.

NOTE: Only anomalies from the diag buffer can be acknowledged. An anomaly displayed on several viewers is deleted from all viewers once it has been acknowledged on one viewer.
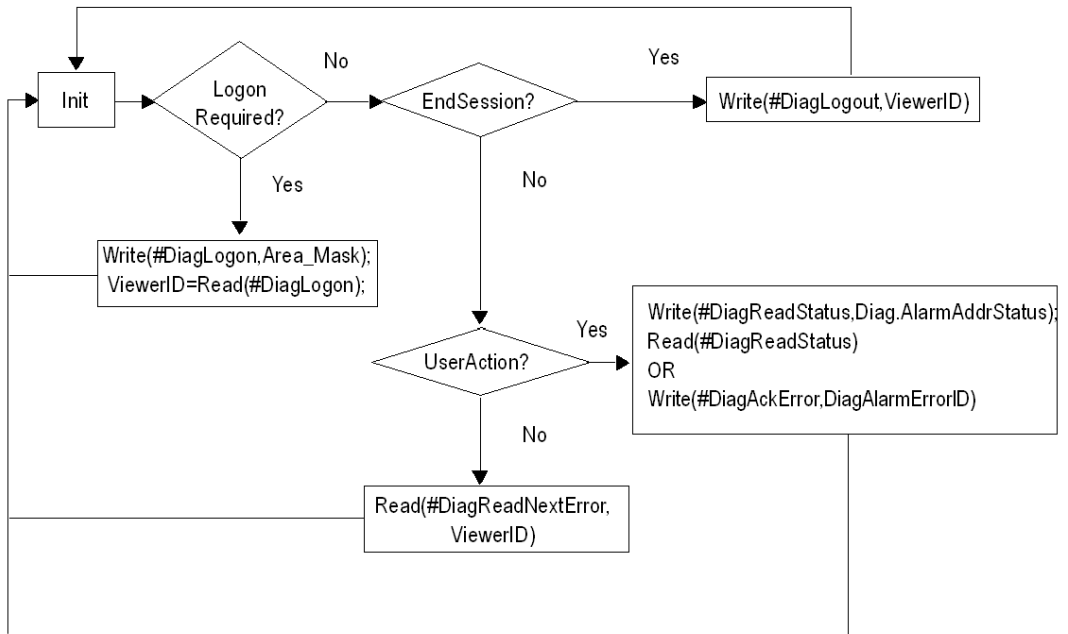An alarm is deleted from the buffer if:

- the alarm no longer exists,
- all viewers have read the alarm,
- the alarm has been acknowledged (after an acknowledgment request).

Description of client sequencing

Description of client sequencing

The graph below shows how and in which order an OPC client uses the specific items for PL7 Diag buffer:

Life cycle of a Diag buffer alarm in a PLC

The graph below shows the life cycle of an alarm inside the Diag Buffer of PL7 PLC. We can see that a client which does not acknowledge the alarm or which read very slowly the alarm can force the PLC to maintain the alarm inside the Diag Buffer (with risk of overflowing), until the alarm is read (by each connected client) and aknowledged (if necessary). We can also see that an alarm is no longer available if its state is inactive and it has been read. So we can no more call **#DiagReadStatus** on this alarm, for example.
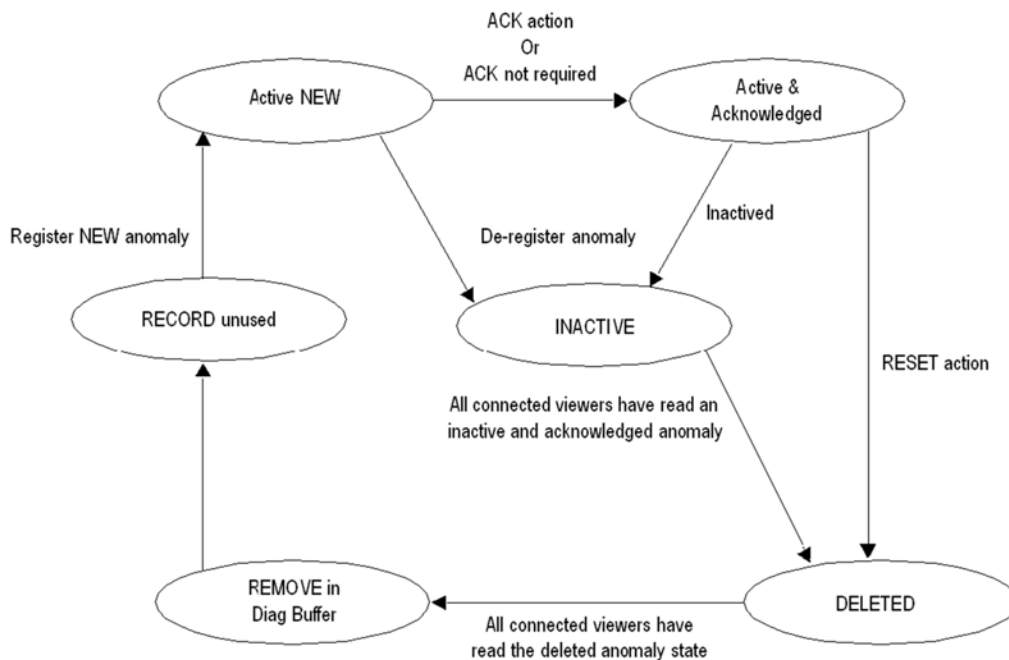
Installation of the Diag buffer

General

Before starting up an OPC client, it is advisable to create aliases for each of the PLCs to be monitored. In order to make the installation of the diag buffer easier.

With these aliases it is easier to declare PLC addresses during the creation of an OPC client.

When an OPC client wishes to use the diag buffer, it must define a handle and use this handle only once in the creation of a group.

To do this, at each call of the IOPCServer::AddGroup( ) method, the hClientGroup parameter (4th parameter) must contain a unique value. This value corresponds to the client's `clientHandle`.

As this value must also be unique amongst all the OPC clients using the diag buffer, the following procedure must be considered:

- if during the connection, the return code **OFS_E_DIAG_MMI_ALREADY_CONNECTED** is transmitted, it means that the `clientHandle` is already in use. Another value must therefore be used.
  In order to do this, consult the window which can be accessed via the General->NetManX-WayWindow menu and extend the branch Devices<> @Device<>DiagBuffer connections which gives the list of connected viewers (handle + MMI id).

Possible values for the `clientHandle` are between 0 and 2^{32} - 2 (0 to FFFFFFFE hex). FFFFFFFF hex is reserved.

Example of the settings of the handle with the C++ test client supplied on the OPC Factory Server DVD:

- create a short cut on the executable file OFSClient.exe,
- in the properties of the short cut, add onto the end of the line
  "Target"="C:\ ...\OFSClient.exe" -h10 for example to fix a handle = 10 for this OPC client.

All the examples on the following pages use the test client supplied on the DVD.

For more information on the OPC client see the OFS Client section.

Procedure for Installing the Diag Buffer

As a general rule you need to create two groups per OPC client and then observe the following sequence:

- create an inactive group
- add the specific items (`#DiagLogon`, `#DiagLogout`, `#DiagAckError`, `#DiagReadStatus`)
- connect to the zone to be monitored (use of `#DiagLogon`)
- create an active group
- add the item `#DiagReadNextError`

● **an inactive group**:
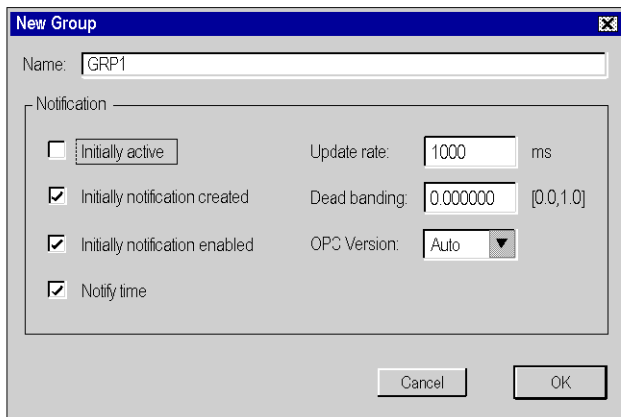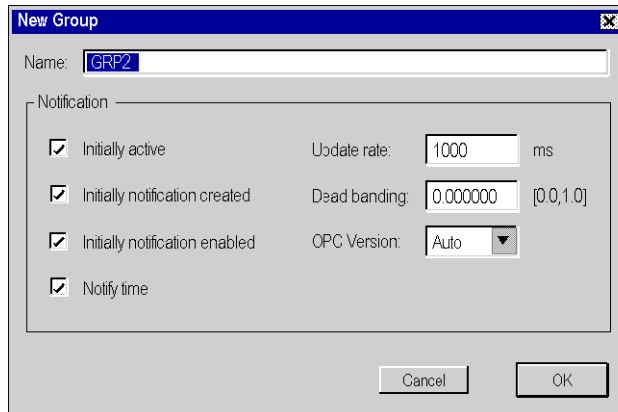
1- To connect to the diag buffer, the OPC client must add the #DiagLogon *(see page 320)* specific item to the group. The connection is established when the OPC client writes and validates the zone number of the PLC to be monitored in this item. If the write is successful, the client obtains its "viewer identifier" number by carrying out a read (1 if it is the first connected)

2- To disconnect from the diag buffer, the OPC client must add the #DiagLogout *(see page 321)* specific item to the group. The disconnection is carried out when the client writes any value in this item.

3- For acknowledgment, the OPC client requires the #DiagAckError *(see page 322)* specific item in the group.

4- To update the Diag Buffer status, the OPC client needs to add the #DiagReadStatus *(see page 323)* specific item.

The following screen shows the installation of specific items:
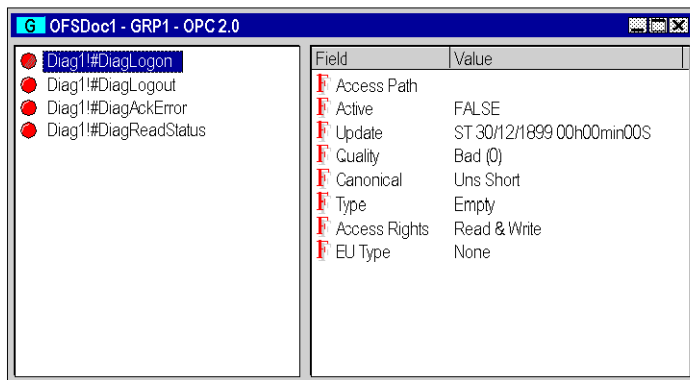
- **An active group**: this group must be created or activated after actual connection with the `#DiagLogon` item.

To reset the alarms coming from diag buffer, the client needs to add the #DiagReadNextError *(see page 322)* specific item to the group.

The screen below shows a 120 byte table *(see page 332)* in which the detected error *(see page 332)* code of the activated alarm can be found. Each byte represents a specific piece of information:
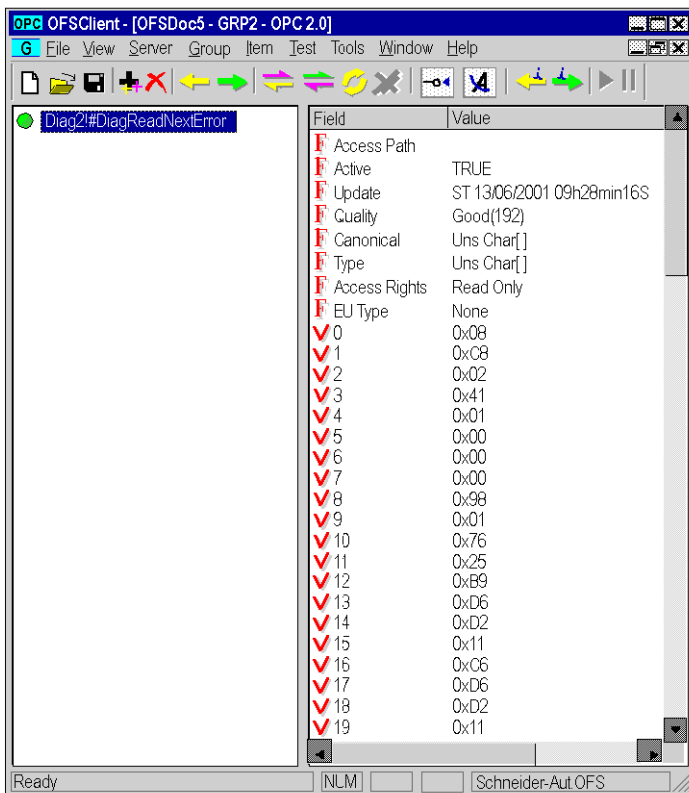
Example of translation of bytes V12 to V15 which represent the time of the beginning of the alarm. The table describing the time stamp format of the Diag buffer enables you to extract the different values.

The values read are: V15=11 hex, V14=D2 hex, V13=D6 hex, V12=B9 hex.

	V15		V14		V13		V12	
Hexadecimal	1	1	D	2	D	6	B	9
Binary	0001	0001	1101	0010	1101	0110	1011	1001
Decoding	4		7	9	13		26	57

Date	Years	Months	Days	Hours	Minutes	Seconds

Calculation of the year: 4 + 1997 = 2001

The result is therefore 13 h26 min57 s, 9/07/2001.

Diag buffer table formats

Description

The 120 byte table *(see page 327)* (alarm reset after a read on #DiagReadNextError) is structured in the following way:

Illustration of the structure of the bytes table:
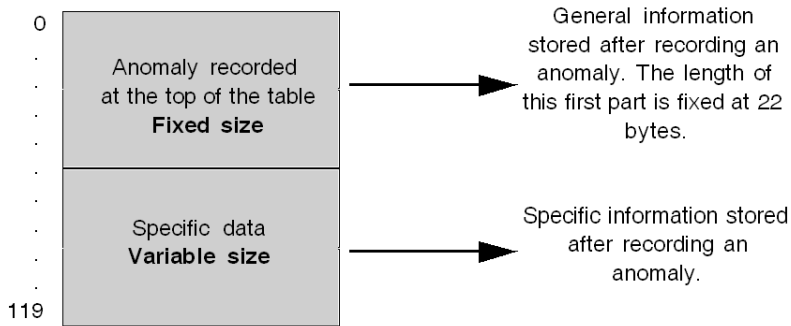
Information Retrieved by the Diag Buffer at the Top of the Table

Description

The illustration details the contents of the first 22 bytes in the table:
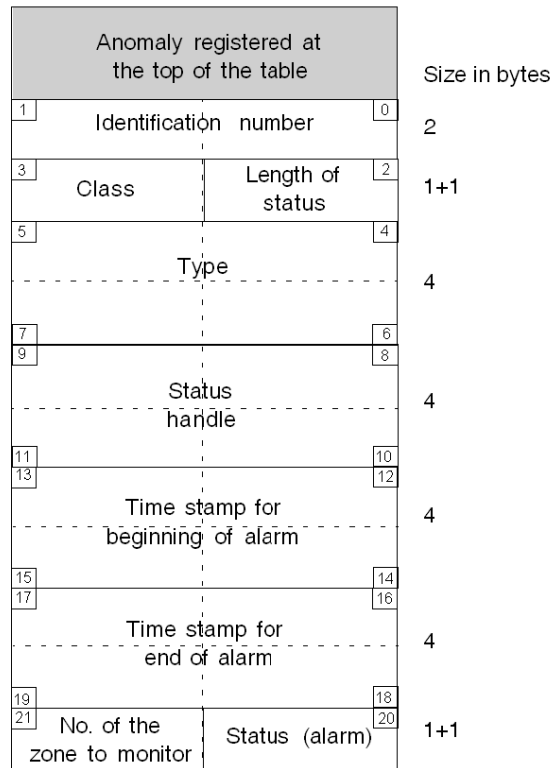
Definition of the Contents of the Table

- Identification number (coded on 2 bytes): an identifying number which is given for acknowledgment. This number must be written in the `#DiagAckError` item to acknowledge an alarm,
- Length of status (coded on 1 byte): depends on the DFB which has been programmed. If the value is 2, it is "status 0" which means that in the "type", the value status 0 will be read. If the value is 4, it is "status 0 & status 1" which means that in the "type" status 0 & status 1 will be read.
- Class (coded on 1 byte): determines the class of the anomaly.

The table gives the definition of the code retrieved in this byte:

Symbol	Value (hex)	Comment
OFS_DIAG_CLASS_DFB_EV_DIA	40	EV_DIA detected error
OFS_DIAG_CLASS_DFB_MV_DIA	41	MV_DIA detected error
OFS_DIAG_CLASS_DFB_NEPO_DIA	42	NEPO_DIA detected error
OFS_DIAG_CLASS_DFB_ALARM	43	ALRM detected error
OFS_DIAG_CLASS_DFB_USERA	4 A	USER DFB detected error
OFS_DIAG_CLASS_DFB_USERB	4B	USER DFB detected error
OFS_DIAG_CLASS_DFB_USERC	4C	USER DFB detected error
OFS_DIAG_CLASS_DFB_USERD	4D	USER DFB detected error
OFS_DIAG_CLASS_DFB_USERE	4E	USER DFB detected error
OFS_DIAG_CLASS_DFB_USERF	4F	USER DFB detected error
System detected error class		
OFS_DIAG_CLASS_DFB_SYSTEM_ASI0	80	STGENE of ASI_DIA detected error
OFS_DIAG_CLASS_DFB_SYSTEM_ASI1	81	STSLABS of ASI_DIA detected error
OFS_DIAG_CLASS_DFB_SYSTEM_ASI2	82	STSLKO of ASI_DIA detected error
OFS_DIAG_CLASS_DFB_SYSTEM_ASI3	83	STSLNC of ASI_DIA detected error
OFS_DIAG_CLASS_DFB_SYSTEM_IO	84	IO_DIA detected error
New characteristic of the PL7v4		
OFS_DIAG_CLASS_DIAGSYSTEM	85	system detected error (Task, Arithm)
OFS_DIAG_CLASS_SYT_LOCALIO	86	LOCAL IO detected error
OFS_DIAG_CLASS_SYT_REMOTIO	87	REMOTE IO detected error
OFS_DIAG_CLASS_SYT_BUFFERFULL	88	Diag Buffer full

- Type (coded on 4 bytes): this is the type of anomaly which is retrieved by the diag buffer:
  - Diag-DFB: status value, coding on 2 bytes for "length of status" = 2, 4 bytes for "length of status" =4.
  - Grafcet: system detected anomaly. It occurs when the execution time exceeds the expected time.
  For more information see PL7 documentation on DFBs.
- Status handle (coded on 4 bytes): this value must be used during a `#DiagReadStatus write`,
- Time stamp at the beginning of the alarm (coded on 4 bytes): time and date when the alarm was triggered,
- Time stamp at the end of the alarm (coded on 4 bytes): time and date when the alarm disappeared,

Time stamp format:

Field	Comment	Bits	Value	no. of bits
Sec	seconds	0...5	0...59	6
Min	minutes	6...11	0...59	6
Hour	hours	12...16	0...23	5
Day	days	17...21	1...31	5
Mon	month (January = 1)	22...25	1...12	4
Year	current year - 1997(2001 = 4)	26...31	0...63	6

● Status (alarm): the instantaneous status of the current alarm,

  ❍ bit 0: Status:
    0: disappeared,
    1: active.
  ❍ bit 1: Acknowledgment:
    0: acknowledged,
    1: not acknowledged or acknowledgment not requested.
  ❍ bit 2: Type of alarm (with or without acknowledgment):
    0: acknowledgment not requested,
    1: acknowledgment requested.
● Number of the zone to be monitored: PLC zone from where the diag buffer retrieved the anomaly. Grafcet anomalies always belong to the common zone.

Specific information sent back by the Diag buffer in the table

Specific data types

There are two types of specific data:

- DFB specific data,
- "other" specific data.

Diag buffer specific data

The diagram below describes the Variable Size Specific data section for classes between OFS_DIAG_CLASS_DFB_EV_DIA and OFS_DIAG_CLASS_DFB_SYSTEM_IO (see *Definition of the Contents of the Table, page 333*):

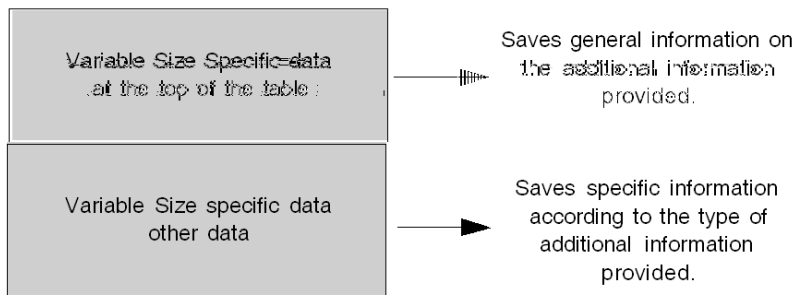| Specific DFB	Size in bytes
Length of comment (bytes) + comment	1 + variable
Length of "instantiated" name (bytes) + "instantiated" name	1 + variable
Length of the type of DFB (bytes) + type of DFB	1 + variable
Length of the program address + program address	1 + variable

Definition of the contents of the table

- Length of comments + comments:
  The first part gives the length of the comments and then the DFB message.
- Length of the "instantiated" name + "instantiated" name:
  The first part gives the length of the "instantiated" name then the DFB "instantiated" name.
- Length of file name + file name:
  The first part gives the length of the file name then the file name.
- Length of the program address + program address:
  The first part gives the program address length then the program address which corresponds to a DFB execution anomaly.

"Other" specific data

The diagram below describes the Variable Size Specific data section for classes between OFS_DIAG_CLASS_DIAGSYSTEM and OFS_DIAG_CLASS_SYST_BUFFERFULL (see *Definition of the Contents of the Table, page 333*).

Specific data gives more information according to the class recorded.

Illustration:

Variable Size Specific data at the top of the table

The diagram below gives the structure of Variable Size Specific data at the top of the table:

- Length of comments + comments:
  The first part gives the length of the comments and then the diagnostics DFB message.
- Length of the "instantiated" name + "instantiated" name:
  The first part gives the length of the "instantiated" name then the "instantiated" name of the diagnostic anomaly.
- Size information:
  The content gives the size of the buffer's additional information.
- Type:
  The content gives the type of additional information of specific data.

Chapter 19
Communication

Aim of this Chapter

The aim of this chapter is to describe the product's communication methods.

What Is in This Chapter?

This chapter contains the following sections:

Section 19.1
Communication

Aim of this Section

The aim of this section is to describe the type of communication used by the OFS server.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction	341
X-Way addressing modes	343
Direct addressing modes	347

Introduction

General

- the OFS server allows the use of several different communication media simultaneously: a client application can, for example, gain access to a PLC using Fipway and to another using ISAway.
- the OFS server provides X-Way and Modbus network transparency: A client application can access PLCs in a PLC network architecture including bridges for switching between communication media.

- OFS server behavior in the event of an inoperable communication with the PLC (PLC missing, disconnected, etc.):
  - all the requests corresponding to one group will be transmitted, both for reading items and for writing items.
  - from a performance point of view, this means that the execution period for the read or write primitive may rise to n times the timeout period (where n is the number of requests associated with the group).

Note:
There are no request retries on timeout.

NOTE: For networks with logical connections, if the connection is broken, the server automatically tries to re-establish it.
E.g.: TCP-IP.
When the XIP driver is used more than one device connected and one of these is absent, communication with devices connected by XIP is blocked for a few seconds as XIP uses Winsock and awaits the end of the TCP/IP timeout. After this timeout, everything should come back normal except, of course, communication with the missing PLC.

The OFS server will indicate communication anomalies to the client application in the following way: each item belonging to an inoperative request will be marked `invalid` *, whether it is for a group's synchronous or cyclic read request.

* Whatever the method used to perform the read, `invalid` means that the Quality attribute is `Bad`. `Valid` means that the Quality attribute is `Good`.

Comments:

- the client application can determine whether the PLC has been reconnected by re-addressing a synchronous read request to the group concerned,
- during the cyclic read of a group, the quality of items (Quality attribute) will change from `Bad` to `Good` when the PLC is reconnected. The feedback mechanism *(see page 225)* describes the Quality attribute associated with an item.

The OFS server allocates for:

- X-Way: a socket (communication channel) for each driver for PL7 or UNITY-type devices, and up to 16 sockets for XTEL or ORPHEE-type devices
- Modbus Plus: a path to each device (PM), or up to 16 paths to each device (DM)
- TCP-IP (non-XIP): Up to 16 sockets to each device *(see page 348)*.
- USB: Up to 4 sockets to each device *(see page 348)*.

NOTE: Modbus Plus paths are opened and closed dynamically according to requirements. Therefore, even with only one SA85 card (8 DM paths), it is possible to dialog with more than 8 devices.

X-Way addressing modes

Description

Example of access through a network:
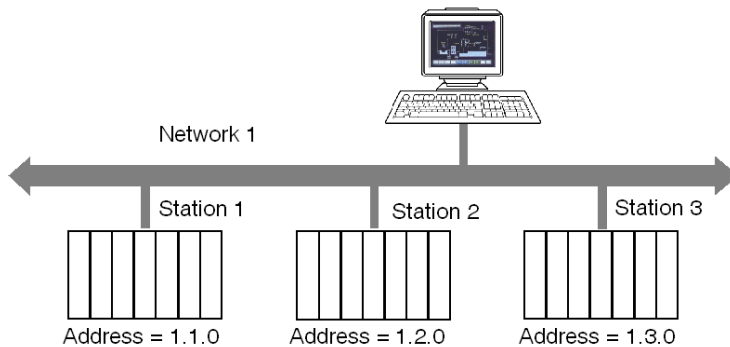
Addressing to 3 levels:

Allows a station connected to the network at any point of the X-Way communication architecture to be reached.

Illustration:

The Network and Station values make up the station address.

- Network: value between [1.127] or 0 = my network.
- Station: value between [1.63] or 254 = my station or 255 = diffusion.

The value "Gate" refers to the communication entity within the station: system server (Gate 0, the most common), the terminal port (Gates 1,2,3), 1 K asynchronous server (Gate 7), etc.

In the case of multiprocessor stations such as PLCs, each processor module built into the system can support communication entities, frame routing requiring supplementary addressing levels (inter-station routing capabilities). PLC "processor modules" are situated in the PLC's racks or offset on field buses.

Addressing to 5 levels:

It is generally used for devices connected on a Uni-Telway bus.

Illustration:

Network	Station	Gate=5	Module	Channel
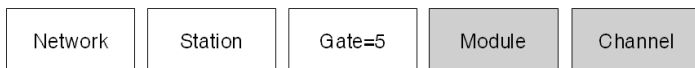
- Module: physical location of the communication module in the rack. Its value must be defined as follows: (Master rack number * 16) + Number of master module.
- Channel: address of the device connected to the communication module. Its value must be defined as follows: (Master channel number * 100) + slave Ad0 number.

Addressing to 6 levels:

This is similar to addressing to 5 levels. It was created for extended services (FIPIO, communication module integrated in rack).

Illustration:

Network	Station	Gate=8	Selector	Conn-ection point	Reference
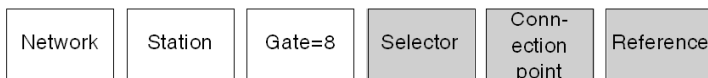
- **Selector**: designates a communication module on the CPU (2) or in a separate module (1).
- **Connection point**: device address, if the destination module is FIPIO. Physical positioning in the PLC rack, if the destination module is a PLC card.
- **Reference**: communication entity in the device (similar to the Gate number).

Examples:

5 level addressing:

Network 1

Station 1

Rack 0
Master
address = 1.1.0

Station 2

Rack 1

Module = 16*1 + 1 = 17

Master
module 1

Slave 4

Slave address = 1.1.5.17.4 (if the master is on channel 0).
Slave address = 1.1.5.17.104 (if the master is on channel 1).

6 level addressing:

Network 1,
Station 3,
Gate 8 (FIPIO),
Module 2 (CPU communication module,
Address 14,
Gate 0 (UNITE target PLC server),
→ 1.3.8.2.14.0
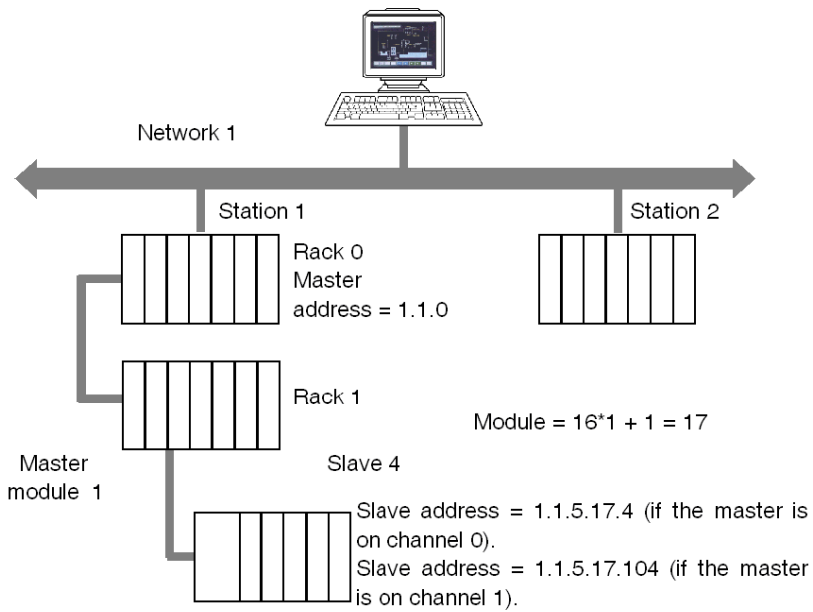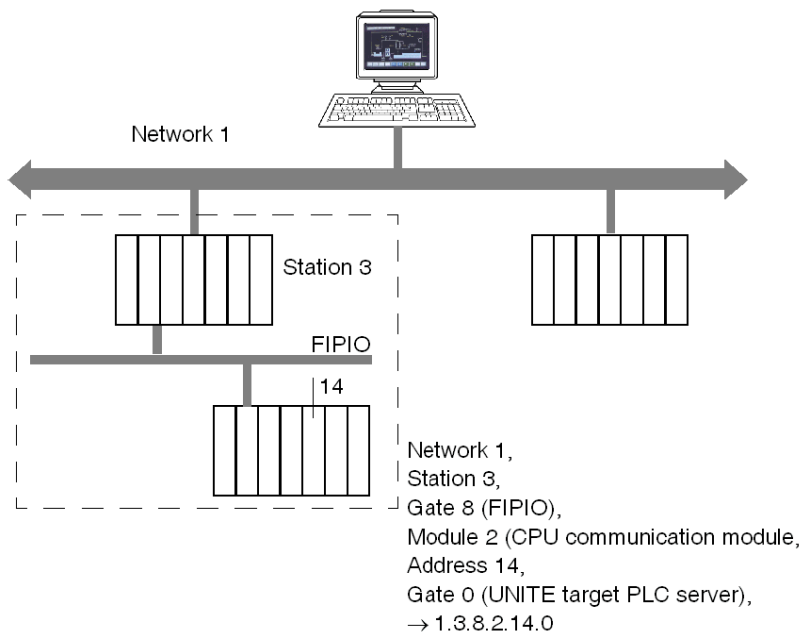
For more information on the X-Way address, see "X-Way Communication" documentation, ref. TSX DR NET.

NOTE: In point to point connections (Uni-Telway, ISAway, PCIway), the default address 0.254.0 can be used to reference the PLC.
0.254.0 can be used to access to the Fipio Master when we are connected through the privilegied terminal @63
0.254.5.17.104 can be used to access to the Uni-Telway slave @4 which is connected on the rack 1 module 1 channel 1 when we are connected on the local PLC.
0.254.8.2.14.0 can be used to access to the Fipio connection point 14 when we are connected through the privilegied terminal @63.
With Ethway and XIP, it is possible to use gate 7, which accepts large frames (up to 1024 bytes). In order to do this, the PL7 application must be configured in periodic mode (MAST task). The "1 K service" option must be checked in the alias definition page.
Example: normal address: XIP01:1.2, to use gate 7: XIP01:1.2.7

Direct addressing modes

Description

- over TCP/IP, the only information required is the IP address. This can be given as four groups of numbers separated by dots or by a DNS name e.g. "My Station". In this case, the DNS scanning feature should be enabled *(see page 125)*.
- over Modbus Plus, the syntax is:
  <access level>.<node1>.<node2>.<node3>. <node4>.<node5>
  The access level can be:
  - ○ PM = Program Master,
  - ○ DM = Data Master.
- over USB, there is no information.

The node number should be used to specify the full path. To access a device without a bridge, only access mode and node are required.

For TCP/IP - Modbus Plus bridges, the syntax is:

MBT:<IP bridge address>;<Modbus Plus device node number>

The IP bridge address corresponds to the number entered into the configuration tool's "MBP bridge index" box. This configuration is detailed in the network section of the device *(see page 80)*.

For example:

Address of node 7

PM.7 or DM.7
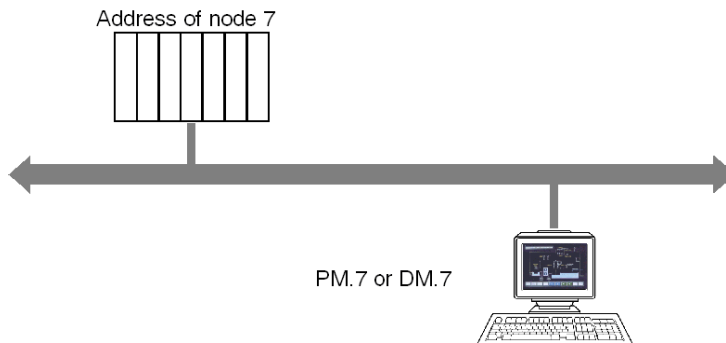
Section 19.2
Multi-Channel Feature

Multi-Channel Feature

Description

Certain communication protocols are half-duplex networks which means that after sending one request, the server should wait for the answer before sending the next request. This is the case for most protocols used by OFS except on X-Way Unity- or PL7-type PLCs. The only way to speed up communications is to open several channels between the sender and the receiver.

You may open between 1 and 16 channels for each device and you may configure that number either statically with the OFS Configuration Tool or dynamically with the specific #MaxChannel item. The value that gives optimal performance depends on the PLC being accessed (number of requests it can process per cycle) and the communication card being used (notably on Concept-type PLCs). To obtain this data, please refer to the PLC and communication card documentation.

NOTE: The multi-channel function is not significant for Unity Pro or PL7-type PLCs using an X-Way network (full-duplex protocol) or with a serial Modbus driver (single channel only) for all PLC types.

NOTE: For Uni-Telway, the maximum number of channels is linked to the number of slaves declared in the Uni-Telway driver configuration.

NOTE: The specific item #NbrMaxPendingReq cannot theoretically be greater than the number of channels open on a half-duplex protocol. If the value is greater, the requests are placed in a queue. These two parameters have a major effect on the communication performance of OFS.

NOTE: To summarize, on a half-duplex protocol, the maximum number of requests transmitted in parallel to a device is the smaller value between #NbrMaxPendingReq and the number of channels actually open.
On a full-duplex protocol, only the value #NbrMaxPendingReq is taken into account.

Section 19.3
PLC Link Redundancy

Aim of This Section

The aim of this section is to describe how to manage communication redundancy with a Unity PLC.

What Is in This Section?

This section contains the following topics:

Presentation

Description

The PLC link redundancy function is an optional feature allowing to establish a redundant link between OFS and Unity PLCs. The redundancy of communication relies on two physical communication paths using two different IP addresses attributed to the same PLC with a unique alias.

The switch from Primary communication path to the Standby communication path is insured automatically by OFS, and is done transparently from the OPC client application. There is no impact at SCADA level when switching from Primary communication path to Standby communication path mode.

This feature can be used in both stand-alone or Hot Standby (HSBY) PLC architecture supporting Modbus TCP.

NOTE: Use two Ethernet ports located in the PLC local rack (for example one CPU and one Ethernet module, or two Ethernet modules).
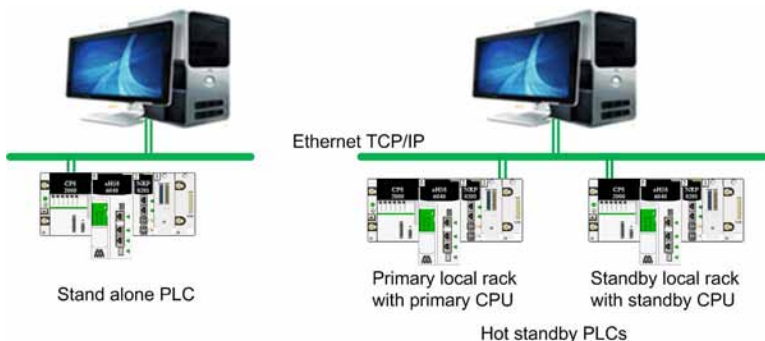
NOTE: The purpose of this feature is not to support the following architecture because in that case, the switch of IP@ is insured by the PLC itself and is therefore transparent to OFS.
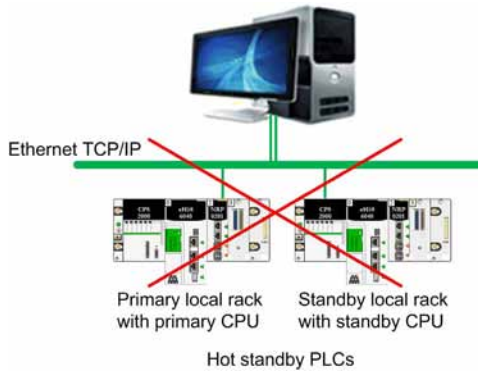
General Principle

Introduction

OFS is connected to the PLC via two physical links and two IP addresses. The two links established between OFS and the PLC are the **Primary communication path** and the **Standby communication path**. In the case that one link no longer functions, the other one takes over.

The communication parameters must be the same for both primary and secondary paths. Both paths must be either on the same network, or be routed through the same gateway and/or router.

Primary Communication Path

The first communication path that successfully allows PLC and application information retrieval is elected as the Primary communication path. The Primary is **ONLINE** and the communication is available with the OFS server.

Standby Communication Path

Once the Primary communication path is elected, two periodic checks of the Standby communication path are performed every 10 seconds:
● A Standby communication path **connection check** (communication OK/NOK).
● A Standby communication path **consistency check** (firmware version, application, request length OK/NOK).

Communication Paths Status

Initial status: the Primary communication path is supposed to be **ONLINE**.

Then, the connection check with the Standby communication path is performed:
1. If the Standby communication path is not identified (never connected), then the Standby communication path is **OFFLINE** and no switch can be operated. The Primary communication path remains **ONLINE**.
2. If the Standby communication path is identified, then the consistency check can be done:
   ○ If the Standby communication path consistency check is successful, the Standby communication path is **ONLINE**. A switch can be operated from Primary to Standby communication paths.

   ○ If the Standby communication path consistency check is unsuccessful, the system is not operational. Both Primary and Standby communication paths are **OFFLINE**. The communication becomes unusable and no switch can be operated.

Operating Modes

Switch Cases

A switch can be triggered in 2 cases:

1. When communication on the Primary communication path is not usable (send or receive error).

2. On demand, by using `#SwitchPrimaryAddress` specific item *(see page 353)*.

NOTE: A send or receive error can be a request timeout due to a device communication latency and cannot be differentiated from a communication interrupt.

In both cases, to avoid any requests being lost, the last request is sent again to the new Primary communication path. Therefore, there is no effect on client application variables when switching from Primary to Standby communication paths.

Switch Management

The switch from one communication path to the other is managed as follows:

- If Standby communication path was **ONLINE** at the last check, the switch can be operated. The Standby communication path becomes the new Primary communication path.
- If Standby communication path was **OFFLINE** due to disconnection at the last check, a switch attempt can be operated. In that case, a switch will be performed on each access, as long as a new Primary communication path is not established.
- If Standby communication path was **OFFLINE** due to inconsistency at the last check, no switch can be operated.

Due to the periodicity of the Standby communication path consistency check, some abnormal inconsistent configuration may not be detected (for example, the last Standby path consistency check returned **ONLINE** status but the Standby path has lost consistency since). Those situations should nevertheless never happen during runtime phase (that means that the network configuration changed in an inconsistent way).

⚠ **CAUTION**

INCONSISTENT CONFIGURATION NOT DETECTED

Keep the Primary communication path and Standby communication path connected to the same PLC.

Failure to follow these instructions can result in injury or equipment damage.

Specific Item on Unity Pro Devices:

The `#SwitchPrimaryAddress` specific item is supported only on Unity devices *(see page 193)*.

NOTE: When PLC link redundancy is configured, `#DisableDevice` has `ReadOnly` user rights. Disabling both Primary and Standby communication path would interrupt Primary and Standby communication path consistency check and disturb switch operating modes.

Configuration

PLC Link Redundancy

Two device address fields are available in the OFS configuration:

Device name	M580
Device address A	MBT:192.168.3.1/U
Device address B	MBT:192.168.3.2/U

NOTE:
The following rules are applied:
- The `Device address B` is enabled only if `Device address A` is:
  - `MBT:x.y.z.t/U` (MBT:<IP> and /U ≡ programming with Unity Pro) or
  - `MBT:myDNS/U` (DNS scanning TCP/IP and /U ≡ programming with Unity Pro).
- The `Device address A` and `Device address B` have the same network type.
- The `Device address B` cannot be equal to `Device address A` and cannot be configured in any other alias.

Configuration

For easier configuration, the device address wizard on `Device address B` is initialized with `Device address A`.

This figure shows the device address wizard:
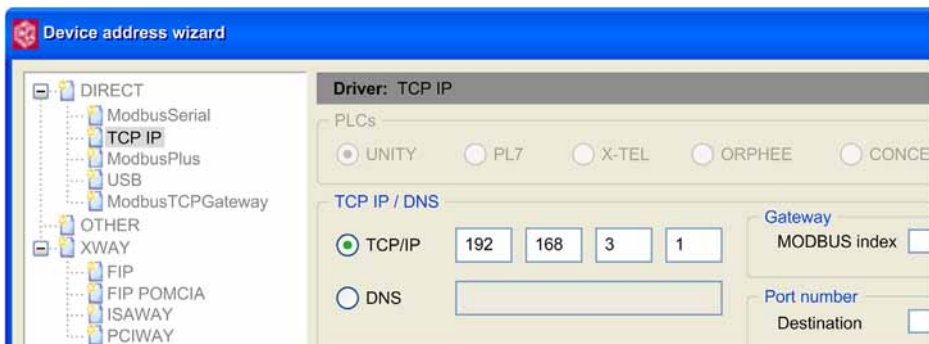
NOTE: The DNS is supported for both `Device address A` and `Device address B`.

Runtime

OFS Network Windows Information

The OFS network window displays additional information when address redundancies are configured.

This figure shows the new OFS network window:
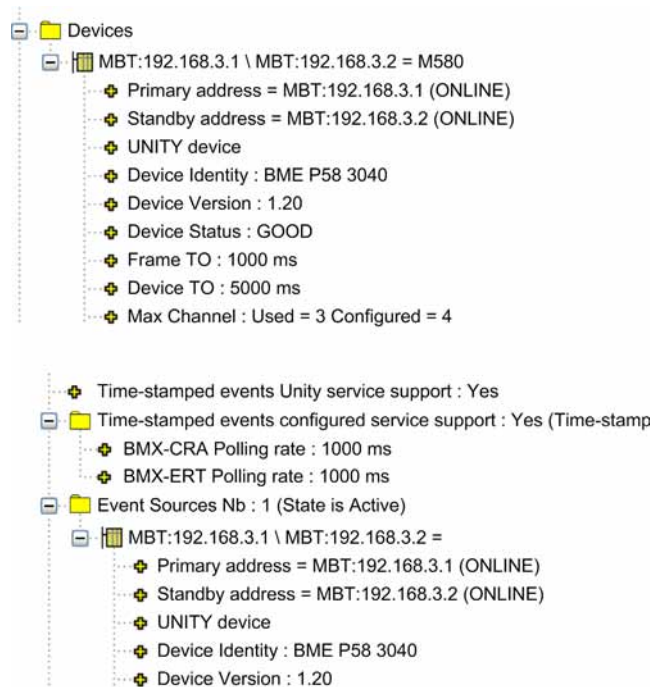
Section 19.4
Device Connection Advanced Operating Mode

Aim of This Section

The aim of this section is to describe the device connection advanced operating mode.

What Is in This Section?

This section contains the following topics:

Presentation

Description

On a device disconnection, OFS maintains the communication flow on the network in order to recover the communication as soon as possible.

Two parameters allow you to adjust the communication flow recovery:
- `Reconnection retry number`
- `Disconnection timeout`

NOTE: Those parameters are not applicable on MBTG (Modbus TCP gateway) and XWAY gateway devices, but they are applicable on devices connected behind MBTG or XWAY gateway.

Reconnection Retry Number

On device disconnection, this parameter indicates the number of reconnecting retries that are done (including any communication operation such as polling, synchronous, and asynchronous read/write operations, and so on). If the device remains disconnected after the retries, any communication operation will be ignored during the `Disconnection timeout`.

The valid range is [0...100].

Disconnection Timeout

The `Disconnection timeout` is linked with the `Reconnection retry number` parameter. The `Disconnection timeout` indicates the duration for which the communication operations are ignored. Once the timeout elapsed, if the device remains disconnected, a new cycle of connection retries is started (as defined by `Reconnection retry number`).

The valid range is [0...1440 min].

NOTICE
DEVICE COMMUNICATION IS MAINTAINED DISABLED
To keep device communication enabled, configure the value of the `Disconnection timeout` parameter carefully adapted to your system, or keep the default value.
Failure to follow these instructions can result in equipment damage.

Configuration

Enhanced Adjustment Information

A new **Enhanced adjustment information** section is available. These two information are `Reconnection retry number` (0... 100) and `Disconnection timeout` (0....1440 mn).

The figure shows the **Enhanced adjustment information** section:

Adjustment Information	
Max channels	16
Max Pending	0
Device timeout (ms)	5000
Frame timeout (ms)	1000
Enhanced adjustment information	
Reconnection retry number	0
Disconnection timeout (mn)	0

NOTE: The default value is 0 for both parameters.

Runtime

OFS Network Windows Information

The OFS network window displays additional information when device connection advanced operating mode is configured.

This figure shows the new OFS network window with operating mode configured:
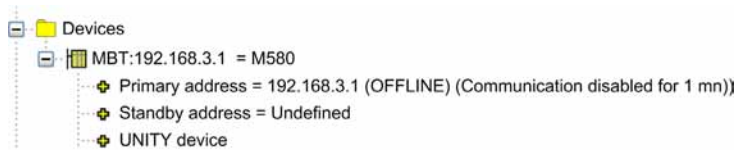
Chapter 20
Performance

Aim of this Chapter

This chapter aims to provide the user with details of the characteristics which can be used to improve server performance depending on the application and requirements.

What Is in This Chapter?

This chapter contains the following sections:

Section 20.1
Static characteristics

Aim of this Section

The aim of this section is to describe the static characteristics of OFS, and in particular the rules for generating and optimizing network requests. The objective here is to minimize as much as possible the number of requests.

What Is in This Section?

This section contains the following topics:

Data Items in a Request

Maximum size of requests

The following tables specify the maximum size of data in bytes in a single request. These sizes are useful as any data items accessed in the same request are from the same PLC cycle and so are consistent in size.

This size can be used to calculate the number of items of the same type which can be read or written in the same PLC communication request given that a word takes up two bytes, a double word 4 bytes, and a floating point 4 bytes.

For bits, you should count eight bits per byte except when reading with PL7 PLCs over an XWAY network where each byte can only contain 4 bits.

Example: for PL7 PLC on XIP, 248%MB or 62%MD or 124%MW or 992%M can be read in one request and 244%MB or 61%MD or 122%MW or 1960%M can be written in one request

Unity Pro devices

The tables below gives the size in bytes of data items a single request for Unity Pro devices:

Communication medium	Read	Write
XIP	249	235
XIP Built-in channel	256	242
TCP-IP Built-in channel	1022	1008
Pciway	224	210
USB	1022	1008
USB X-Way (USBX)	1020	1006
Fipway	123	109
Uni-Telway	241	227
Ethway	249	235
Modbus Plus	250	236
Modbus RTU	249	235

PL7 devices

The tables below gives the size in bytes of data items in a single request for PL7 devices:

Communication medium	Read	Write
XIP	248	244
XIP/Ethway 1 K (1)	1016	1012
Ethway	248	244
Fipway	120	116

Communication medium	Read	Write
ISAway	230	226
Uni-Telway	120	116

(1) : In this specific case, several PLC cycles are required to access the values.

Concept devices

The tables below gives the size in bytes of data items in a single request for Concept devices:

	Read	Write
Located variables	250	200
Unlocated variables for Concept 2.5 and later	244	1
Concept 2.2 unlocated variables	246	1

NOTE: The frame length for a Concept device is fixed (256 bytes). It does not depend on the communication media.

X-Tel devices

The tables below gives the size in bytes of data items in a single request for X-Tel devices:

Communication medium	Read	Write
Ethway	120	114
Fipway	120	114
Uni-Telway	120	114

Device ORPHEE

The tables below gives the size in bytes of data items in a single request for ORPHEE devices:

Communication medium	Read	Write
Ethway	1022	1016

Use of groups

Description

Dividing items into different groups can have an effect on the construction of network requests. For each device, the items are separated into independent sets if necessary. However, the sets will not be determined by the groups themselves but by the update periods of the groups.

Illustration:
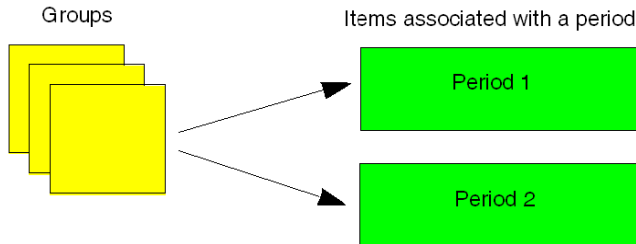
In summary:

- the groups do not influence the generation of network requests: declaring items in two different groups with the same period is the same, in terms of the requests generated, as declaring items in a single group.
- requests are generated not within a group, but within batches made up of items belonging to groups with the same period.

Optimizing requests

Description

Optimization is carried out individually for each set of items corresponding to a device and a frequency.

Optimization algorithms follow two stages:

- **Compacting**: grouping in tables of items of the same type which have similar or consecutive addresses. For writing, this regrouping is only carried out if the items are strictly consecutive. From the original items you obtain a list of elements to send to the PLC to read or write. Compacting is also applied for non-located data if the version of Concept used is 2.5 or higher. On Series 7 type PLCs, compacting is not carried out for unitary bits. For bit tables, it is only carried out if their number is multiplied by 8.
- **Concatenation**: construction of requests by optimizing the possibilities offered by the protocol. Certain protocols let you define access to several objects of different types in the same request. OFS automatically adjusts the size of requests to the maximum that is admissible:

OFS uses different communication protocols according to the devices being accessed.

List of protocols used with the algorithms in use:

- access to PL7 devices on X-Way (UNITE V2 protocol):
  - compacting in all cases,
  - concatenation for reading,

- access to PL7 and ORPHEE devices on non-X-Way networks (Modbus protocol):
  - compacting in all cases.
  - concatenation not possible.

- access to XTEL and ORPHEE devices on X-Way (UNITE V1 protocol):
  - compacting in all cases.
  - concatenation not possible.

- access to CONCEPT devices (Modbus protocol):
  - compacting in all cases. The types of basic data are fewer, and the possibilities for compacting are therefore greater than for UNITE protocol.
  - concatenation possible only for unlocated data of CONCEPT 2.5 devices and later.

- access to UNITY devices:
  - compacting in all cases.
  - concatenation in all cases.

Writing Concept structure type variables

Description

Concept gives you the opportunity to build data structures, made up of members of different types.

Unitary writing of bits: it is important to note that when the fields of bit or byte type are declared in the structure, they are not compacted. In fact, each of these fields is the object of a write request. Example: a structure made up of 2 bits and three consecutive words would give rise to 3 requests if the fields are written unitarily. Please note however that the write of the structure in its entirety would give rise to a single request.

Writing non-aligned fields: when the members are not aligned on the 16 bit boundaries, the writing of one of these members, where this cannot be carried out with a single request, is forbidden.

Illustration:

 Aligned structure: Non-aligned structure:

The real type member is considered as being stored on 3 addresses. The write would require 2 access bytes and 1 access word. When that is possible, it is preferable to build structures while taking into account the alignment criteria.

Important: the write of a complete structure, including when the members inside are not aligned remains possible.

Addressing of discrete I/O modules for M580, M340, and Premium devices

General

The "Read operation" and "Write operation" sections use a type of optimization known as "module optimization".

NOTE: For M580, topological syntax is supported only if the General \ Configuration\M580 preferred I/O data type value in Unity Project Settings is set to Topological.

Read operation

Items addressing the same module are compacted for discrete Input/Output modules

For example, for a discrete input **module**, reading of the following objects generates a request:

Object	Comment
%I1.0	input bit of rack 0, **module** 1 and channel 0
%I1.0.ERR	detected error on the channel of rack 0, **module** 1 and channel 0
%I1.2	input bit of rack 0, **module** 1 and channel 2
%I1.3.ERR	detected error on the channel of rack 0, **module** 1 and channel 3
%I1.6	input bit of rack 0, **module** 1 and channel 6
%I1.31	input bit of rack 0, **module** 1 and channel 31

However, if the module detected error bit is added to the objects above, given that it alone generates one request, the reading of all the objects will take two requests:

Object	Comment
%I1.0	input bit of rack 0, **module** 1 and channel 0
%I1.0.ERR	detected error on the channel of rack 0, **module** 1 and channel 0
%I1.2	input bit of rack 0, **module** 1 and channel 2
%I1.3.ERR	detected error on the channel of rack 0, **module** 1 and channel 3
%I1.MOD.ERR	detected error on the module of rack 0, **module** 1 and channel 3
%I1.6	input bit of rack 0, **module** 1 and channel 6
%I1.31	input bit of rack 0, **module** 1 and channel 31

Write operation

The concatenation of items (i.e. the construction of requests by optimizing the possibilities offered by the protocol) addressing the same **module** is performed for discrete Input/Output modules.

OFS automatically adjusts the size of requests to the maximum admissible by the protocol.

For example, on a Uni-Telway bus, for a discrete output module, writing of the following objects generates a request:

Object	Comment
%Q2.0	output bit of rack 0, **module** 2 and channel 0
%Q2.1	output bit of rack 0, **module** 2 and channel 1
%Q2.3	output bit of rack 0, **module** 2 and channel 3
%Q2.10	output bit of rack 0, **module** 2 and channel 10
%Q2.11	output bit of rack 0, **module** 2 and channel 11
%Q2.31	output bit of rack 0, **module** 2 and channel 31

Addressing of analog I/O modules for M580, M340, and Premium devices

General

The "Read operation" and "Write operation" sections use a type of optimization known as "channel optimization".

NOTE: For M580, topological syntax is supported only if the General \ Configuration\M580 preferred I/O data type value in Unity Project Settings is set to Topological.

Read operation

In read mode, objects are compacted and concatenated on items addressing the same **channel** of an Input/Output module.

For example, reading of the following objects generates a request:

Object	Comment
%IW1.0.2	input word of rack 0, module 1, **channel** 0 and position 2
%IW1.0.3	input word of rack 0, module 1, **channel** 0 and position 3
%IW1.0.10	input word of rack 0, module 1, **channel** 0 and position 10
%ID1.0	double input word of rack 0, module 1, **channel** 0 and position 0
%ID1.0.4	double input word of rack 0, module 1, **channel** 0 and position 4
%ID1.0.6	double input word of rack 0, module 1, **channel** 0 and position 6
%ID1.0.11	double input word of rack 0, module 1, **channel** 0 and position 11

The same applies for Fipio agent objects:

Object	Comment
%IW\0.2.54\0.0	input word at connection point 54 of a base module, **channel** 0 and position 0
%IW\0.2.54\0.0.1	input word at connection point 54 of a base module, **channel** 0 and position 1
%IW\0.2.54\0.0.2	input word at connection point 54 of a base module, **channel** 0 and position 2
%IW\0.2.54\0.0.29	input word at connection point 54 of a base module, **channel** 0 and position 29
%IW\0.2.54\0.0.30	input word at connection point 54 of a base module, **channel** 0 and position 30
%IW\0.2.54\0.0.31	input word at connection point 54 of a base module, **channel** 0 and position 31

If you want to address different channels, you should allow one request per addressed channel. In this example, 5 requests are generated:

Object	Comment
%IW1.0	input word of rack 0, module 1, **channel** 0 and position 0
%IW1.1	input word of rack 0, module 1, **channel** 1 and position 0
%IW1.3	input word of rack 0, module 1, **channel** 2 and position 0
%IW1.4.2	input word of rack 0, module 1, **channel** 4 and position 2
%IW1.15	input word of rack 0, module 1, **channel** 15 and position 0

The module detected error bit generates an additional request, whereas the module channel bit does not. If the module channel bit does not generate an additional request, the module detected error bit will be included in the same request.

Write operation

In write mode, objects are concatenated on items addressing the same **channel** of an Input/Output module.

For example, on a Uni-Telway bus, writing of the following objects generates a request:

Object	Comment
%QD1.0	double output word of rack 0, module 1, **channel** 0 and position 0
%QW1.0.2	output word of rack 0, module 1, **channel** 0 and position 2
%QW1.0.3	output word of rack 0, module 1, **channel** 0 and position 3
%Q1.0	output bit of rack 0, module 1, **channel** 0 and position 0

Restrictions and Advice for Input/Output Objects on PL7 Devices

Input/Output Performance

Reading Input/Output items generates a large number of requests. You should therefore beware of any loss of performance which may occur, especially when addressing in addition items other than I/O items.

Defining an I/O Item when the Device or I/O Module is not Connected

If the device and/or the I/O module are not connected when an item is defined, "module optimization *(see page 368)*" is not performed.

As a result, the channels of discrete I/O modules are addressed using "channel optimization" *(see page 370)*, that is with one request per channel.

Management of the Fallback/Forcing State of a Discrete Output Module

- When an output channel of a discrete module is in a fallback state, OFS detects this and sets the quality for the related item to `Uncertain`,
- when an output channel is in a forced state, no specific operation is performed as the value displayed corresponds to the current forcing value.

NOTE: Write operations cannot be taken into account as long as the channels concerned remain in a fallback state.

When a Premium is in **Stop** mode after a download has been performed, the discrete output channels are not in a fallback state and the `%Q` are assigned the quality `Good`.

In contrast, when a Micro is in **Stop** mode after a download has been performed, the discrete output channels are not in a fallback state but the `%Q` are assigned the quality `Uncertain`.

Management of the Fallback/Forcing State of an Analog Output Module

- OFS does not have the capacity to detect that an ANA module output channel has a fallback or forced status. An item's quality in relation to an analog module's output channel is always therefore `Uncertain`.
  As a result, detection of these states must be performed by the PLC application.

NOTE: Write operations cannot be taken into account as long as the channels concerned remain in a fallback state.

Access to I/Os on Gate 7

- X-Way addressing mode cannot be used with gate 7 *(see page 343)* to access I/O objects.

Supported I/O Modules

- Only the following I/O modules are supported: the families TSX DEY, TSX DSY, TSX DMY, TSX DEZ, TSX DSZ, TSX DMZ, the families TSX AEY, TSX ASY, TSX AEZ, TSX ASZ, TSX AMZ and the Momentum and TBX families.

Section 20.2
Dynamic Performance

Dynamic performance

Introduction

The dynamic performance of OFS depends on several parameters and can be measured according to several characteristics (configuration response time, read/write response time, volumes of data exchanged, sensitivity to anomalies) and along two lines (OFS communication with devices and OFS communication with OPC clients). In certain cases it is necessary to configure different OFS parameters to obtain better performance. For example, if devices are accessed via different types of network and a lower performance network is used on somewhere on the network path. Below, we provide you with the server adjustment parameters which have an influence on performances for OFS communication with devices.

Multi-channel feature

On half-duplex networks *(see page 348)*, this parameter can be used to send several requests to a device simultaneously. The higher the value, the better the performance you will obtain for communication with the device.

Specific item

The number of requests sent *(see page 193)* in parallel to a device (see specific item #NbrMaxPendingRequest).

This parameter is calculated automatically by OFS according the device accessed. However, as OFS does not know everything about the system environment, it is sometimes necessary to adjust it to use different values. You may wish to reduce it to allow other interfaces (programming workshop, diagnostics tools, another device) to communicate more easily with the device. You make also wish to reduce it if the route used to access the device may be subject to bottlenecks (E.g.: Access via XIP then UNITELWAY). It is not advisable to increase this value. However, in some cases (on TCP/IP or XIP networks) doing so may make it possible to process larger volumes of data.

Frame timeout

This parameter specifies the maximum tolerated time period for obtaining a response to a request from a device.

It is advisable to set it at a reasonably high value because:

● if it is too low, "false" timeouts (the response arrives after the specified period has elapsed) may have a negative effect on performance.
● if it is too high, feedback times may be too long and missing devices may be detected too late.

The user must therefore find the compromise best suited to his requirements.

Sampling speed

The lower the value, the better the response times you obtain for synchronous access will be. If an unsuitable period is configured (one which is too short relative to actual requirements), this may have a negative effect on performance. However, this parameter only has an influence on the machine time (for PC on which OFS is installed) and not on network bandwidth.

Group update rate

OFS manages communication with devices according to the group update rate. It is therefore advisable to create groups of variables according to the update rates required by the client so OFS can optimize access to devices. You should however be careful not to create groups that are too small as this will generate a large number of requests.

Section 20.3
Estimation of Network Performance

Aim of this Section

This section provides you with some data that can be helpful in anticipating the performance you can expect from your network.

What Is in This Section?

This section contains the following topics:

PLC Capacity

PLC Capacity for OFS on Ethernet network

Each PLC can handle a different number of requests at each scan of the MAST task. The following tables summarize the request handling capacities of each PLC product range.

PLC Quantum

Communication adapter	CPU	Request size	Request per PLC scan time
Copro	140 CPU 6••	1024	16 for located or unlocated
NOE	140 CPU 6•• 140 CPU 5•• 140 CPU 4•• 140 CPU 3••	256	From 12 to 16 for located From 1 to 4 for unlocated
NWM	140 CPU 6•• 140 CPU 5•• 140 CPU 4•• 140 CPU 3••	256	From 1 to 4 for located 1 for unlocated
NOE 1024 (*)	140 CPU 6•• 140 CPU 5•• 140 CPU 4•• 140 CPU 3••	1024	12 for located or unlocated
(*) On CPU modules with OS version 2.80 or higher and on NOE modules with OS version 4.60 or higher.			

PLC Premium

Communication adapter	CPU	Request size	Request per PLC scan time
Copro	TSX P57 6•• TSX P57 5•• TSX P57 4••	1024	16 for located or unlocated
ETY / WMY	TSX P57 6•• TSX P57 5•• TSX P57 4••	256	16 for located or unlocated
ETY / WMY	TSX P57 3••	256	12 for located or unlocated
ETY / WMY	TSX P57 2••	256	8 for located or unlocated
ETY / WMY	TSX P57 1•• TSX P57 0••	256	4 for located or unlocated

PLC M340

Communication adapter	CPU	Request size	Request per PLC scan time
Copro	BMX P34 20••	1024	8 for located or unlocated
NOE	BMX P34 10•• BMX P34 20••	1024	8 for located or unlocated

PLC M580

Communication adapter	CPU	Request size	Request per PLC scan time
Embedded port	BMX P58 1020	1024	8
Embedded port	BMX P58 20x0	1024	12
Embedded port	BMX P58 30x0	1024	16
Embedded port	BMX P58 40x0	1024	16

Request Capacity:

The size of the answer frame gives the number of needed requests:

Number of Variables	256 bytes (240 data bytes)	1024 bytes (1000 data bytes)
5.000 words	42 requests	10 requests
5.000 BOOL	3 requests	1 request
30.000 words	250 requests	60 requests
30.000 BOOL	16 requests	4 requests

The address of each variable can take up to 8 bytes depending on the module and protocol. The Unite or Modbus protocol, allows up 240 bytes of data.

NOTE: If the variables are not contiguous, the number of requests is greater.

Estimation of Time to Read Several Variables:

Task time to 10 ms

PLC	PL7 TSX P57 1••	CONCEPT	UNITY PREMIUM TSX P57 5••		UNITY QUANTUM 140 CPU 651 60		M340	M580
Protocol	Unite	Modbus	Unity		Unity		Unity	Unity
Module	ETY	NOE	COPRO	ETY	COPRO / NOE 1024	NOE	CPU/NOE	Embedded port BMXP5830x0
Request size	256	256	1024	256	1024	256	1024	1024
Nb of requests per scan	4	12	16	4	12	4	8	16
Nb of needed requests (5.000 / 30.000 words)	42 / 250	42 / 250	10 / 60	42 / 250	10 / 60	42 / 250	10 / 60	10 / 60
5.000 words	220 ms	80 ms	20 ms	220 ms	20 ms	220 ms	40 ms	20 ms
30.000 words	1.26 s	420 ms	80 ms	1.26 s	100 ms	1.26 s	160 ms	80 ms

Task time to 50 ms

PLC	PL7 TSX P57 1••	CONCEPT	UNITY PREMIUM TSX P57 5••		UNITY QUANTUM 140 CPU 651 60		M340	M580
Protocol	Unite	Modbus	Unity		Unity		Unity	Unity
Module	ETY	NOE	COPRO	ETY	COPRO / NOE 1024	NOE	CPU/NOE	Embedded port BMXP5830x0
Request size	256	256	1024	256	1024	256	1024	1024
Nb of requests per scan	4	12	16	4	12	4	8	16
Nb of needed requests (5.000 / 30.000 words)	42 / 250	42 / 250	10 / 60	42 / 250	10 / 60	42 / 250	10 / 60	10 / 60
5.000 words	1.1 s	400 ms	100 ms	1.1 s	100 ms	1.1 s	200 ms	100 ms
30.000 words	6.3 s	2.1 s	400 ms	6.3 s	500 ms	6.3 s	800 ms	400 ms

Task time to 200 ms

PLC	PL7TSX P57 1••	CONCEPT	UNITY PREMIUM TSX P57 5••		UNITY QUANTUM 140 CPU 651 60		M340	M580
Protocol	Unite	Modbus	Unity		Unity		Unity	Unity
Module	ETY	NOE	COPRO	ETY	COPRO / NOE 1024	NOE	CPU/NOE	Embedded port BMXP5830x0
Request size	256	256	1024	256	1024	256	1024	1024
Nb of requests per scan	4	12	16	4	12	4	8	16
Nb of needed requests (5.000 / 30.000 words)	42 / 250	42 / 250	10 / 60	42 / 250	10 / 60	42 / 250	10 / 60	10 / 60
5.000 words	4.4 s	1.6 s	400 ms	4.4 s	400 ms	4.4 s	800 ms	400 ms
30.000 words	25.2 s	8.4 s	1.6 s	25.2 s	2 s	25.2 s	3.2 s	1.6 s

Chapter 21
Client-alive Service

Client Alive Service

Functionalities

In certain cases, OFS needs to detect OPC clients' improper disconnection (proper disconnections are detected) to go to a falling back state and self deactivate all active groups. The OPC client signals it is alive by writing periodically a **#Client Alive** specific item. The OPC client handling the group is considered as missing if no **#Client Alive** write action has been performed during a configured client-alive timeout.

NOTE: Improper disconnections are managed automatically by OFS OPC driver.
This following screen shows the client-alive service that appears in the **OFS Server settings → Options** panel:
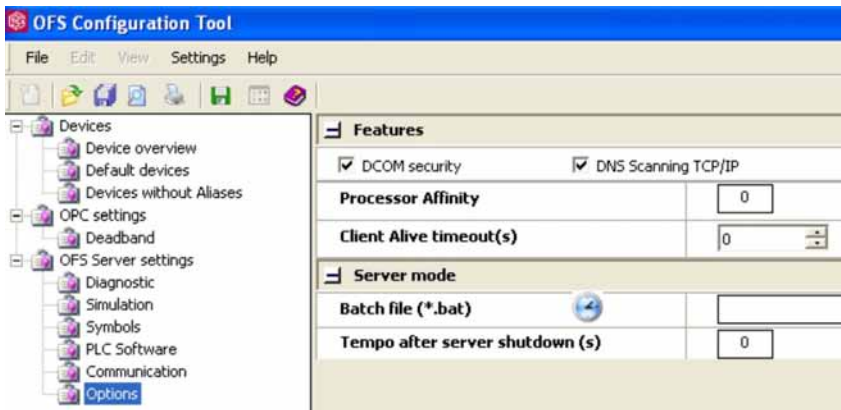
The Client Alive time-out(s) service configuration:
The range of values is 0 to 30 by increments of 2.
The value is 0: the service is disabled.

NOTE: A **#Client Alive** can be added in any group except the reserved event group. The monitoring applies to all active groups managed by the client, including the reserved event group.

NOTE: A **#Client Alive** specific item is not linked to an alias but to the system. To keep normal item syntax, the browse publishes a **<<system>>** root branch and **#Client Alive** leaf. The full syntax is **<<system>>! #Client Alive**.

NOTE: The client-alive monitoring of a client starts when the first **#Client Alive** write is performed (if the service is not disabled). This allows the service to remain optional for other OPC clients.

OFS Runtime:

- A monitoring task computes periodically the time elapsed between 2 consecutive **#Client Alive** write operations. If that time is superior to the configured **Client Alive timeout**, then all active groups owned by the client are deactivated by OFS.
  Periodically write **#Client Alive** at times **Client Alive timeout**. If it is not the case, the groups are deactivated in an unexpected way. A typical value is `<#Client Alive writing_rate>` `= <client-alive Timeout> /2`

Part VII
Developer Guide

Chapter 22
Advice

Aim of this Chapter

The aim of this chapter is to provide developers tips on how to use the OFS product.

What Is in This Chapter?

This chapter contains the following topics:

Programming

Description

The main phases of application client programming (using either VB and Automation Interface 2.0 or C++ and Custom Interface) are as follows:

- creation of a connection with the OFS server (in local or offset mode):
  OPC-AUTOMATION: Connect()
  OPC-CUSTOM: CoCreateInstance() + QueryInterface() for IOPCServer + Get GroupCollection(),
- creation of one or more GROUPS:
  OPC-AUTOMATION: GroupCollection \ Add() + Get ItemsCollection()
  OPC-CUSTOM: IOPCServer \ AddGroup()+ QueryInterface() for IOPCItemMgt,
- creation of ITEMS in an existing group:
  OPC-AUTOMATION: ItemsCollection \ AddItem() or AddItems()
  OPC-CUSTOM: IOPCItemMgt \ AddItems(),
- READ or WRITE of group ITEMS:
  OPC-AUTOMATION: ptr \ ASyncRead() group or ptr \ ASyncWrite() group
  OPC-CUSTOM: IOPCASyncIO2 \ Read() or IOPCASyncIO2 \ Write(),
- destruction of existing GROUPS (this may include the destruction of all items contained within these groups):
  OPC-AUTOMATION: GroupCollection \ Remove() or RemoveAll()
  OPC-CUSTOM: IOPCServer \ RemoveGroup(),
- closing the CONNECTION with the OFS server:
  OPC-AUTOMATION: Disconnect()
  OPC-CUSTOM: IOPCServer \ Release().

Cyclic read of a group of items

Installing periodic read for a group of items requires the use of a notification mechanism, set up by carrying out the following additional operations:

1	Activation of a group and at least one of its items
2	Subscription to notification service
3	Regular receipt of notifications ("waken" function)
4	Cancellation of subscription to notification service
5	Deactivating groups and items

NOTE: The above information is only needed for the creation of new personalized applications.

Recommendations

At a Glance

This chapter contains several recommendations to enable optimum use of the server. As a general rule, you should remember that the limit on the number of items that are simultaneously accessible is linked to the communication resources between the OFS server and the devices. The limiting element is the entrance of communication modules on to PLCs.

- for a group containing a large number of items (several thousands), the creation of items or the modification of the group properties (update rate, for example) is made much quicker by deactivating the group beforehand and then reactivating it when the operation is complete. This point is particularly important when using synchronous groups ($ and $$), as for each item created, the destination of the new item is checked with respect to the first item created in the group.
- when using a large number of items (several thousands), divide them up into several groups to adapt the update rate and therefore be able to desynchronize them. In order to avoid communication peaks with devices:
- when developing an application, it is preferable to use the "AddItems" method which is more powerful than the simple "AddItem" method.

Part VIII
Appendices

Chapter 23
Appendices

Aim of this Chapter

The aim of this chapter is to introduce the appendices for this book.

What Is in This Chapter?

This chapter contains the following sections:

Section 23.1
Compatibility of the OFS server

OFS server compatibility

Definition

OFS is compatible with OPC 1.0 A and 2.0.

In particular, the OFS server accepts the notion of a mono-request, mono-PLC synchronous group. Syntactically, the name of a synchronous group starts with "$" (see *The Different Groups of Items, page 261*).

The OFS server is also compatible with the notion of a system group dedicated to a PLC address and driver pair:

The system groups refer to a given device and are used to manage specific items related to that device. System groups are distinguished from user groups by their name, which must include the prefix "_SYS=".

A system group only contains the following specific items starting with "#":

- #PLCStatus" for managing the PLC operating mode,
- #TimeOut" for managing a communication medium timeout,
- #NbrRequest" for ascertaining the number of requests sent to this device.

Specific items and system groups cannot be activated. Notification and asynchronous read/write are not possible.

Section 23.2
Detected Error codes

Detected Error Codes Defined by OLE, OPC and the OFS Server

General

For information on the code sent, simply run the scoder.exe utility then enter the code and click "OK".

Section 23.3
Modbus and UNITE Request Codes used by OFS

Modbus and UNITE Request Codes Used by OFS

Description

This is a list of all the request codes used by the OFS server. If your device does not support a request code, this feature will not be available. If you do not use the feature, the request code WILL NOT BE generated.

For Modbus devices, even if the request code is supported, the maximum length may not be.

Modbus request codes used by OPC Factory Server:

Request code (hex)	Function name	Max. length used	OFS features using the request code
01	Read Coil Status	2000	Read of 0x items
02	Read Input Status	2000	Reading 1x items
03	Read Holding Registers	125	Read of 4x items and device detection (with 0 reg)
04	Read Input Registers	125	Read of 0x items
05	Force Single Coil		Write of a single 0x item
0F	Force Multiple Coil	800	Write of several 0x items
10	16 Preset Multiple Registers	100	Write of any number of 4x items
11	Report Slave ID		Read of device operating mode
16	Mask Write 4X registers		Write of a byte item located in the 4x area
2 A	IEC Runtime FC		Access to unlocated Concept variables (read/write)
5 A	Internal Unity request		Access to Unity PLC
7E	Modsoft FC		Starting / Stopping the device

UNITE request codes used by OPC Factory Server:

Request code (hex)	Function name	OFS features using the request code
0F	Identify	Device detection
24	Start	Starting the device
25	Stop	Stopping the device
33	Initialize	Initializing the device
36	Read Object	Read of all items on TSX Series 7, S1000
37	Write Object	Write of all items on TSX Series 7, S1000
38	Read Object List	Read of all items on Premium, Micro
4F	Read CPU	Device detection and read of the device's operating mode on Premium, Micro
5 A	Internal Unity request	Access to Unity PLC
83	Write Generic Object	Write of all items on Premium, Micro
FA	Mirror	Detection of max PDU size

Section 23.4
Recommendations

Location of an Anomaly

Description

The table below shows a number of situations that can be easily avoided. If the suggested solution is not sufficient, contact Schneider Electric SA support services.

Component	Anomaly	Remedy
Configuration Tool Installation	Program start up retrieves the code "Ox1AD".	Reinstall the MDAC component. It is supplied on the DVD in the REDIST directory.
Configuration tool Startup	Anomaly while executing the program (untimely stopping of the PC, etc.) or incorrect startup (for example, damaged database).	If you have backed-up aliases: If the configuration tool still starts up, retrieve the last alias file using the "get archive" menu. If the configuration tool does not start up, try to copy the back-up file saved in "alias2 K.mdb" in the configuration tool directory manually. This operation will overwrite the working database, which is probably corrupted. Then try to restart the program. If you have not backed-up: Uninstall the configuration tool, then reinstall it. The compatibility window will be displayed during the first start up of the configuration tool. Select YES to start the recovery procedure.
Configuration tool Installation	After upgrading from version 2.0 to the current version, the parameters set with the v2.0 configuration tool no longer appear.	It is likely that this is due to the fact that no alias has been declared, the only case in which recovery is not possible.
Configuration tool Backup copy	After the aliases have been backed up, the back-up file cannot be located.	If you have backed up via the network neighborhood, it is essential that a directory name be given. If not, the file will be backed up in the configuration tool's default directory or in the directory containing the configuration tool start up short cut.
Configuration tool Startup	Alias recovery does not work.	Check in the properties of the short cut used to run the configuration tool that the quotation marks in the chain of ofsconf.exe characters are not double quotation marks.
Configuration tool On-line help	On-line help does not work.	On-line help requires at least Internet Explorer v3.02.

Component	Anomaly	Remedy
Configuration tool	My new configuration tool parameters are not taken into account.	Confirm the configuration tool and close and then reopen the server.
Configuration tool	NOK connection between the client and the remote server.	Check not only the DCOM parameters, but also the 'DCOM Security' option of the configuration tool.
Driver	Communication with the PLC is not working.	For X-Way, check with the "X-Way TEST" utility of the "X-Way Driver Manager" whether communication is possible. If it isn't: See either the driver or the connections (see appropriate manuals), If it is: Check the required consistency level; it is likely that the PLC application version may be different to that of the symbols file.
Driver	The message "Identity failure for XIP01: 20.22" appears in the debug window of the server (this message can appear for other drivers).	Check that the compatible drivers for OFS are installed.
Driver	Communication with the Ethway driver does not work.	Check the driver installation, check the settings in NetAccess (refer to the product documentation provided with the Driver DVD).
Driver	The use of driver instances greater than 1 does not work.	NetAccess must be set up.
Installation	There may be anomalies if the destination directory access path (the directory under which the server and configuration tool should be installed) is too long and the hard drive on your PC has been converted from FAT to NTFS.	In this case, try to use short file names (for example, C:\OFS - instead of C:\Program Files\Schneider Electric\OFS).
Server, Target device: Premium	Performance is greatly inferior to that expected and/or indicated in the documentation.	Make sure: ● that you have not inadvertently checked the "X-Tel" option in the properties page. ● the OFS is configured to obtain optimum performance. For example, if devices are accessed via different types of network and a lower performance network is used on somewhere on the network path (E.g.: Access via XIP then UNITELWAY). The parameters which may have an influence on communication performance are: ❍ the number of requests sent in parallel, ❍ the frame timeout.

Component	Anomaly	Remedy
Server Installation	The message "QueryInterface(IID_IOPCServer) returned E_NOINTERFACE for server Schneider-Aut.OFS" appears when an OPC client tries to connect to the server.	It generally appears when you have chosen not to restart the PC after installation. Restart it.
Server Detected Error codes	Messages referenced by a numerical code are displayed in the diagnostics window or in the client.	A decoding program is supplied: run scoder.exe from the server's installation directory.
Server Concept link	The message "cannot connect to local cc2cat" or "unable to load Concept .prj file" appears.	Check that the cc2cat is installed correctly and saved. See readme.txt in the \ConceptLink directory of the DVD.
Server System configuration	Use of screen savers.	The use of screen savers is not recommended with OFS server, except the password-protected blank screen ("lock computer" or "lock WorkStation" option). It is not advisable to use power saving.
Server Installation	User rights.	The OFS server can only be installed if the session is opened with an ADMINISTRATOR account. The server and any local client can work perfectly well under the same non ADMINISTRATOR account.
Server Remote client	Remote access anomalies.	In order to be used remotely by an OPC DCOM client, the OFS server must be started up using an ADMINISTRATOR account. Check that the user rights are correctly managed: adjust with the DCOMCNFG.exe tool.
Server Development of a VB client	"User-defined type not defined" message during the compilation of the declaration of the "OPC Server" object in Visual Basic.	Instruction: Dim WithEvents OpcFactoryServer As OPCServer. The interfaces presented by the OFS server are not recognized by Visual Basic. You should save these interfaces using the "Tools" > "References" menu in VB6 SP3 then select SA OPC Automation 2.0 as this OPC Automation manager contains the server's "Library Type".
Server Use of symbols	EOL_E_OPEN_SYMBOLS_FAILURE obtained when creating a user group with a symbols file.	If you have not given an absolute path for the file, check that the "Symbols" option is correct in the configuration tool. If you have not defined this option, check that the C:\OPC_SYMB directory exists and that it contains your file. If you declare a group name, make sure that there is no space after the = sign. For example: "grpName= symb.scy" becomes "grpName=symb.scy".

Component	Anomaly	Remedy
Server Target device: Series 7	Unoperational declaration of an item and display in the X-Way path: Answer Thread: Invalid Protocol (UNITE V1?) for Answer 253.	Check in the device's alias address that the "X-Tel" label is present (validation of the X-Tel API parameter in the assisted address entry screen).
Server OFS manager	The "Device Identity" field shows "????".	If it is a type S1000 device, this is normal, the PLC's Ethernet module does not return any identification.
Server Uni-Telway	Message "X-Way: Build Request for UNTLW01 :0.254.0 : No Free Socket" appears frequently in the diagnostics window.	The Uni-Telway driver is adjusted with too low a number of slave addresses. This can be changed depending on the number of requests used.
Server Diag buffer	The quality attribute of the #DiagReadNextError item still says "bad" even when the connection with the device is open.	The "handle" of the OPC client, which contains the specific items of the Diag Buffer, is already in use or the client has connected to the server without specifying their "handle".
Remote station	Installation not possible, displaying the message "Automatic save error". One or several files are not saved automatically (OPCAutoSA2.dll, SAProxy.dll).	The DCOM component is necessary. Install it from the Redist\DCOM folder, then restart the PC. Then resume the installation of the remote station.
OPC server or client	Modifying a group's period takes a lot of CPU time if the group contains a large number of items.	The "Network windows" window, which shows all the requests generated, should be closed since the quantity of traces displayed penalizes the operation. As a general rule, before you modify the period of a group, it is recommended that you first deactivate the group then reactivate it when the operation is complete.
Server Communication	The server no longer responds when lots of items are simulated. "Spurious" messages appear in the Diagnostics window.	Adjust the probability of notifications from the simulator to reduce the number of notifications sent to the OPC client. Increasing the power of the PC can raise the limit.
Server Display	In the Windows Task Manager you notice an increase in the amount of memory taken up by OFS.EXE.	If the server is in Diagnostics mode, a large quantity of displayed messages causes an increase in the amount of memory used (up to 4 Mb). This is normal. By closing the HMI windows of the server, the memory is freed up.
Server Installation	The message "CoCreateInstance returned REGDB_E_CLASSNOTREG for Server Schneider" appears when an OPC client tries to connect to the server.	This appears when the MSVCP60.DLL component is missing or is not installed correctly. Put this DLL in the Windows directory and reinstall the server.
Server Concept link	With Concept 2.5, while opening the symbols file, the message "SdkConcp: can't open project" appears.	Consult the file \ConceptLink\Version2-5\Readme.txt.

Component	Anomaly	Remedy
OPC client Address format of the items	All the items in a group are set to Quality = Bad (24).	This status can result from the addition of a non existent item in the target device. E.g.: reading %MW10000 when in the PLC words only go up to 8000. Here, you must delete items outside the zone.
Server Writing items	When writing a large number of items, the following messages appear in the diagnostics window: "SyncWriteFailure" followed by "Write Error".	These messages appear due to the fact that the write operation lasted longer than the authorized time limit. Increase the frame time-out for the devices concerned.
Server Notification	Absence or delay in the notification of certain items. The message "Error, request too old" appears in the diagnostics window of the server.	Increase the period of the group. If this is not sufficient, alter the timeout values of the alias. For Modbus, increase the number of channels allocated for communication.
Server Accessing items	Access via Modbus to more than 1000 consecutive bits on a Premium PLC is not possible.	Make sure that the option /T is indicated at the end of the device address.
Server Activating items	The activation of a significant number of items is not possible if they were created in the active group.	This is due to overloading of the PC. The following advice may help to resolve it: ● create the items in an inactive group, then activate the group, ● check that 'Verbose' server mode is not selected in order to keep traces to a minimum, ● increase the power of the PC.
Server Push Data	In Modbus, the values of the items defined in the Push Data zone are not updated.	The base address to be indicated in the alias properties should only contain the offset of the address. Therefore, to indicate a base address in Modbus corresponding to 402000 for example, simply indicate 2000 as the value in the "Base address" field.
Symbol	The symbols from a Concept V2.5 file cannot be accessed.	Make sure the .VAR file is present in the Concept project space. This .VAR file is generated by the Concept workshop (refer to the product documentation) and absolutely must exist in order for OFS to be able to access the symbols.

Glossary

A

Address

Builder name for a PLC variable. For example "%MW1".

Alias

An alias is a shortcut that may be used when a network address for the device is necessary (single replacement string). The use of an alias is also a very practical way to disconnect your OPC application from network addresses of devices that may be modified when necessary.

ASP

Active **S**erver **P**age allows a Web site builder to dynamically create web pages. It supports the code written in compiled languages such as by Visual Basic, C++, C #, etc.

C

CCOTF

Configuration **C**hange **O**n **T**he **F**ly.

Client application

Software using the primitives provided by a server application, via mechanisms (interfaces) implemented by OLE.

CLR

Common **L**anguage **R**untime is part of the .Net framework. It is the program that controls execution of programs written in all supported languages allowing them to understand each other. It also controls the security aspect.

CLS

Common **L**anguage **S**pecification allows the user to optimize and ensure the interoperability of languages by defining all functions that developers may use in numerous languages.

COM

Component **O**bject **M**odel: foundations of the OLE 2.0 standard.

CRA

Communicator **R**emote **A**daptater: drop end communicator.

CRP

Communicator **R**emote **P**rocessor: I/O network head module or bus head communicator.

D

DCOM

Distributed **COM**: COM model distributed over a TCP-IP network.

F

FIP

Factory **I**nstrumentation **P**rotocol.

FTP

File **T**ransfer **P**rotocol is the standard internet protocol that is used for exchanging files between computers and the internet.

G

GAC

Global **A**ssembly **C**ache contains all assemblies necessary for .NET and manages different versions of assemblies.

H

Handle

Single value identifying the object.

HTML

Hyper**T**ext **M**ark-up **L**anguage is the language used to describe Web pages.

HTTP

Hyper**T**ext **T**ransfert **P**rotocol is the protocol used for transferring HTML pages.

I

IDE

Integrated **D**evelopment **E**nvironment is a program that includes a code editor, a compiler, a detected error analyzer and a graphic interface.

IIS

Internet **I**nformation **S**erver is the ftp, Web or HTTP server developed by Microsoft to work under Windows.

Impersonation

Ability to execute a thread with a different security context than that of the thread's owner in a client/server application. When a client contacts a server, the server typically runs with the security context of some service account that has access to every resource that it might possibly need to carry out a request.

J

JRE

Java **R**untime **E**nvironment is a subgroup of the Sun Java development kit that may be embedded in an application. JRE provides minimum conditions (an environment) for running a Java application.

L

LCID

Language **C**ode **ID**entifier.

M

Multi-clients

Several client applications simultaneously access the same server application.

O

OFS

OPC Factory **S**erver: OLE server for exchanging data with the PLC.

OLE

Object **L**inking and **E**mbedding: object for linking and embedding. In particular supplies the OLE Automation interface, a technique which enables a server to display the methods and properties to a client.

OPC

OLE for **P**rocess **C**ontrol.

OPC group

Controls a collection of **OPC** items, that is a list of PLC variables.

OPC item

PLC variable on a PLC and a given communication medium.

OPC server

Controls a collection of OPC groups. Hierarchical root of the OPC model.

P

PLC

Programmable **L**ogic **C**ontroller: programmable controller (industrial).

Primitive

OPC function

R

RCW

Runtime **C**allable **W**rapper: The main function is to group calls between .Net client and the non-managed COM object.

RDE

Read **D**ata **E**ditor: The OFS RDE is used to display and edit the variables of devices from a table based on a Java application or window.

Remote server

The client and server application are located on 2 separate stations, linked by the Microsoft TCP-IP network.

S

Server application

Software presenting the primitives to the client applications, via mechanisms (interfaces) implemented by **OLE**.

SOAP

Simple **O**bject **A**ccess **P**rotocol a Microsoft protocol using HTTP and XML for information exchange.

Socket

Communication channel established between the OFS server and one or more PLCs, on a given communication media. The number of sockets available depends on the communication medium.

SOE

Sequence **O**f **E**vents.

SP

Service **P**ack: operating system corrections and upgrades

Symbol

Identifier attributed by a designer to a control system. For example "PUMP". A symbol cannot start with the prefix '%'.

U

UNC

Universal **N**aming **C**onvention.

V

VB

Visual **B**asic: consumer language supporting OLE Automation.

VBA

Visual **B**asic for **A**pplications: Script language using Basic syntax included in the MS-Office Suite.

W

Wintel

Windows/**Intel**: describes a PC equipped with a 32-bit Windows operating system and an Intel x86 processor.

WSDL

Web **S**ervice **D**escription **L**anguage provides a basic model in XML format for describing Web services.

X

XML

e**X**tensible **M**arkup **L**anguage is an derived extensible meta-language used for structuring data.

Index