# OPC Factory Server V3.33

## User Manual

03/2009

35008244.05

www.schneider-electric.com

Schneider Electric assumes no responsibility for any errors that may appear in this document. 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.

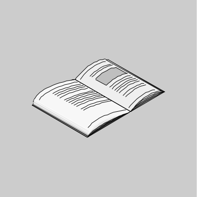
All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

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, 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 an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

**⚠ WARNING**

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

**⚠ CAUTION**

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

**CAUTION**

**CAUTION**, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage.

### 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.

# About the Book

## At a Glance

### Document Scope

This manual describes the software installation of the OFS product.

### Validity Note

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

### User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

# Introduction to the OFS product

# I

# Uses of the OFS product

# 1

## Aim of this Chapter

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

## What's 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 TSX Compact, micro, TSX Momentum, TSX/PCX Premium, Quantum, M340, TSX Series 7 and TSX S1000 families in order to supply the OPC clients with data.

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

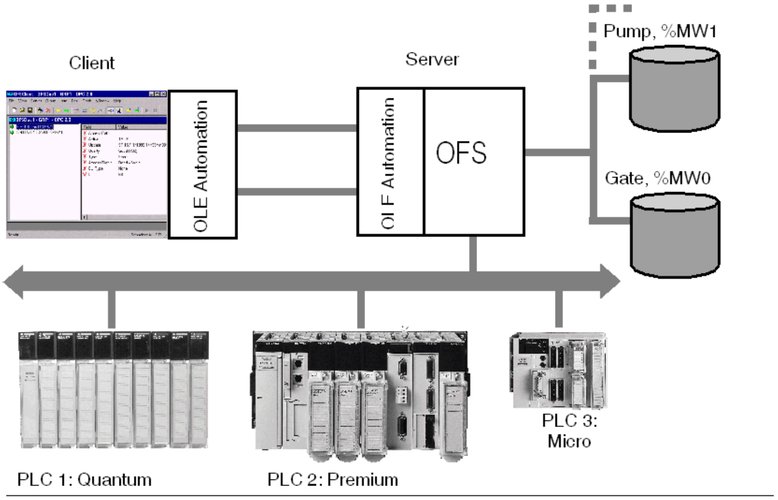
OFS is compatible with versions OPC 1.0A and OPC 2.0. It will function with an OPC client up to version 2.0a and with two types of OPC software, i.e.:

- Supervisory software (see distributor offer): The OFS server plays the role of a driver by 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 above,
- Microsoft Visual C++, version 6.0 SP3 or above,
- Microsoft VBA in Excel, version 8.0 (Office 97) or higher,
- Microsoft Visual C#,

The illustration below shows an OFS interface:



The OFS server provides the interface between Schneider 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 1.0A 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 *(see page 194)* 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 with the Schneider Electric SA Quantum, PCX Premium, Micro, Momentum, Compact, Series 7, S1000 and M340 PLC ranges 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 3.33 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<br>TSX ETH 200 | ETH030• |
| **Modbus TCP/IP** | TSX ETY110• (TCP/IP)<br>TSX ETY410• (TCP/IP)<br>Built-in channel<br>TSX ETY510• (TCP/IP) | TSX ETZ410<br>TSX ETZ510 | | |
| **Uni-Telway** | Built-in channel<br>TSX SCP11• | Built-in channel<br>TSX SCP11• | TSX SCM22• | |
| **Fipway** | TSX FPP20 | PCMCIA<br>TSX FPP20 | TSX P•7455<br>TSX FPP20 | |
| **ISAway** | ISA Bus | | | |
| **PCIway** | PCI bus | | | |
| **Modbus** | TSX SCP11• | TER CPU port | TSX SCM22• | JB cards• |
| **Modbus Plus** | TSX MBP100 | TSX MBP100 | | |
| **USB** | Built-in channel | | | |
| **USB Fip** | Built-in channel | | | **USB Fip** |

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

## Different Access Modes for Server or Simulator

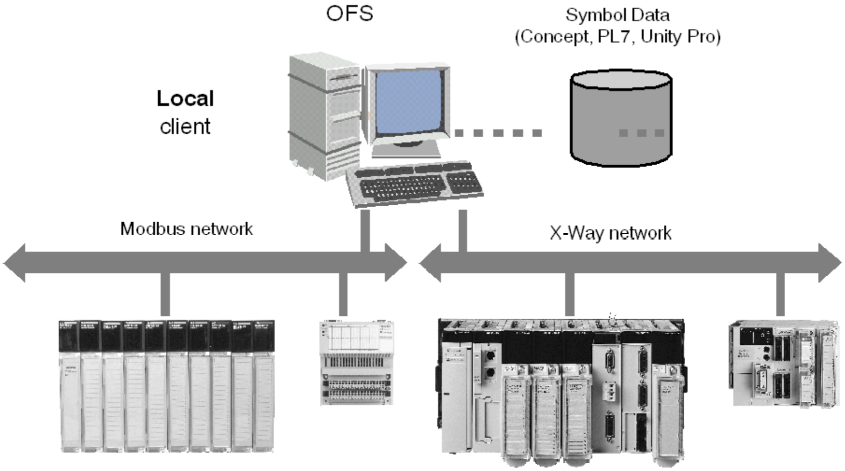
### Description

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.

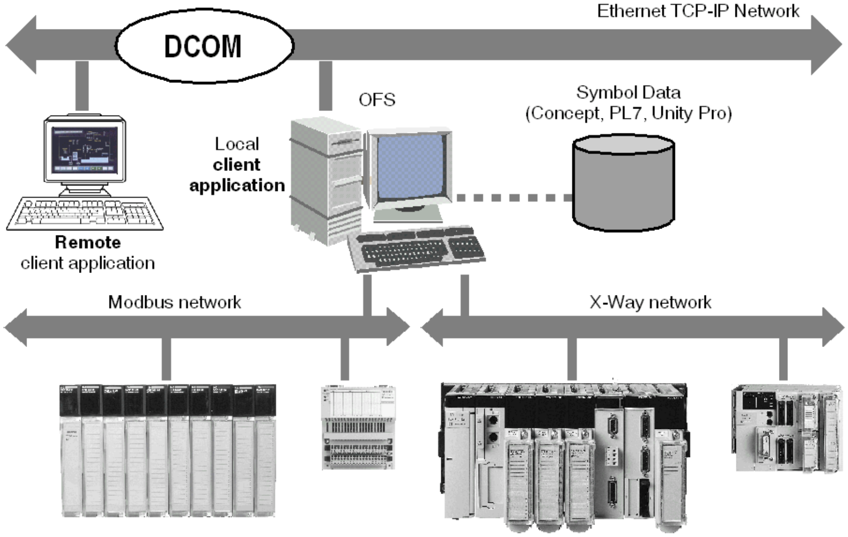
### Local access

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:

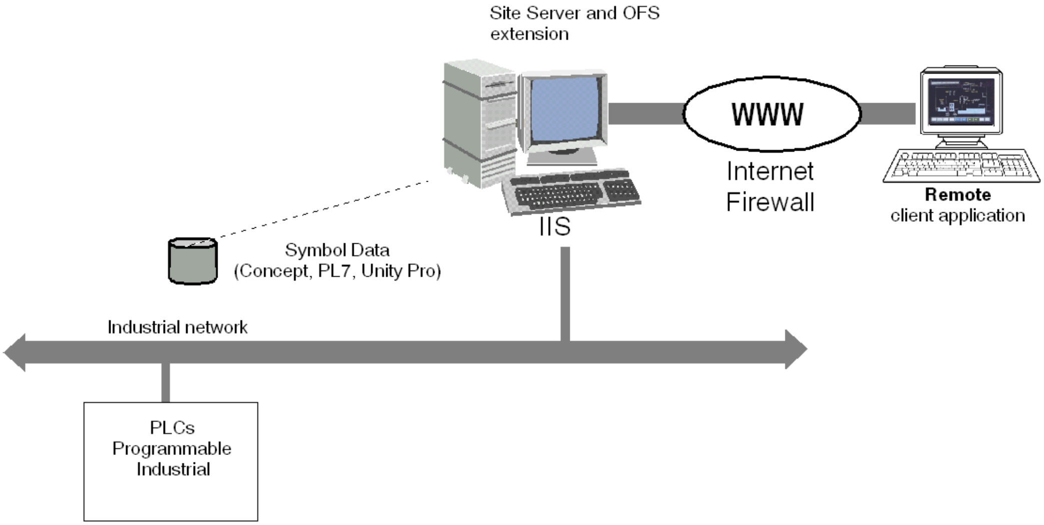
**NOTE:** DCOM *(see page 52)* must be configured correctly before launching remote operation.

### 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 61)* 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.

### OPC: NET API

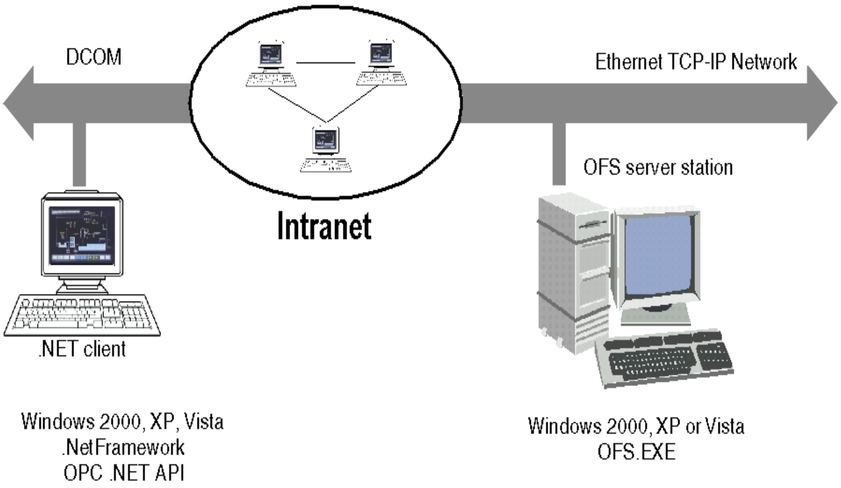
This API was developed to provide interoperability between existing OPC specifications and applications developed in the .NET environment. The API supports servers based on COM and SOAP/XML via the same set of interfaces.

## 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 81)* 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

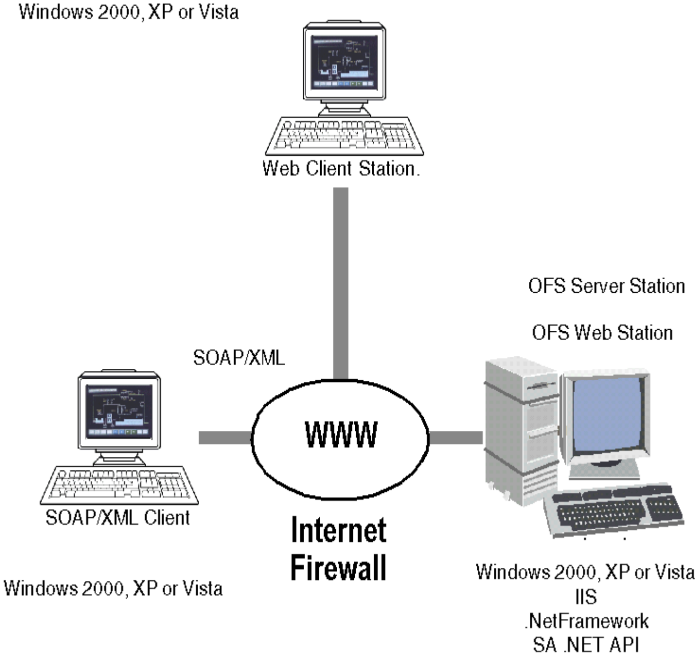
## Description

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 81)* and IIS *(see page 61)* 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:



# Installing the OFS product

II

## Aim of this section

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

## What's in this Part?

This part contains the following chapters:

# Contents of the OPC Factory Server Product

# 2

## OFS Contents

### Contents of the product

The OFS product has 2 CDROM including:

- Installation instructions,
- Different drivers.

The OFS CD-ROM includes:

- The OFS Server,
- The OFS Manager,
- the OFS Configuration Tool,
- Documentation (English/French/German languages),
- Sample symbol tables, Sample applications,
- Two OPC test clients (Win32 and .NET),
- A Web client (for accessing page view, data editor and server status),

**NOTE:** the OFS product does not contain any cable for communication between the PC and the PLC.

# Hardware and software configuration of the OPC Factory Server product

# 3

## Hardware and Software Configuration

### Description of the Configuration

The OFS product requires a PC "Wintel" platform: Intel x86 monoprocessor with one of the following Microsoft Windows 32-bit operating systems:

- Windows Vista
- Windows XP professional,
- Windows 2000 professional upgraded with Service Pack 1 (or later).

To determine which version of Windows and which service pack is installed on your PC, proceed as follows:

- **Windows 2000 & XP professional:**
  Open "Parameters\Configuration Panel\System". The system version is displayed under the general folder, and must be at least: version 5.00.2195 (Service Pack 1) for Windows 2000.

Your PC needs to meet the following hardware requirements to run the OFS software:

| | **Microsoft 2000 & XP** | | **Microsoft Vista** | |
|---|---|---|---|---|
| | **Frequency** | **RAM** | **Frequency** | **RAM** |
| Minimum configuration | 566 MHz | 128 MB | 1.2 GHz | 512 MB |
| Nominal configuration | 850 MHz | 256 MB | 3 GHz | 1 GB |

**NOTE:** In Unity environment, it's better to use the configuration recommended by Unity Pro.

Minimum resolution: 1024x768 pixels.

Recommended resolution: 1280x1024 pixels.

# Product installation procedure

# 4

## Aim of this Chapter

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

## What's in this Chapter?

This chapter contains the following topics:

## Preparing to Install the OFS Product

### Basics

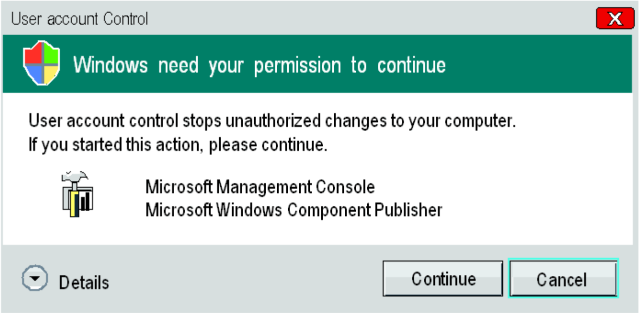
**NOTE:** In Windows Vista, XP or 2000, it is necessary to have administrator rights.

**NOTE:** Two versions of OFS 3.33 are offered:

- S or small that is used to install the following options:
  - OPC Data Access Station (number of items limited to 1000),
  - OPC Data Access Remote Client,
  - .Net Interface.
- L or large that is used to install the following options:
  - OPC Data Access Station (no restriction on the number of items),
  - OPC Data Access Remote Client,
  - .Net Interface,
  - OPC XML Server,
  - 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 the necessary environment for the development and connection of OPC .Net, as well as a OPC .Net 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 you get this kind of message when installing OFS 3.33 on a Windows Vista computer, click Continue:



User account Control

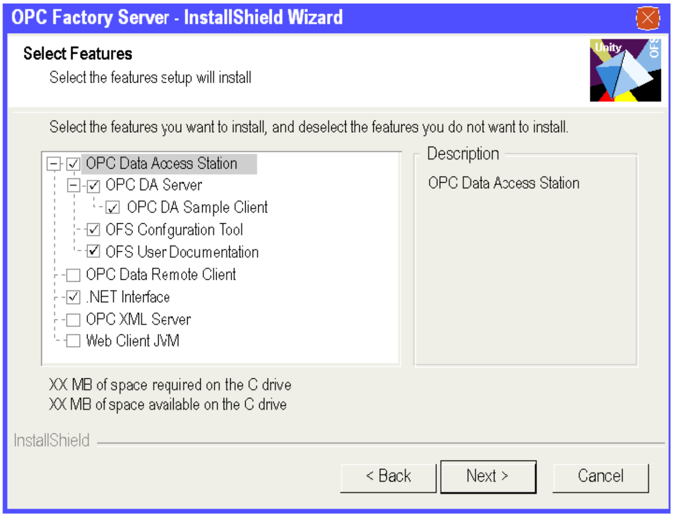
Windows need your permission to continue

User account control stops unauthorized changes to your computer.
If you started this action, please continue.

Microsoft Management Console
Microsoft Windows Component Publisher

## Launching the Installation

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

| Step | Action |
|---|---|
| 1 | Put the CD in the drive. The installation automatically starts. Just follow the on-screen messages. |
| 2 | In the **Customer Information window**, enter the part number and the serial number provided on the CD box, or enter **DEMO** in part number field to obtain an evaluation version of the product.<br>***Result***: The product install selection screen appears :<br><br>OPC Factory Server - InstallShield Wizard<br>Select Features<br>Select the features setup will install<br>Select the features you want to install, and deselect the features you do not want to install.<br>☑ OPC Data Access Station<br>☑ OPC DA Server<br>☑ OPC DA Sample Client<br>☑ OFS Confguration Tool<br>☑ OFS User Documentation<br>□ OPC Data Remote Client<br>☑ .NET Interface<br>□ OPC XML Server<br>□ Web Client JVM<br>Description<br>OPC Data Access Station<br>XX MB of space required on the C drive<br>XX MB of space available on the C drive<br> |

| Step | Action |
|---|---|
| 3 | Among the available features and depending on the customer information, select the ones you want to install and click on **Next.**<br>**Note**: You can display the description of a feature as well as its components by putting the focus on it.<br>**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 145)* 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 351)* (*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 150)* 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 87)*.
- 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.
- Performing an uninstall does not affect the parameter settings data stored in the registry.
- 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 options:

- **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 150)* (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 145)* (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 options:

- **OPC .Net API**: SDK (software development kit), from the OPC foundation, used to develop a .Net Client or an OPC XMLDA Client (under SOAP/XML),
- **.NET Sample Client**: this is an OFS test utility client, operating in a .NET environment.
- **.Net framework (verification)** : the .Net framework installation is proposed in version 1.1 if that version is not already supported by the machine. This version is the one that was used when checking the correct operation of the .Net station.

## 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 CD-ROM. 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 61)*.

The installation program then proposes:

- **.Net framework**: the .Net framework installation is proposed in version 1.1 if that version is not already supported on the machine. This version is the one that was used to check the correct operation of the Web station.
- **OFS Web Site**: provides data via tables to the Web Client such as the editor and the data viewer and the server status page.
- **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 63)* IIS according to the site’s security and access.

**NOTE:** In any case, IIS must be installed before .Net framework. If .Net framework 1.0 or 1.1 is already present on the machine when installing IIS, you must first uninstall it.

**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)** : the installation of the JVM is proposed in version 1.5.0 if JVM is not already installed or if the installed version is earlier than 1.4.2.
- **Internet Explorer (verification)**: if no Internet Explorer is installed on the machine, or if the version is too old (earlier than IE5.1), the installation program asks the user to update the machine to the required level.

# Driver Installation

## Description

Drivers are installed using their specific CD-ROM.

The OFS server can use drivers already installed on your machine provided that they are not too old. The compatibility table below indicates the minimum version you should have installed to be sure that OFS will operate properly. Use of OFS with older versions is neither supported nor guaranteed.

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

Compatibility Table:

| **Drivers** | **Minimum version** depending on operating system | |
|---|---|---|
| | Windows 2000 / XP | Windows Vista 32 |
| Uni-Telway | 1.10 | 2.0 |
| FIPway FPC10 | 1.4 | Not supported |
| FIPway PCMCIA | 1.2 | Not supported |
| ISAway | 1.2 | Not supported |
| Ethway | 1.5 | Not supported |
| X-Way / TCP/IP | 1.11 | 2.0 |
| PCIway PCX 57 | 1.1 | 2.0 |
| USB HE terminal port PCX 57 | 1.2 | 2.0 |
| Modbus Serial | 1.8 | 2.0 |
| USB Fip | 1.0 | 2.0 |

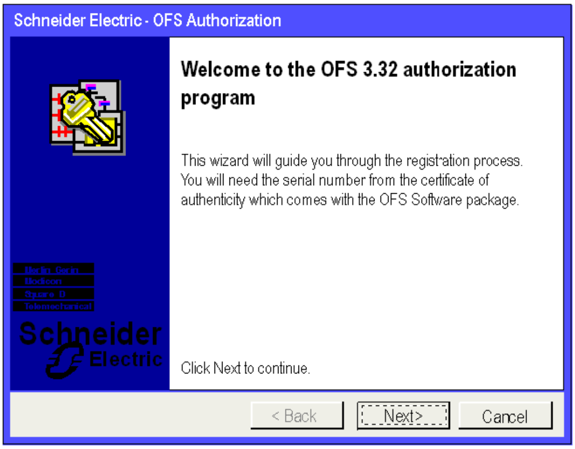
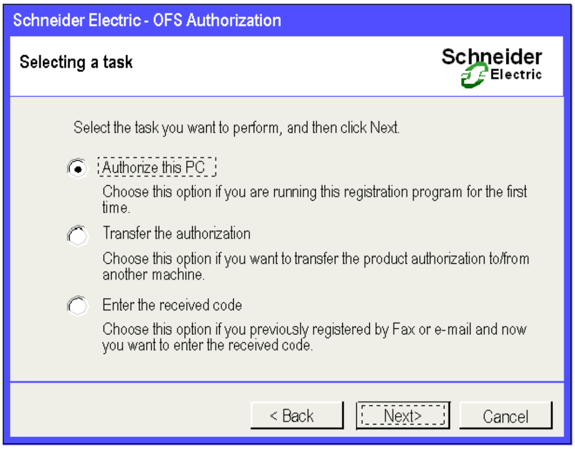
# Registration

5

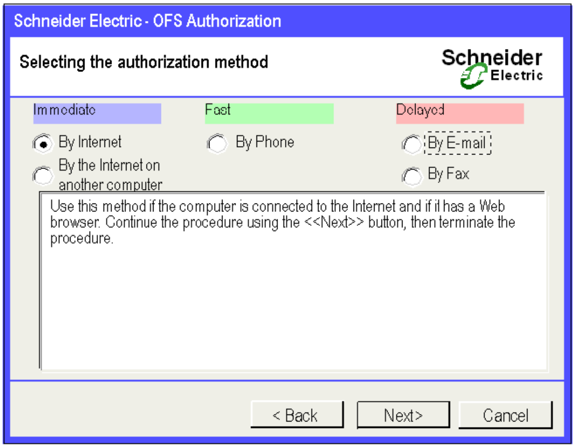
## OFS Authorization

### 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:<br><br>Schneider Electric - OFS Authorization<br>**Welcome to the OFS 3.32 authorization program**<br>This wizard will guide you through the registration process. You will need the serial number from the certificate of authenticity which comes with the OFS Software package.<br>Click Next to continue.<br> |
| 2 | Click **Next** , a new window is displayed with 3 choices:<br>● the PC's authorization to know the various ways to make an authorization request,<br>● the transfer of the authorization to or from a computer storage device,<br>● the entry of the code after receiving it.<br>The window below displays the 3 options:<br><br>Schneider Electric - OFS Authorization<br>**Selecting a task**<br>Select the task you want to perform, and then click Next.<br>● Authorize this PC<br>Choose this option if you are running this registration program for the first time.<br>○ Transfer the authorization<br>Choose this option if you want to transfer the product authorization to/from another machine.<br>○ Enter the received code<br>Choose this option if you previously registered by Fax or e-mail and now you want to enter the received code.<br> <br>Click **Next**. |

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>3</td><td>
By choosing the 'Authorize this PC' option in step 2, a new window is displayed. This window offers 5 ways to request the authorization:



Selecting the authorization method

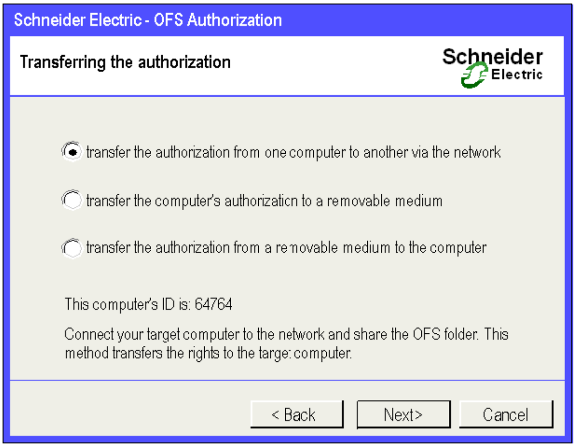
Immediate
- ● By Internet
- ○ By the Internet on another computer

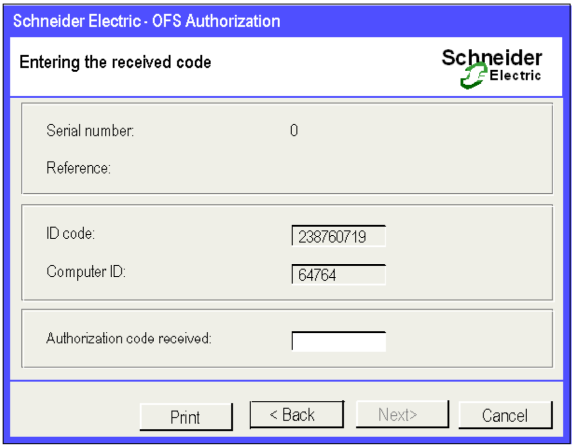
Fast
- ○ By Phone

Delayed
- ○ By E-mail
- ○ By Fax

Use this method if the computer is connected to the Internet and if it has a Web browser. Continue the procedure using the &lt;&lt;Next&gt;&gt; button, then terminate the procedure.

Click one of the methods according to your requirements (immediate, fast, or with a delay) then click ‘Next’ and follow the remaining procedures.
</td></tr>
</table>

| Step | Action |
|---|---|
| 4 | By choosing the ‘Transfer the authorization’ option in step 2, a new window is displayed.<br>The window below offers 3 types of authorization transfer:<br>● transfer the PC’s authorization to a PC on the network,<br>● transfer the PC’s authorization to a removable medium,<br>● transfer the PC’s authorization from a removable medium to this PC.<br>**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.<br>**Note**: The removable medium must not be write-protected.<br><br>Schneider Electric - OFS Authorization<br>Transferring the authorization<br>● transfer the authorization from one computer to another via the network<br>○ transfer the computer's authorizaticn to a removable medium<br>○ transfer the authorization from a re novable medium to the computer<br>This computer's ID is: 64764<br>Connect your target computer to the network and share the OFS folder. This method transfers the rights to the targe: computer.<br> |

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

# Workstation setup

# III

## Aim of this section

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

## What's in this Part?

This part contains the following chapters:

# Workstation Configuration

# 6

## Aim of this Chapter

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

## What's in this Chapter?

This chapter contains the following sections:

# 6.1 Configuration of stations for a WIN32 environment

## 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 using Windows 2000

**Server (Windows 2000)**:

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.

| Step | Action |
|---|---|
| 1 | Start DCOMCnfg.exe located in the Winnt\System32 directory. |
| 2 | In the **Default Properties** tab, check that:<br>● the option**Enable Distributed COM on this computer** is selected,<br>● the field **Default Authentication Level** is set to **Connect**,<br>● the field **Default Impersonation Level** is set to **Identify**. |

| | |
|---|---|
| 3 | In the Application tab, select the folder **Schneider-Aut OPC Factory Server** in the list, then click on **Properties**. The dialog box **Schneider-Aut OPC Factory Server Properties** will appear.<br>● click on the **Identity** tab. The option **The interactive user** should be selected. The **This user** option with an appropriate password should be selected if nobody is logged on to the server or if the session is modified or interrupted for no reason,<br>● click on the **General** tab, the authentication level should be set to **Default**<br>● click on the **Lcation** tab, the option **Run application on this computer** should be selected.<br>● in the **Security** tab, select **Use custom access permissions**.<br>  ● click on **Modify**, the **Access Permission** dialog box appears.<br>  ● click on Add, add the users, then authorize access (the SYSTEM, INTERACTIVE and NETWORK users should be in this list, others such as Everyone can be added),<br>  ● click on OK to close the dialog boxes.<br>● in the Security tab, select Use Custom Launch Permissions.<br>  ● click on **Modify**, the **Launch Permission** dialog box appears.<br>  ● Click on **Add**, add the users, then authorize them to start up the server (the SYSTEM, INTERACTIVE and NETWORK users should be in this list, others such as Everyone can be added),<br>  ● click on OK to close the dialog boxes. |

**NOTE:** If the client and the server do not belong to the same 2000 domain or if there is no 2000 domain, remember that identical users with identical passwords must be created on both machines (note case sensitivity).

**Client (Windows 2000):**

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 | Start DCOMCnfg.exe located in the c:\Winnt\System32 directory. |
| 2 | In the **Default properties** tab, check that:<br>● the option **Enable Distributed COM on this computer** is selected,<br>● the field **Default Authentication Level** is set to **Connect**,<br>● the field **Default Impersonation Level** is set to **identify**. |
| 3 | In the **Default 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. |

## Configuration using Windows XP-SP2 or Vista

**Server (Windows XP-SP2/Vista)**:

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 a XP-SP2 or Vista 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 on 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 **DCOMCNGF**. |
| 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:<br>● **Enable Distributed Com on this computer** must be checked,<br>● **Default Authentication Level** is set to **Connect**,<br>● **Default Impersonation Level** is set to **Identify**. |
| 5 | Click the **COM Security** tab. |
| 6 | Click the **Edit Defaults** in **Access Permissions**. |
| 7 | Click on **Add**, enter **Everyone** local account then click on **OK**. |
| 8 | In the **Access Permission** dialog, check that permissions for **Everyone** are<br>● Local Access: **Allow** checked,<br>● Remote Access: **Allow** checked. |
| 9 | Click on **OK** to close the dialog box window. |
| 11 | Click on **Edit Defaults** in **Launch and Activation Permissions**. |
| 12 | Click on **Add**, enter **Everyone** local account then clock on **OK**. |

| | |
|---|---|
| 13 | In the **Launch and Activation Permissions** dialog, check that permissions for **Everyone** are<br>● Local Launch: **Allow** checked,<br>● Remote Launch: **Allow** checked,<br>● Local Activation: **Allow** checked,<br>● Remote Activation: **Allow** checked. |
| 14 | Click on **OK** to close the dialog box window. |
| 15 | Click **Edit Limits** in **Access Permissions**. |
| 16 | In the **Access Permission** dialog, check that permissions for **ANONYMOUS LOGON** are<br>● Local Access: **Allow** checked,<br>● Remote Access: **Allow** checked. |
| 17 | Click on **OK** to close the dialog box window. |
| 18 | Click **Edit Limits** in **Launch and Activation Permissions**. |
| 19 | In the **Launch and Activation Permissions** dialog, check that permissions for **Everyone** are:<br>● Local Launch: **Allow** checked,<br>● Remote Launch: **Allow** checked,<br>● Local Activation: **Allow** checked,<br>● Remote Activation: **Allow** checked. |
| 20 | Click on **OK** to close the dialog box window. |
| 21 | Click on **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 on **Schneider-Aut OPC Factory Server** to display the properties. |
| 3 | Click on the **Location** tab, the option **Run application on this computer** should be selected. |
| 4 | Click on the **Identity** tab. The option **The interactive user** should be selected. |
| 5 | Click on the **General** tab, the authentication level should be set to **Use Default**. |
| 6 | Click on the **Security** tab<br>● the **Launch and activation Permissions** should be set to **Use Default**,<br>● the **Access Permissions** should be set to **Use Default**. |
| 7 | Close the **Properties** dialog with **OK**.<br>**NOTE:** It is recommended to restart the system. |

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

### Client (Windows XP/Vista):

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,<br>● 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:<br>● the **Enable Distributes COM (DCOM) on this computer** option is selected,<br>● the field **Default Authentication Level** is set to **Connect**,<br>● 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 on the following Policy item **DCOM: Machine Access Restrictions in Security Descriptor Definition Language (SDDL) syntax**. |
| 4 | Click on **Edit Security** button to open the **Access Permission** dialog. |
| 5 | Click on **Add**, enter **Everyone** local account then click on **OK**. |
| 6 | In the **Access Permission** dialog, check that permissions for **everyone** are:<br>● Local Access: **Allow** checked,<br>● Remote Access: **Allow** checked. |
| 7 | Click on **OK** to close the dialog box window. |
| 8 | Click on **OK** to close the **Policy Settings** dialog box. |
| 9 | Double click on the following Policy item **DCOM: Machine Launch Restrictions in Security Descriptor Definition Language (SDDL) syntax**. |
| 10 | Click on **Edit Security** button to open the **Launch and Activation Permission** dialog. |
| 11 | Click on **Add**, enter **Everyone** local account then click on **OK**. |
| 12 | In the **Launch and Activation Permission** dialog, check that permissions for **Everyone** are:<br>● Local Launch: **Allow** checked,<br>● Remote Launch: **Allow** checked,<br>● Local Activation: **Allow** checked,<br>● Remote Activation: **Allow** checked. |
| 13 | Click on **OK** to close the dialog box window. |
| 14 | Click on **OK** to close the **Policy Settings** dialog box. |
| 15 | Double click on the following Policy item **Network access: Let Everyone permissions apply to anonymous users**. |
| 16 | Set the value to **Enabled**. |
| 17 | Click on **OK** to close the **Policy Settings** dialog box. |

# 6.2 Configuration of 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.

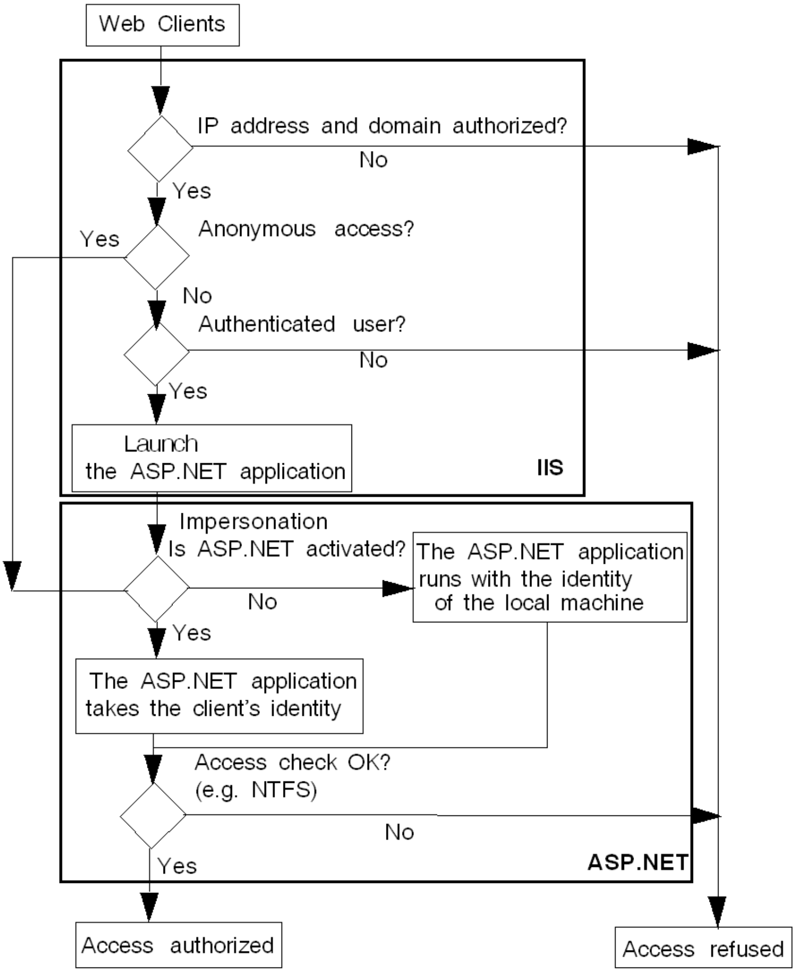
### What's in this Section?

This section contains the following topics:

## Authorization to Connect to a Web Site

### Process

The following figure describes the authorization process for connecting to a Web site:


Web Clients
IP address and domain authorized?
No
Yes
Anonymous access?
Yes
No
Authenticated user?
No
Yes
Launch the ASP.NET application
IIS
Impersonation
Is ASP.NET activated?
The ASP.NET application runs with the identity of the local machine
No
Yes
The ASP.NET application takes the client's identity
Access check OK?
(e.g. NTFS)
No
Yes
ASP.NET
Access authorized
Access refused

## Authentication

Used to identify the user. For example, a user must provide a user name and password. They are checked with an authority (for example, a database or a Windows domain server).

The identification and access control of a client to resources are important components of several distributed applications.

IIS and ASP.NET provide several authentication methods.

**NOTE:** For additional information about Microsoft security, refer to specialized publications.

## Authorization

The process that grants or refuses access to resources for specific users.

## Impersonation

Impersonation enables a server process to run using the client's login information. When the server impersonates the client, all the operations it executes are carried out using that client's login information.

The advantage is that it facilitates the authorization management: NTFS authorizations applicable to the specified account are used. If the user is not authenticated and activity impersonation is activated, then the IUSR account (or the Internet Guest account with very limited authorizations) is used.

## ASP.NET Impersonation

Impersonation is the execution by ASP.NET of code in the context of an authenticated and authorized client. By default, ASP.NET does not use impersonation; it executes code under the account of the ASP.NET process, which is typically the ASP.NET account. This is contrary to ASP behavior, which uses impersonation by default. In IIS, the default identity is the NetrworkService account.

**NOTE:** Impersonation may impact performance and scalability. In general, it is more inefficient to impersonate a client than to make a direct call.

## Delegation

Like impersonation, delegation enables a server process to run using the client's login information. However, delegation goes further in authorizing the server process to call other computers in the client's name, to access remote resources.

## Configuring the IIS component Using Windows 2000 or XP

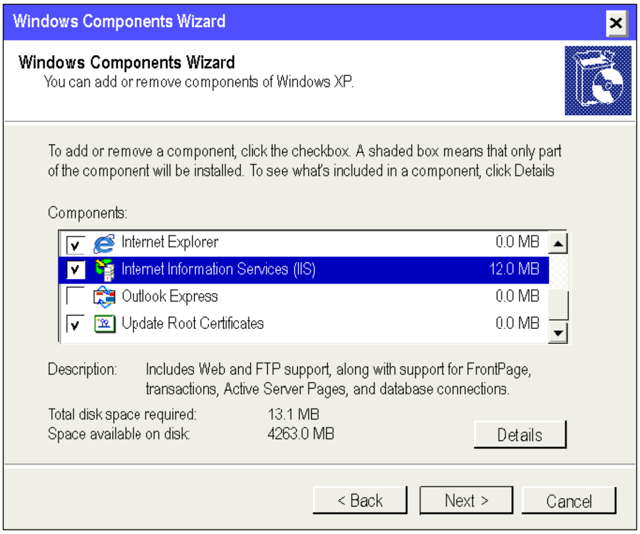
### Introduction

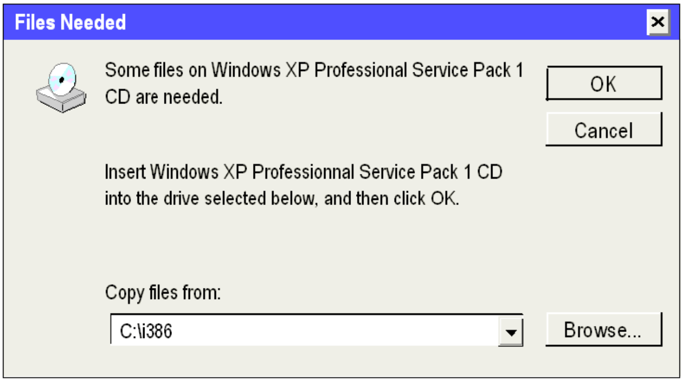
After installing Web station on the machine, the user must validate the Web server functions of the machine via the IIS component. The IIS configuration must be defined manually on the Web Station machine.

**NOTE:** to access the ASP.NET features, IIS must be installed with the latest security updates and before .NET framework

### Installing IIS

The following table describes installation of IIS.

| Step | Action |
|---|---|
| 1 | In the **Start** menu, select **Control Panel**. |
| 2 | Select **Add or Remove Programs**, then **Change or Remove Programs** at the top left of the window. |
| 3 | Select **Windows Components**.<br>**Result**: The following window appears:<br><br>Windows Components Wizard<br>Windows Components Wizard<br>You can add or remove components of Windows XP.<br>To add or remove a component, click the checkbox. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details<br>Components:<br>☑ Internet Explorer 0.0 MB<br>☑ Internet Information Services (IIS) 12.0 MB<br>☐ Outlook Express 0.0 MB<br>☑ Update Root Certificates 0.0 MB<br>Description: Includes Web and FTP support, along with support for FrontPage, transactions, Active Server Pages, and database connections.<br>Total disk space required: 13.1 MB<br>Space available on disk: 4263.0 MB<br> <br>Check **IIS Services** and click **details**. |
| 4 | Check **FTP Services**. |
| 5 | Select **World Wide Web Services**. |
| 6 | Click **Details**. |

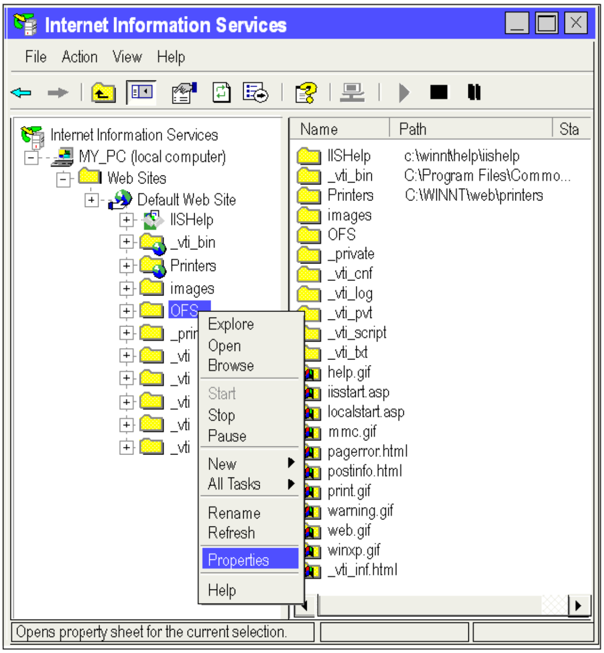
| Step | Action |
|---|---|
| 7 | Click **World Wide Web Services**. |
| 8 | Confirm your choices by clicking **OK** then **OK**. |
| 9 | Then click **Next** to start the IIS installation.<br>During the installation, the following window may appear:<br><br>Files Needed<br>Some files on Windows XP Professional Service Pack 1 CD are needed.<br>Insert Windows XP Professionnal Service Pack 1 CD into the drive selected below, and then click OK.<br>Copy files from:<br>C:\i386<br> <br>But the Windows CD is not mandatory, as the files may already be located in the **c:\i386** folder. |

## Configuring IIS

The following table describes configuration of IIS.

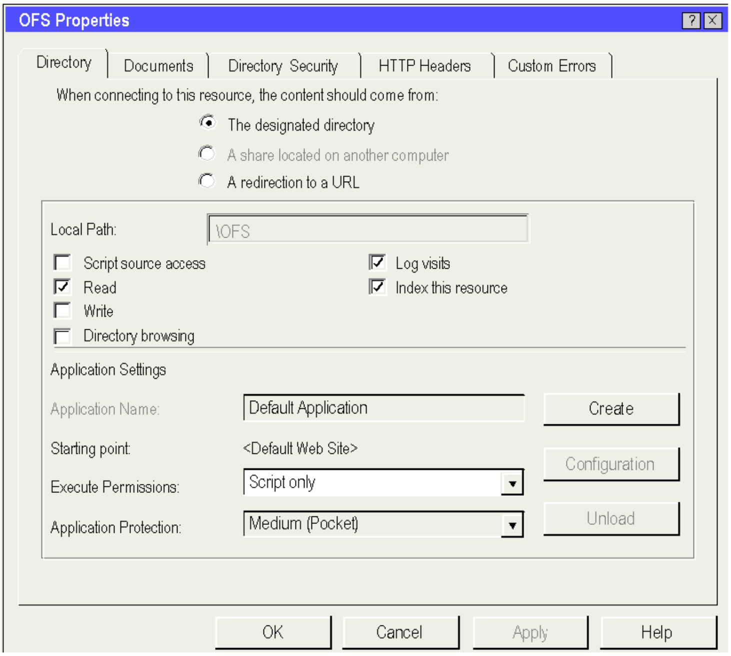
<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Create the &lt;root&gt;\OFS and &lt;root&gt;\OFS\ftp folders (example: c:\inetpub\wwwroot\OFS and c:\inetpub\wwwroot\OFS\ftp).</td></tr>
<tr><td>2</td><td>In the <b>Start</b> menu, select <b>Control Panel</b>.</td></tr>
<tr><td>3</td><td>Click <b>Administrative Tools</b> then choose <b>Internet Services Manager (IIS)</b>.</td></tr>
<tr><td>4</td><td>

In the tree, expand **Web Sites**, **Default Web Site**.
**Result**: the following window appears:



Opens property sheet for the current selection.

</td></tr>
</table>

| Step | Action |
|---|---|
| 5 | Right click on the OFS folder proposed by the user, and choose **Properties**.<br>**Result**: the following window appears:<br><br> OFS Properties<br>Directory<br>Documents<br>Directory Security<br>HTTP Headers<br>Custom Errors<br>Create<br>Configuration<br>Unload<br>OK<br>Cancel<br>Apply<br>Help<br>When connecting to this resource, the content should come from:<br>● The designated directory<br>○ A share located on another computer<br>○ A redirection to a URL<br>Local Path: \OFS<br>□ Script source access<br>☑ Read<br>□ Write<br>□ Directory browsing<br>☑ Log visits<br>☑ Index this resource<br>Application Settings<br>Application Name: Default Application<br>Starting point: <Default Web Site><br>Execute Permissions: Script only<br>Application Protection: Medium (Pocket) |
| 6 | Click the **Create** button to publish the OFS site. |

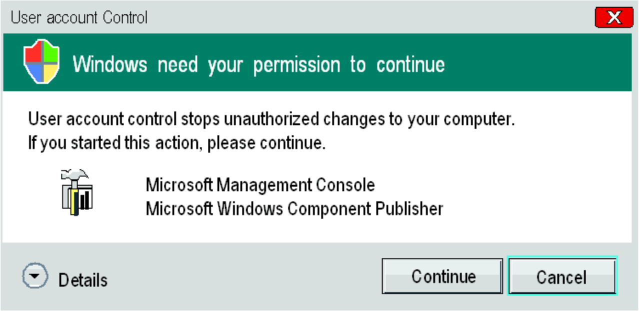
## Configuring the IIS component Using Windows Vista

### Introduction

After installing Web station on the machine, the user must validate the Web server functions of the machine via the IIS component. The IIS configuration must be defined manually on the Web Station machine.

**NOTE:**

- To access the ASP.NET features, **IIS must be installed with the latest security updates and before .NET framework 1.1**. If .NET framework 1.1 is already installed and IIS is not configured yet, it must be uninstalled.
- During the installation, Windows Vista might ask you to confirm your action by displaying the following dialog box. Click **Continue**.



User account Control

Windows need your permission to continue

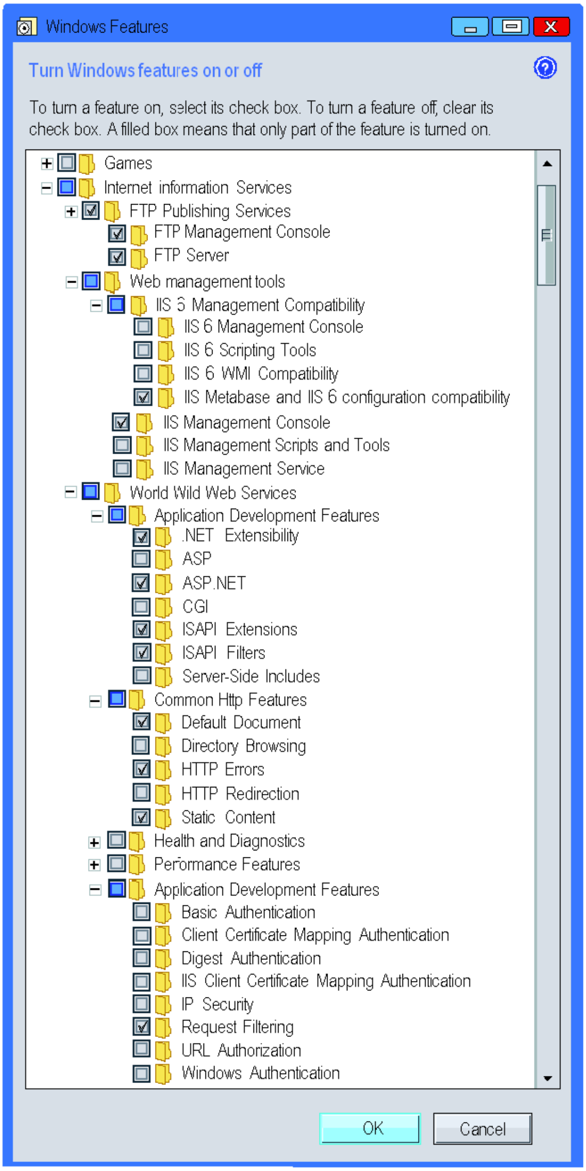
User account control stops unauthorized changes to your computer.
If you started this action, please continue.

Microsoft Management Console
Microsoft Windows Component Publisher

### Installing IIS

Follow this procedure to install the IIS component in Windows Vista environment:

| 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.<br>**Result**: The Turn Windows features on or off window appears: |

| Step | Action |
|---|---|
| 3 | Select the following item:<br><br>Windows Features<br>Turn Windows features on or off<br>To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.<br>+ □ Games<br>− ■ Internet information Services<br>+ ☑ FTP Publishing Services<br>☑ FTP Management Console<br>☑ FTP Server<br>− ■ Web management tools<br>− ■ IIS 3 Management Compatibility<br>□ IIS 6 Management Console<br>□ IIS 6 Scripting Tools<br>□ IIS 6 WMI Compatibility<br>☑ IIS Metabase and IIS 6 configuration compatibility<br>☑ IIS Management Console<br>□ IIS Management Scripts and Tools<br>□ IIS Management Service<br>− ■ World Wild Web Services<br>− ■ Application Development Features<br>☑ .NET Extensibility<br>□ ASP<br>☑ ASP.NET<br>□ CGI<br>☑ ISAPI Extensions<br>☑ ISAPI Filters<br>□ Server-Side Includes<br>− ■ Common Http Features<br>☑ Default Document<br>□ Directory Browsing<br>☑ HTTP Errors<br>□ HTTP Redirection<br>☑ Static Content<br>+ □ Health and Diagnostics<br>+ □ Performance Features<br>− ■ Application Development Features<br>□ Basic Authentication<br>□ Client Certificate Mapping Authentication<br>□ Digest Authentication<br>□ IIS Client Certificate Mapping Authentication<br>□ IP Security<br>☑ Request Filtering<br>□ URL Authorization<br>□ Windows Authentication<br>OK Cancel |
| 4 | Confirm your choices by clicking **OK**. |

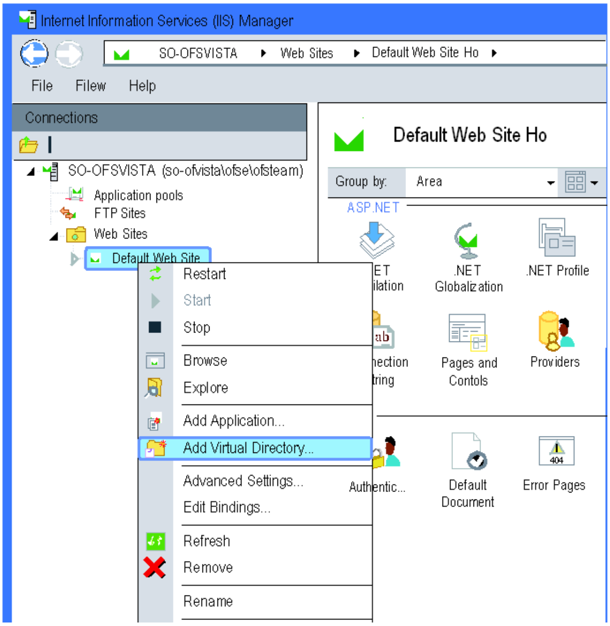
## Enabling ISAPI Extension

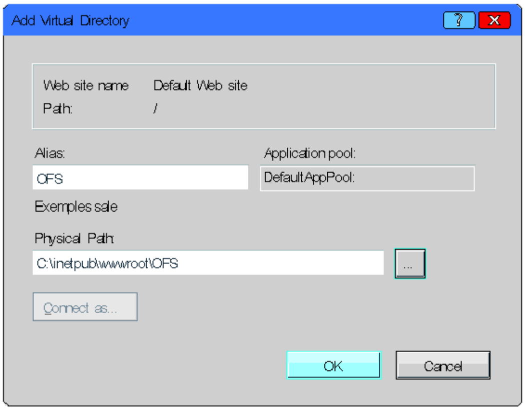
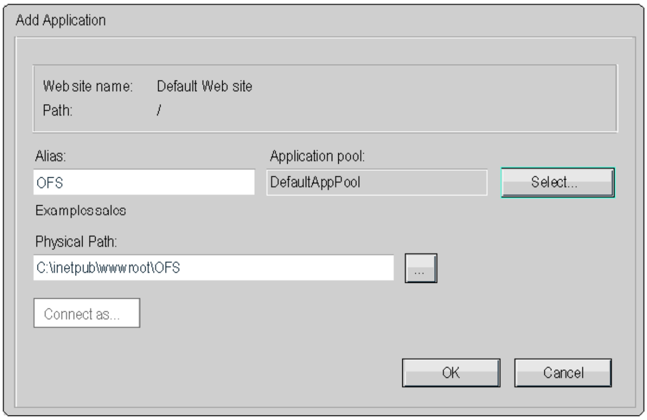
Prior configuring ISS, you must enable the ASAPI extension for ASP.NET 1.1:

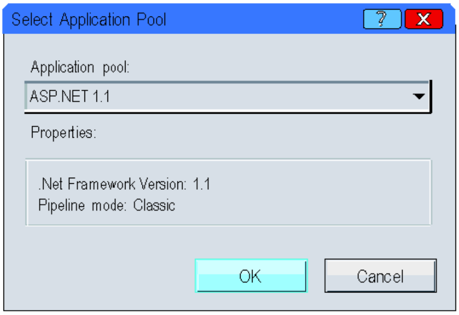
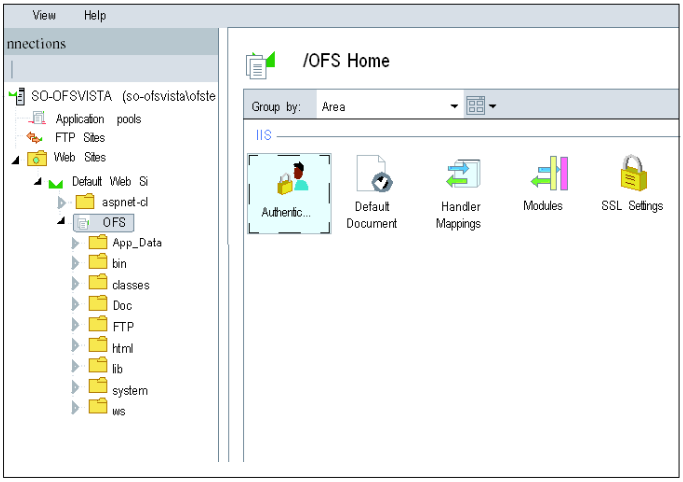
- Click **Start** →**Run...**.
- Type: *%windir%\Microsoft.NET\Framework\v1.1.4322\aspnet_regiis -enable*, where *%windir%* is the environment variable corresponding to the installation folder of Vista.
- Click **OK**.

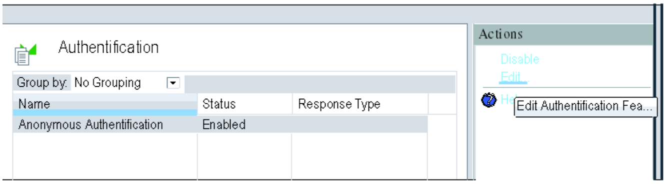
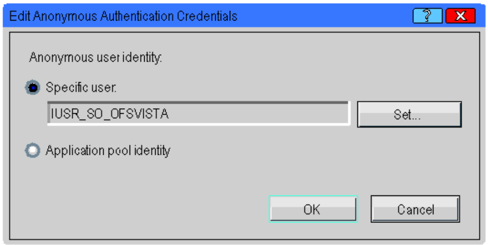
## Configuring IIS

The following table describes configuration of IIS.

| Step | Action |
|---|---|
| 1 | Create the *<root>\OFS* and *<root>\OFS\ftp* folders.<br>**Example:** *c:\inetpub\wwwroot\OFS* and *c:\inetpub\wwwroot\OFS\ftp*). |
| 2 | In the **Start** menu, select **Control Panel**. |
| 3 | Click **Administrative Tools** then select **Internet Information Services Manager (IIS)**.<br>**Result**: The **IIS Manager** window appears:<br> |

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

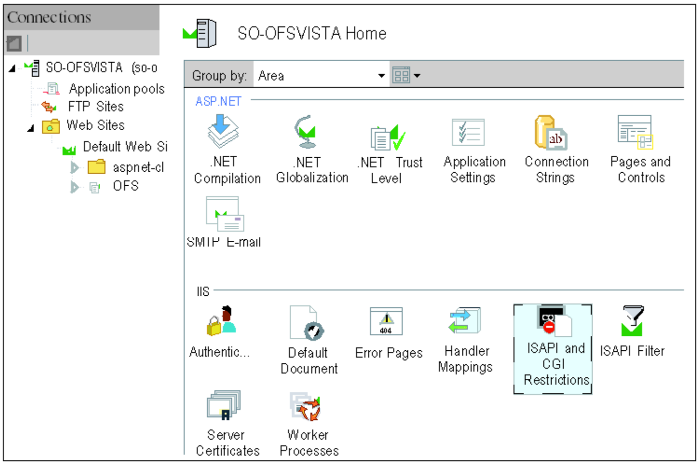
| Step | Action |
|---|---|
| 7 | Click **Select...**.<br>**Result**: The following window appears:<br><br>Select Application Pool<br>Application pool: ASP.NET 1.1<br>Properties:<br>.Net Framework Version: 1.1<br>Pipeline mode: Classic<br> <br>Select **ASP.NET1.1** and click **OK**.<br>Click **OK** in the Add application window. |
| 8 | Click the OFS folder in the IIS manager window (the icon changed and represents now the earth).<br>**Result**: the following window appears:<br><br> View<br>Help<br>nnections<br>SO-OFSVISTA (so-ofsvista\ofste<br>Application pools<br>FTP Sites<br>Web Sites<br>Default Web Si<br>aspnet-cl<br>OFS<br>App_Data<br>bin<br>classes<br>Doc<br>FTP<br>html<br>lib<br>system<br>ws<br>/OFS Home<br>Group by: Area<br>IIS<br>Authentic...<br>Default Document<br>Handler Mappings<br>Modules<br>SSL Settings |

| Step | Action |
|---|---|
| 9 | Double-click the **Authentication icon**.<br>**Result**: the following window appears:<br><br>Authentification<br>Group by: No Grouping<br>Name \| Status \| Response Type<br>Anonymous Authentification \| Enabled<br>Actions<br>Disable<br>Edit<br>Edit Authentification Fea...<br>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.<br>**Result**: the following window appears:<br><br>Edit Anonymous Authentication Credentials<br>Anonymous user identity:<br>● Specific user:<br>IUSR_SO_OFSVISTA<br>Set...<br>○ Application pool identity<br>OK<br>Cancel<br>Click **Set...** and type the name of the low privilege anonymous account created by IIS.<br>Typically the format of the name is *IUSR_WorkstationName*.<br>Click **OK**. |

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>11</td><td>

Click on the computer name in the tree.

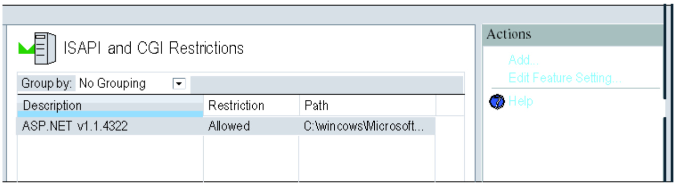
**Result**: the following window appears:



</td></tr>
<tr><td>12</td><td>

In the features view, select **ISAPI and CGI Restrictions**.

**Result**: the following window appears:



| Description | Restriction | Path |
|---|---|---|
| ASP.NET v1.1.4322 | Allowed | C:\wincows\Microsoft... |

In the list of extensions, check that the status of ASP.NET v1.1.xxxx is set to Allowed.

If the status is set to Not allowed:

- Select the ASP.NET v1.1.xxxx extension.
- Click **Edit Feature Settings** on the right side of the window.
- In the **Edit ISAPI or CGI restrictions** window, select the **Allow extension path to execute** checkbox.
- Click **OK**.

</td></tr>
</table>

## FTP Publication of the Site

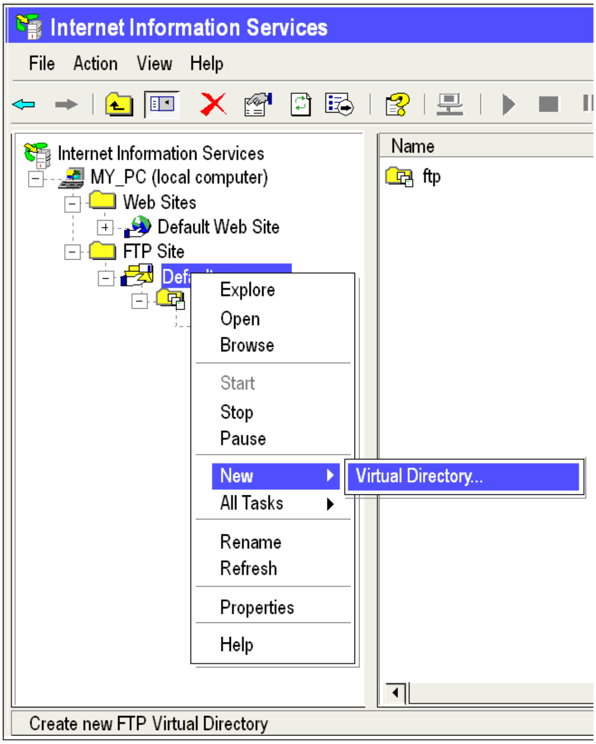
### Introduction

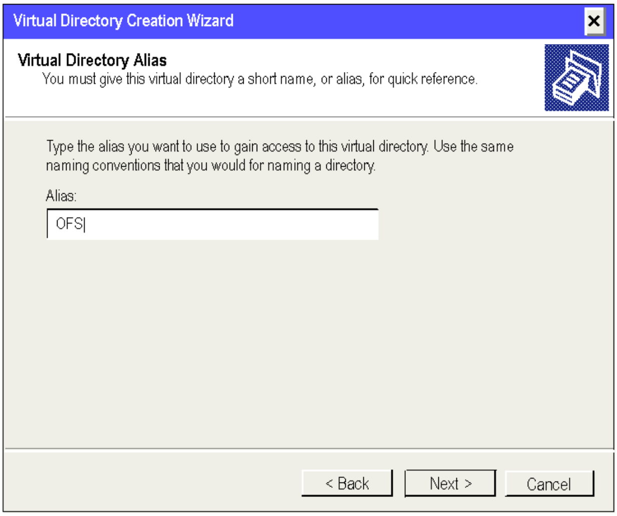
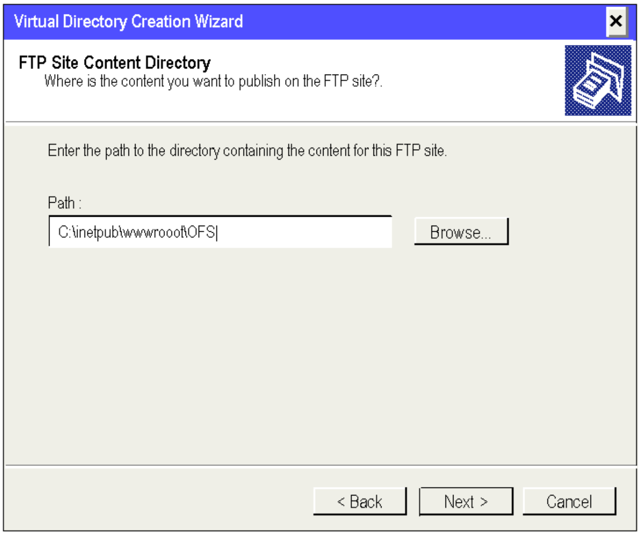
The Web data editor requires that an FTP site with the URL **<host>\ofs\ftp** be published.

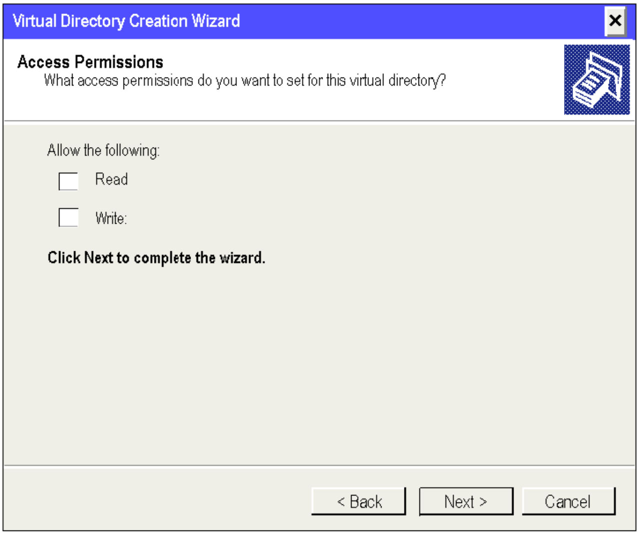
**NOTE:** The following procedure shows you how to publish a FTP site using Windows 2000/XP. If you use Windows Vista, the procedure is identical, only the GUI changes.

### Publish a FTP Site

The following table shows how to publish an FTP site:

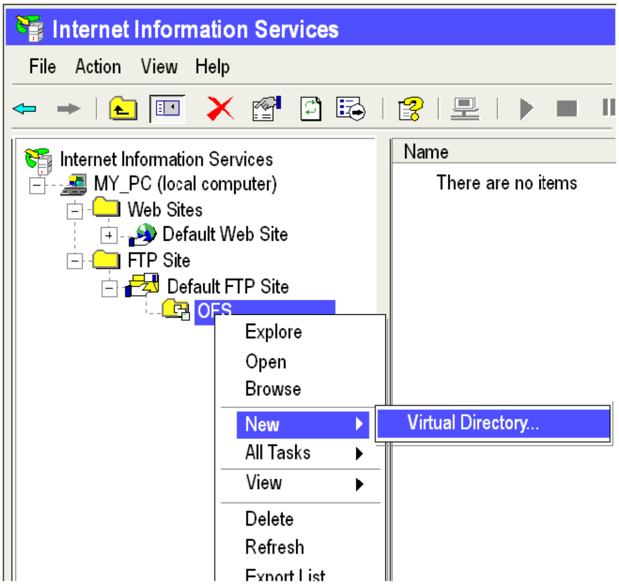
| Step | Action |
|---|---|
| 1 | ● Start **IIS** from the administration tools.<br>● Right click **Default FTP Site**.<br>● Choose **New**, **Virtual Directory**.<br>**Result**: the **IIS** service window is displayed as shown below:<br><br>Create new FTP Virtual Directory |

| Step | Action |
|---|---|
| 2 | In the FTP creation wizard, enter **OFS**.<br><br>Virtual Directory Creation Wizard<br>**Virtual Directory Alias**<br>You must give this virtual directory a short name, or alias, for quick reference.<br>Type the alias you want to use to gain access to this virtual directory. Use the same naming conventions that you would for naming a directory.<br>Alias:<br>OFS<br> <br>Click **Next >**. |
| 3 | Click **Browse...** to specify the OFS path defined in step 1 of the IIS configuration procedure *(see page 67)*.<br>**Example:** *c:\inetpub\wwwroot\ofs*<br><br>Virtual Directory Creation Wizard<br>**FTP Site Content Directory**<br>Where is the content you want to publish on the FTP site?.<br>Enter the path to the directory containing the content for this FTP site.<br>Path :<br>C:\inetpub\wwwrooot\OFS<br> <br>Click **Next >**. |

| Step | Action |
|---|---|
| 4 | Configure the wizard window as shown below:<br><br>Virtual Directory Creation Wizard<br>**Access Permissions**<br>What access permissions do you want to set for this virtual directory?<br>Allow the following:<br>□ Read<br>□ Write:<br>**Click Next to complete the wizard.**<br> <br>Do not check either box in the wizard screen.<br>Thus **ftp://<host\ofs** is not accessible.<br>Click **Next >**. |

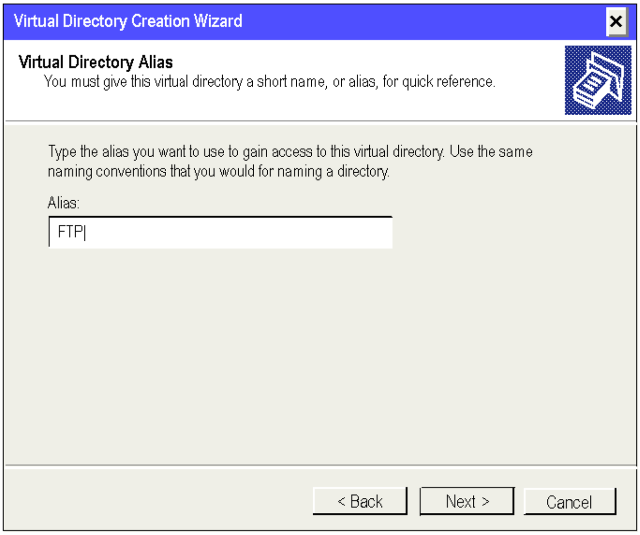
<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>5</td><td>

Restart the wizard from the FTP site **ofs** and create the FTP site.



</td></tr>
<tr><td>6</td><td>

In the FTP creation wizard, enter **FTP**.



Virtual Directory Creation Wizard

**Virtual Directory Alias**

You must give this virtual directory a short name, or alias, for quick reference.

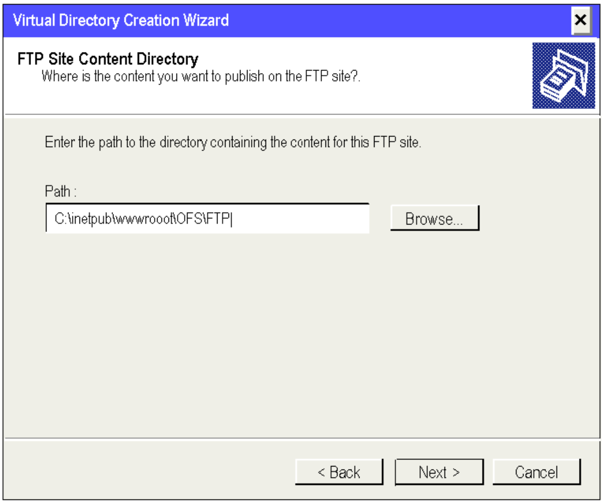
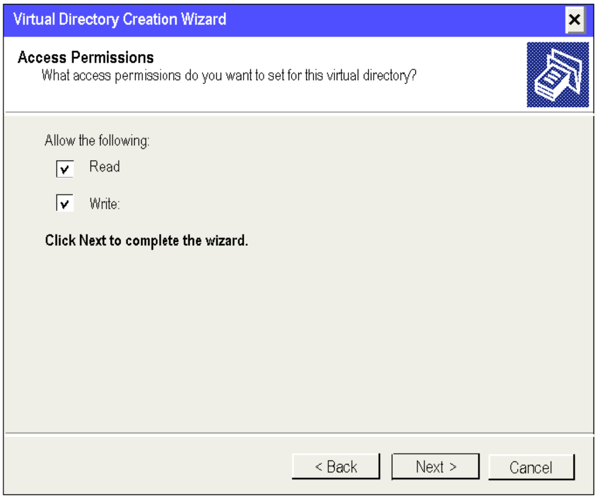
Type the alias you want to use to gain access to this virtual directory. Use the same naming conventions that you would for naming a directory.

Alias:

FTP|

Click **Next >**.

</td></tr>
</table>

| Step | Action |
|---|---|
| 7 | Click **Browse...** to specify the FTP path (e.g. ) defined in step 1 of the IIS configuration procedure *(see page 67)*.<br>**Example:** *c:\inetpub\wwwroot\ofs\ftp*<br><br>Virtual Directory Creation Wizard<br>**FTP Site Content Directory**<br>Where is the content you want to publish on the FTP site?.<br>Enter the path to the directory containing the content for this FTP site.<br>Path : C:\inetpub\wwwrooot\OFS\FTP<br> <br>Click **Next >**. |
| 8 | Configure the next window as shown below:<br><br>Virtual Directory Creation Wizard<br>**Access Permissions**<br>What access permissions do you want to set for this virtual directory?<br>Allow the following:<br>☑ Read<br>☑ Write:<br>**Click Next to complete the wizard.**<br> <br>Check both boxes in the wizard screen.<br>Click **Next >**. |
| 9 | Exit from the wizard. **ftp://<host>\ofs\ftp** is now published. |

## Security of the FTP Site

Microsoft recommends that users access the FTP service using an 'anonymous' account. Anonymous users are represented in security as the IUSR_machinename account for WWW and FTP services. The WWW and FTP services have different roles and security. You can create a different anonymous account for the WWW service and the FTP service, to more clearly differentiate them (for security or auditing purposes).

## The FTP Account

Follow the instructions to create an IUSR_FTP user account and make it a member of the Administrator group.

| Step | Action |
|---|---|
| 1 | Right click on **Default FTP Sites** and select **Properties**. |
| 2 | In the **Properties of the Default FTP Site** window, select the **Security Accounts** tab. |
| 3 | Check **Anonymous access**. |
| 4 | Set User Name and Password in the **Account used for Anonymous Access** dialog box with the user name **IUSR_FTP** and the password. |
| 5 | Check **Allow IIS to control password**. |

## FTP Restriction of Anonymous Access

The IUSR_FTP configuration as an Administrator account cannot be appropriated for ensuring secure access to the server. In this case, configure IUSR_FTP as the member of a group with restricted permissions. Then follow the instructions below to grant access rights to the IUSR_FTP user account.

| Step | Action |
|---|---|
| 1 | Find the file **namespace.zip** located in c:\Inetpub\wwwroot\ofs\ftp. |
| 2 | Right click on this file and select **Properties**. |
| 3 | In the **General** tab, make sure the file is not read-only. |
| 4 | In the **Security** tab, click **Add**. |
| 5 | In the **Select Users or Groups** window, enter <machine name>**\IUSR_FTP**. |
| 6 | Click **OK** to add it to the list. |
| 7 | Select the created user and **grant** him **total control** in the **Security** tab. |
| 8 | Click **OK** to confirm and close the window. |
| 9 | Repeat the steps with the file **password.rde** located in c:\Inetpub\wwwroot\ofs\ftp. |

## Various Types of Access (Windows 2000/XP only)

### Overview

Four minimal Intranet configurations are presented (3 anonymous and one secure access types).

The following are provided for each configuration:

- the ASP.NET Web Service security settings defined in the **web.config** file.
  The **web.config** file, in XML format, is provided when you install Web Station in wwwroot/OFS. To customize the ASP.NET security settings, you must open the file using a text editor or an XML editor and change the values of the 'authentification mode' and 'identity impersonate' items.
- the IIS security level.
  To customize it, open the **Internet Information Service** tool, choose 'Properties' from the pop-up menu of the **OFS\ws\opcxmlda.asmx** file. Use the **File Security** tab to edit the authentication methods.

**NOTE:** The Web.config file and the IIS configuration must match.

### Anonymous IIS Access

Anonymous authentication enables users to access public zones of the Web without asking for a user name or password. Although listed as an authentication method, IIS access does not authenticate the client because the client is not required to provide a user name or password. In this case, IIS provides stored user names and passwords to Windows, using a specific user account.

Advantages:

- improves performance because anonymous authentication does not incur a significant overload,
- does not require the management of individual user accounts,
- if IIS does not verify the password, it can access network resources.

Disadvantages:

- does not authenticate clients individually,
- if IIS does not verify the password, the account can have the local login.

Anonymous IIS access can be achieved in 3 ways:

- an anonymous IIS access with default ASP impersonation,
- an anonymous IIS access with specific ASP impersonation,
- an anonymous IIS access without impersonation.

These three possibilities are described below:

**Anonymous IIS access with default ASP impersonation:**

this configuration is installed by default. It requires that the OFS server be installed on the same machine as the Web service.

- IIS Configuration:
  check anonymous access,
  uncheck Integrated Windows Authentication.
  Set User Name and Password in the **Account used for Anonymous Access** dialog box with the User Name and Password setting **IUSR_machinename**.
- Web.config:
  <authentification mode="none" />
  <identity impersonate="true" />

In this case, ASP.NET impersonates the token it is provided by IIS, which is the anonymous user Internet account (IUSR_machinename).

**Anonymous IIS access with specific ASP impersonation:**

the advantage of this configuration is to provide the application with the same rights as userName.

- IIS Configuration:
  check anonymous access,
  uncheck Integrated Windows Authentication.
- Web.config:
  <authentification mode="none" />
  <identity impersonate="true" userName="domain\user" password="password"/>

User/password authenticates an intranet user (the password appears unencrypted).

In this case, ASP.NET performs a query using the client's credits as a parameter in Web.config.

**Anonymous IIS access without ASP impersonation:**

this configuration requires that OFS be installed on the same machine as the Web Service (if the DCOM Identification Security parameter is set).

- IIS Configuration:
  check anonymous access,
  uncheck Integrated Windows Authentication.
- Web.config:
  <authentification mode="none" />
  <identity impersonate="false"/>

In this case, no authentication or impersonation is performed. ASP.NET performs a query using the credits of the local ASP account.

## Built-in Windows IIS Security Access with ASP Impersonation

The integrated Windows authentication uses a cryptographic exchange with the user's Internet Explorer browser in order to confirm the user's identity.

Advantages:

- the best method for Windows-based intranets,
- can be used in combination with Kerberos (network identification protocol), so you can delegate security credits.

Disadvantages:

- cannot authenticate across a firewall via a proxy, unless you use a PPTP connection,
- does not support delegation to other servers, if NTLM is chosen,
- is supported only by Internet Explorer 2.0 or later,
- Kerberos is only supported by IIS 5.0 or later.
- IIS Configuration:
  check Integrated Windows Authentication.
- Web.config:
  <authentification mode="Windows" />
  <identity impersonate="true" />

In this case, ASP.NET uses Windows authentication in combination with Microsoft IIS authentication. When the IIS authentication is performed, ASP.NET uses the authenticated identity in order to authorize access.

# COM/DCOM Configuration

## Overview

By default, a Web service has very 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 strongly advised to restart the machine after configuring the DCOM security settings.

Two configurations are available:

- an identified OFS access with impersonation,
- an identified OFS access without impersonation.

## 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 | Under Windows 2000, choose the **Distributed COM Configuration Properties/Default properties** or<br>Under Windows XP/Vista, 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 on **Add**, enter the user name of the authorized user,<br>● *(see page 67)* Step 10 with Configuring IIS,<br>● *(see page 78)* with impersonation of ASP identity by default,<br>● *(see page 78)* with specific ASP impersonation,<br>● *(see page 80)* Built-in Windows IIS Security Access with ASP impersonation,<br>then click on **OK**. |
| 8 | Click on **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 54)*.

## Configure an Identified OFS Access without Impersonation

The following table shows how to configure an identified OFS access without impersonation (Windows 2000/XP only).

| Step | Action |
|---|---|
| 1 | Start **DCOMcnfg** from the **Start** menu, then choose **Run**. |
| 2 | Under Windows 2000, choose the **Distributed COM Configuration Properties/Default properties** or<br>Under Windows XP, 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 on **Modify** in **Access Permissions**. |
| 7 | Click on **Add**, enter the ASPNET user name (see *(see page 78)* without ASP impersonation), then click on **OK**. |
| 8 | Click on **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 ASPNET 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 54)*.

# OFS as an NT service

# 7

## OFS as NT Service

### Description

The NT service available only in Windows XP and 2000. It is used to automatically start OFS server each time the extension is launched and stop it automatically each time before the extension stops.

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.

### Procedure

**NOTE:** NT service is controlled by starting the service tool in **Settings/Control Panel/Administrative Tools/Services** (Windows XP & 2000 or **Settings/Control Panel/Services** (NT).

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

- Configure your server (alias, time-out, etc.) preferably using the "hidden" option in the diagnostic folder.
- Check that the OFS server is not already in use and start the Microsoft dcomcnfg tool. This tool's executable file is: Winnt\system32\Dcomcnfg.exe
  - Using this tool, Select the "Schneider-Aut OPC Factory Server" application
  - Select "Properties"
  - Select "Identity"
  - Check "This User" and enter the name and password of a local Administrator-User.
  - Click on "OK"
  - Click on "OK"
  - Close Dcomcnfg
- Run the OFSService.bat batch file which can be found in the directory which contains the server's executable file.

- Start the Services tool.
  OFservice should appear in the list:
  - select OFservice,
  - The default value is "Manual"

  You can then start OFservice and OFS simply by using "Start".
  The same applies for stopping it using "Stop".
  It can be started automatically using "Startup" set to "automatic".
  Close the Services tool.
- 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:** If you are not logged on to the OFS server, you can connect but you can not run it.

### 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 server's executable file. |
| 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 server's executable file. |
| 3 | Start the DCOMcnfg tool. Select the "Schneider-Aut OPC Factory Server" application, then Properties, then Identity and check "The Interactive user" box. Confirm, close DCOMcnfg and restart the machine. |

# User Guide

# IV

## Overview

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

## What's in this Part?

This part contains the following chapters:

# The OFS configuration tool

# 8

## Overview

This chapter introduces the tool for configuring the OFS product.

## What's in this Chapter?

This chapter contains the following sections:

# 8.1 Introducing the Configuration Tool

### Aim of this Section

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

### What's in this Section?

This section contains the following topics:

# 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.

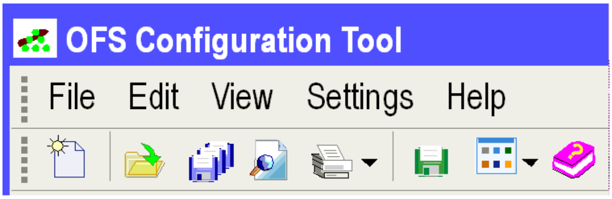
## Configuration Tool Execution

### Description

To launch the OFS Configuration Tool:

- click on 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 tool bar:



The upper part of the window offers a menu bar and a tool bar.

- File menu:
  - The **New Device Alias** option will allow you to create new devices.
  - The **Open Archive** option will allow you to restore a configuration from a backup file. See also the compatibility paragraph *(see page 144)*,
  - 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 a large number of aliases has been declared.
  - The **Save configuration option** will allow you to save all the changes you have done, which will be taken into account after the next server starting up.
  - The **Print Preview** option will allow you to preview your printing.
  - The **Print** option will allow 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 will appear the first time the program is run, which allows existing aliases to be restored. See the Compatibility *(see page 144)* paragraph for further details.

**NOTE:** To retrieve an OFS configuration created with a version of OFS older than version 3.33, you must first run the version 3.33 Configuration Tool and apply the settings before launching the OFS server.

**NOTE:** On Windows XP, an OFS configuration change requires *administrator*'s rights.

# 8.2 Configuration tool

## Introducing the configuration tool

### Presentation

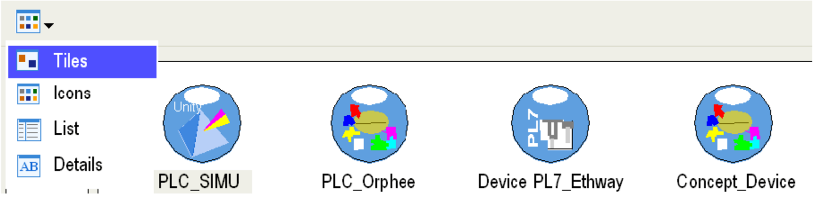
When the configuration tool is opened, one or more aliases are created and the associated properties are set. The client application can create variables on the devices associated with these aliases. In most cases, the Alias definition is sufficient.

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.

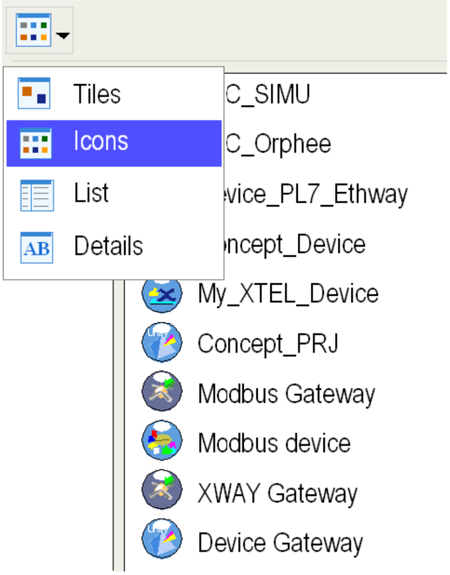
### Tiles display mode

The tiles mode displays as follows:



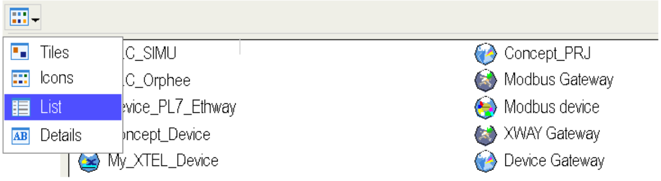
## Icons display mode

The icons mode displays as follows:



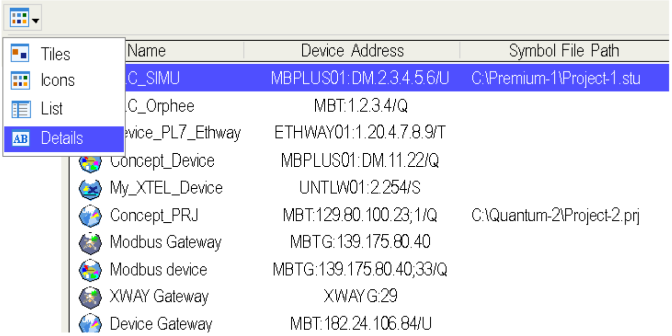
## List display mode

The list mode displays as follows:



## Details display mode

The details mode displays as follows:



| Name | Device Address | Symbol File Path |
|---|---|---|
| C_SIMU | MBPLUS01:DM.2.3.4.5.6/U | C:\Premium-1\Project-1.stu |
| C_Orphee | MBT:1.2.3.4/Q | |
| vice_PL7_Ethway | ETHWAY01:1.20.4.7.8.9/T | |
| Concept_Device | MBPLUS01:DM.11.22/Q | |
| My_XTEL_Device | UNTLW01:2.254/S | |
| Concept_PRJ | MBT:129.80.100.23;1/Q | C:\Quantum-2\Project-2.prj |
| Modbus Gateway | MBTG:139.175.80.40 | |
| Modbus device | MBTG:139.175.80.40;33/Q | |
| XWAY Gateway | XWAY G:29 | |
| Device Gateway | MBT:182.24.106.84/U | |

# 8.3 The Alias folder

### Overview

This section describes alias management.

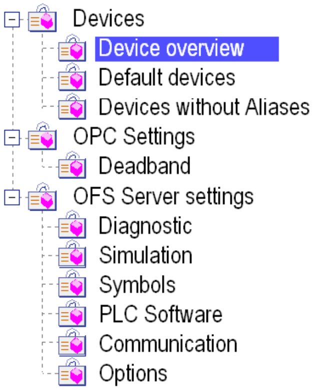
### What's in this Section?

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.<br>**Default devices**: Defines the settings given by default when a new device is created.<br>**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.<br>**Simulation**: Defines the settings for simulation feature.<br>**Symbols**: Defines the settings for symbols files.<br>**PLC Software**: Defines the settings in relation with PLC software.<br>**Communication**: Defines the settings for communication server feature.<br>**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:

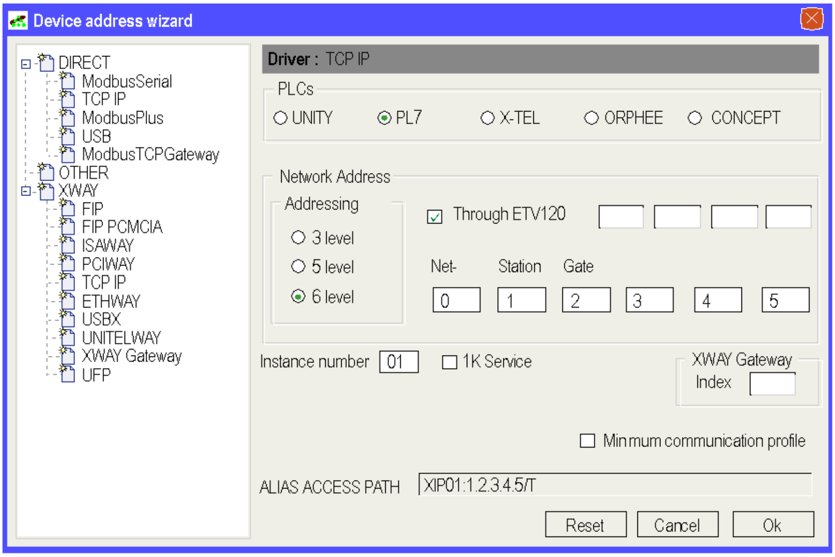
| Field | Description |
|---|---|
| Tree-view | **1st level**: X-Way or Direct addressing (depending on the network being used).<br>The Other family is reserved for future extensions.<br>**2nd level**: Type of network. |
| Alias address | Displays the alias string as the result of the selections. In 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 PLC.
- The X-Way drivers do not support the CONCEPT PLC.

## X-Way parameters

The illustration below shows the X-Way *(see page 319)* addressing modes:


Device address wizard

Driver : TCP IP

PLCs: ○ UNITY ● PL7 ○ X-TEL ○ ORPHEE ○ CONCEPT

Network Address

Addressing: ○ 3 level ○ 5 level ● 6 level

☑ Through ETV120

Net- 0 Station 1 Gate 2 3 4 5

Instance number 01

□ 1K Service

XWAY Gateway Index

□ Minmum communication profile

ALIAS ACCESS PATH XIP01:1.2.3.4.5/T

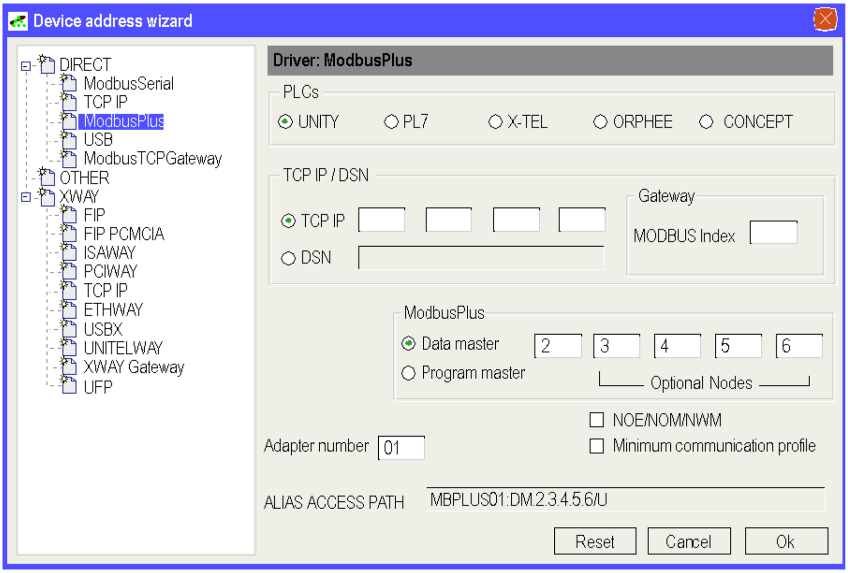
The table below describes the fields relative to X-Way parameters:

| Field | Description |
|---|---|
| PLC's | This area enables you to identify the type of PLC in use:<br>● UNITY ≡ programming with Unity Pro ≡ /U at the end of the alias' address,<br>● PL7 ≡ programming with PL7 ≡ /T at the end of the alias' address,<br>● X-TEL ≡ programming with X-TEL ≡ /S at the end of the alias' address,<br>● 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):<br>If selected, enter an IP address. |
| Addressing levels | Addressing levels for X-Way addresses. See Communication *(see page 319)* 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 319)*.<br>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 111)* as driver. |
| Instance number | One instance per driver installed. Generally equal to 1. Each driver corresponds to a communication card in the PC. |

| | |
|---|---|
| 1K 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.<br>When not checked (by default), the OFS server adjusts the communication parameters optimally, in order to dialogue with the associated device, though performance losses may occur during the adjustment.<br>When checked, the server uses the default communication parameter values, and no adjustment is performed. |

## Direct address parameters

The direct addressing principles are explained in the Communication *(see page 323)* section:



Driver: ModbusPlus

PLCs: ● UNITY ○ PL7 ○ X-TEL ○ ORPHEE ○ CONCEPT

TCP IP / DSN: ● TCP IP ○ DSN

Gateway: MODBUS Index

ModbusPlus: ● Data master ○ Program master

2 3 4 5 6 Optional Nodes

□ NOE/NOM/NWM

□ Minimum communication profile

Adapter number 01

ALIAS ACCESS PATH MBPLUS01:DM.2.3.4.5.6/U

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:<br>● UNITY ≡ programming with Unity Pro ≡ /U at the end of the alias' address,<br>● PL7 ≡ programming with PL7 ≡ /T at the end of the alias' address,<br>● X-TEL ≡ programming with X-TEL ≡ /S at the end of the alias' address,<br>● ORPHEE ≡ programming with ORPHEE ≡ /J at the end of the alias address,<br>● CONCEPT ≡ programming with Concept ≡ /Q at the end of the alias' address. |
| TCPIP/DNS | Modbus TCPIP and Modbus TCPGateway:<br>**TCPIP** : if this option is selected, enter an IP address.<br>**DNS**: If this option is selected, enter device name.<br>**MODBUS Index**: Indicates the Modbus node value of the device.<br>**Gateway**: When ModbusTCPGateway *(see page 109)* is selected, check the box to access the gateway (Modbus TCP <-> Modbus RTU, Modbus TCP <-> Modbus Plus). |
| ModbusSerial | **PLC node**: Modbus RTU only:<br>Premium Unity PLCs are not accessible via the Modbus serial link. |
| Modbus Plus | Modbus Plus only:<br>**Data master**: limited rights (read/write variables).<br>**Program master**: unlimited rights, reserved for the Concept and Unity Pro programming workshop (read/write variables, program modification and configuration).<br>Enter the Modbus address (the first value is compulsory, the others are optional depending on routing levels) |
| NOE / NOM / NWM | This option can be used to increase dynamic performance when a Unity PLC is accessed via an NOE (Ethernet TCPIP) or NOM (Modbus Plus) or NWM (FactoryCast HMI Web Server) communication module and only in these three cases.<br>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.<br>Only the topological objects %MW, %MD and %MF are supported if the box is checked.<br>However, Unity protocol requests can sometimes prove more advantageous to use as they are more economical in  terms of umber of data exchanges with the PLC *(see page 330)*.<br>If your performance is OK, it is recommended that you do not check this box. |
| Minimum communication profile | This option is accessible for PLCs other than Unity.<br>When not checked (by default), the OFS server adjusts the communication parameters optimally, in order to dialogue with the associated device, though performance losses may occur during the adjustment.<br>When checked, the server uses the default communication parameter values, and no adjustment is performed. |

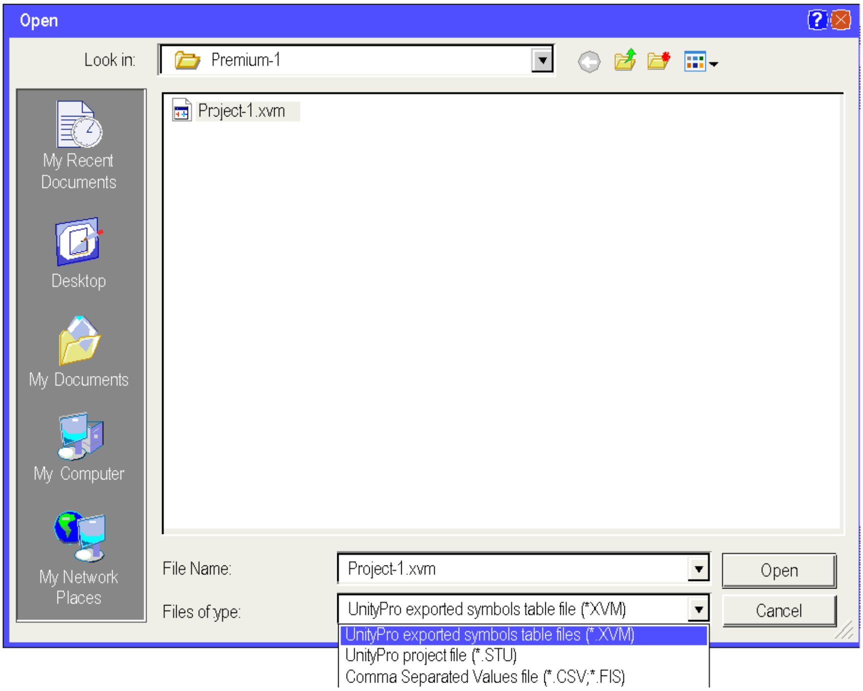
## Associating a Symbols Table File

### 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:



Open

Look in: Premium-1

Project-1.xvm

File Name: Project-1.xvm

Files of type: UnityPro exported symbols table file (*XVM)

My Recent Documents
Desktop
My Documents
My Computer
My Network Places
Open
Cancel
UnityPro exported symbols table files (*.XVM)
UnityPro project file (*.STU)
Comma Separated Values file (*.CSV;*.FIS)

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 100* and see *Symbol management, page 244*) 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 119)*.

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) whether it be under Windows Vista, Windows XP or Windows 2000.

## Link with Concept

### Description

OFS 3.33 supports the following versions of Concept:
- Concept 2.2 SR2,
- Concept 2.5 SR2 (and above).

To install the Concept link, you must simply select the .prj project file (see File table *(see page 100)* and see Symbol management *(see page 244)*) 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 119)*.

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 in Windows Vista/XP/2000 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, simply 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 envisage using unlocated variables:
- the Quantum executable MUST NOT be a stripped version,
- IEC runtime must be activated on the PLC,
- the Unlocated media option must be checked in the properties page. Otherwise, no access to any unlocated variable will be performed.

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 *(see page 244)* 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:

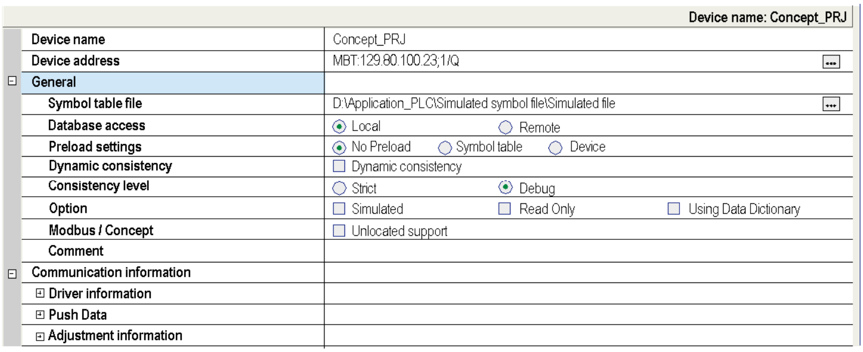
- use of a symbols table file,
- access rights to the variables,
- simulation or real access to the device,
- consistency check between variables and data base,
- waiting time before "Time Out",
- symbol pre-load function for better performance during alias run-time,
- number of channel reservations,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

| | |
|---|---|
| Device name | Concept_PRJ |
| Device address | MBT:129.80.100.23;1/Q |
| ⊟ General | |
| Symbol table file | D:\Application_PLC\Simulated symbol file\Simulated file |
| Database access | ● Local ○ Remote |
| Preload settings | ● No Preload ○ Symbol table ○ Device |
| Dynamic consistency | □ Dynamic consistency |
| Consistency level | ○ Strict ● Debug |
| Option | □ Simulated □ Read Only □ Using Data Dictionary |
| Modbus / Concept | □ Unlocated support |
| Comment | |
| ⊟ Communication information | |
| ⊞ Driver information | |
| ⊞ Push Data | |
| ⊟ Adjustment information | |

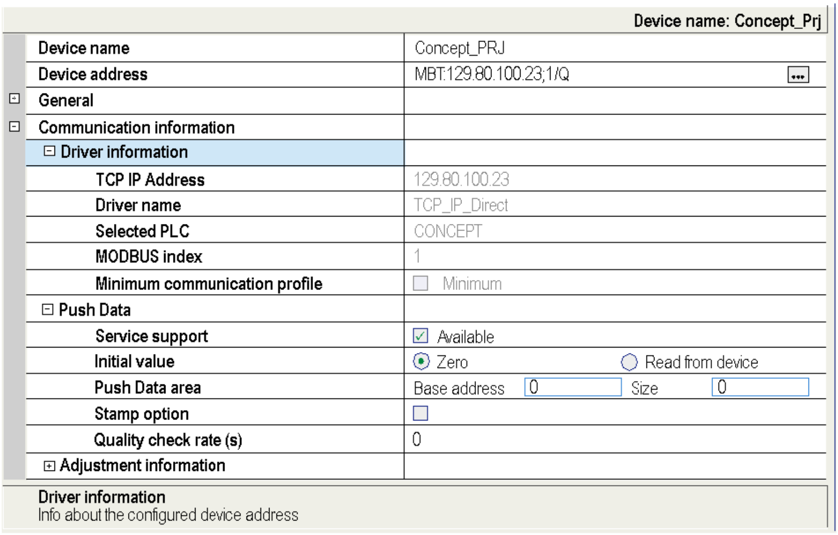
The table below describes the fields relative to the general settings:

| Field | Description |
|---|---|
| Symbol table file | Name and path of the symbols table file. See dedicated section *(see page 100)*. It can be entered and changed here, or directly from the grid.<br>The size of the character string is limited to 255 characters. |
| Database access | *Not available for the devices which can be programmed by X-TEL, ORPHEE and PL7 software workshops.*<br>Allows you to set up the Concept or Unity Pro database in relation to the OFS server:<br>● either the server is on the same machine as the database: Local,<br>● 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:<br>● **None**: by default.<br>● **Symbols table**: pre-loads the symbol table.<br>● **Device**: loads the symbols table if one exists and creates a continuous connection to the device. |
| Dynamic consistency | This option may be used with types .PRJ (Concept) or .STU (Unity Pro) or .XVM (Unity Pro) symbol files.<br>This option is used to define the rule to apply in the case of a detection of an inconsistency *(see page 139)*. |
| 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*.<br>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.<br>● **Strict level**: A variation with the application signature stops the communication.<br>● **Debugging level**: If an 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 135)*.<br>**Read only option**: All variables relating to the device are read only.<br>**Data Dictionary option**: In order to configure the resynchronization of the corresponding device by the OFS server , the ‘Using Data Dictionary’ option must be checked in the device setting. This option is visible in case of TCP IP direct driver and Unity PLC. |

| Field | Description |
|---|---|
| Modbus/Concept | *Specific to the devices programmed using Concept.*<br>Enables support of unlocated variables. This option must be activated 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 102)* section. |
| Comment | Allows to add any comment. |

## Driver and push data settings

The driver and push data settings display this way:



Device name: Concept_Prj

| | |
|---|---|
| Device name | Concept_PRJ |
| Device address | MBT:129.80.100.23;1/Q |
| ⊟ General | |
| ⊟ Communication information | |
| ⊟ Driver information | |
| TCP IP Address | 129.80.100.23 |
| Driver name | TCP_IP_Direct |
| Selected PLC | CONCEPT |
| MODBUS index | 1 |
| Minimum communication profile | □ Minimum |
| ⊟ 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 | |

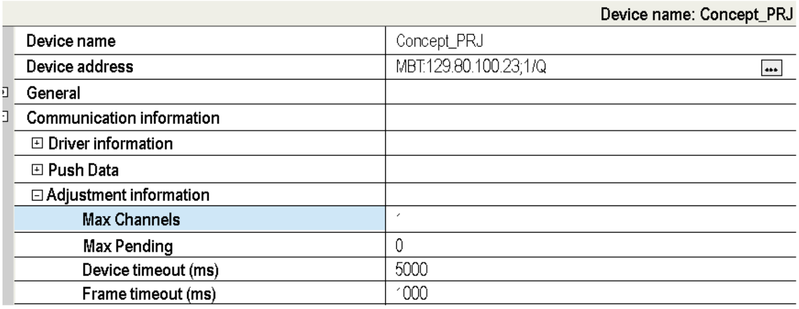
Driver information
Info about the configured device address

The table below 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 122)*. N.B.: if "No Push Data" is selected, other fields are not significant. |

## Adjustment settings

The adjustment settings display this way:



Device name: Concept_PRJ

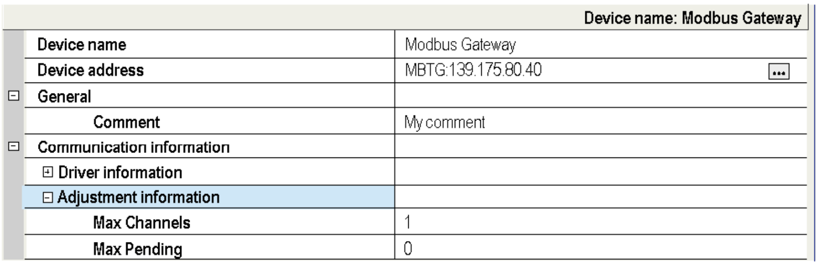
| | |
|---|---|
| Device name | Concept_PRJ |
| Device address | MBT:129.80.100.23;1/Q |
| General | |
| Communication information | |
| ⊞ Driver information | |
| ⊞ Push Data | |
| ⊟ Adjustment information | |
| Max Channels | ´ |
| Max Pending | 0 |
| Device timeout (ms) | 5000 |
| Frame timeout (ms) | ´000 |

The table below describes the adjustment fields:

| Field | Description |
|---|---|
| Max Channels | Number of channels allocated to the device.<br>**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 324)*. |
| 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.<br>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).<br>Range: [3000..32767], at least three times the Frame timeout (or 0 to disable the feature).<br>For more information on this subject, see frame and device timeout *(see page 115)*. |

## Modbus Gateway

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



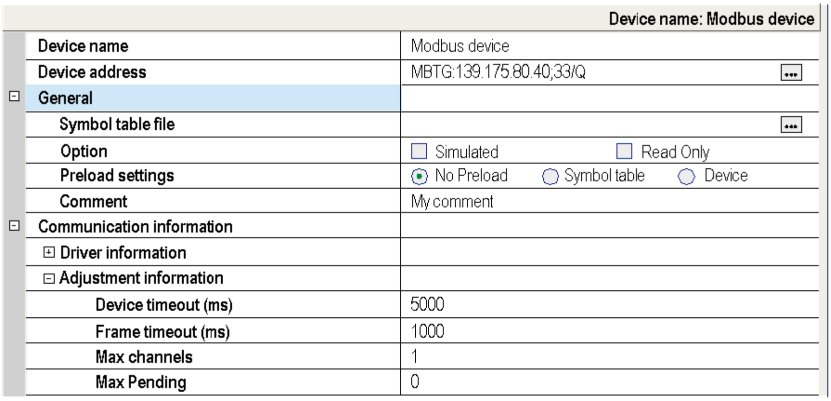
Device name: Modbus Gateway

| | |
|---|---|
| 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.<br>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.<br>**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 324)*. |
| 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.<br>Range: [0..256], 0 is the default value and it indicates that OFScalculates the optimal value. |

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



Device name: Modbus device

| | |
|---|---|
| Device name | Modbus device |
| Device address | MBTG:139.175.80.40;33/Q ... |
| ⊟ General | |
| Symbol table file | ... |
| Option | ☐ Simulated ☐ Read Only |
| Preload settings | ● No Preload ○ Symbol table ○ Device |
| Comment | My comment |
| ⊟ Communication information | |
| ⊞ Driver information | |
| ⊟ Adjustment information | |
| Device timeout (ms) | 5000 |
| Frame timeout (ms) | 1000 |
| Max channels | 1 |
| Max Pending | 0 |

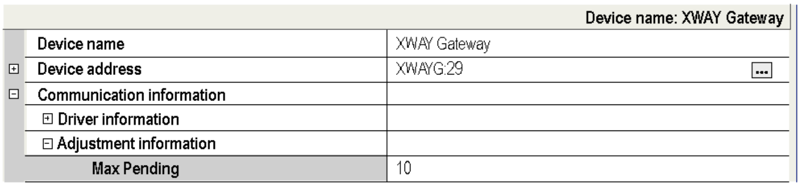
In this example, the alias is associated with a Concept (/Q) project with a **Modbus index** of 33, on a gateway with the IP address 139.175.80.40.

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

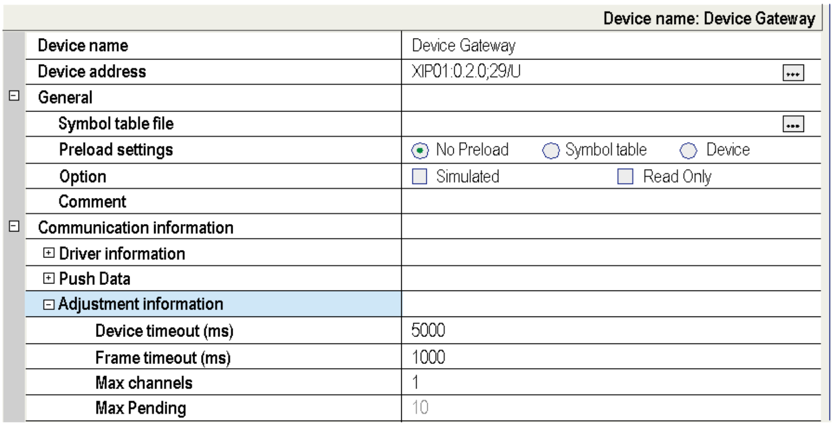


Device name: XWAY Gateway

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

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

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



Device name: Device Gateway

| | |
|---|---|
| Device name | Device Gateway |
| Device address | XIP01:0.2.0;29/U ... |
| ⊟ General | |
| Symbol table file | ... |
| 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 | 10 |

Here, the alias is associated with a Unity (/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.

## Tuning the OFS Network Interface

### Description

The main parameters for tuning the OFS network interface are as follows:

- **Group Min. Rate** (see *Setting the alias properties, page 105*): the value of this parameter should refer to the update rates you plan to use for your OPC groups. If you set the value of this parameter to X, the update rates you can use are X, 2X, 3X .. nX. The rule of the thumb is to set it to X/2 if you plan to use rates X, 2X, 3X,  nX
  **Example**:
  If you plan to use update rates that are 1, 2, 5 seconds, you can set this parameter to 500 ms.
  If you give this parameter too low a value, you waste the PC's processing time.
- **Sampling rate on reception**(see *The Communication folder, page 140* ): this parameter defines the frequency with which the internal entity in charge of receiving the responses will check the network drivers to see if any messages have been received. The rule of thumb is to set it to X/2 if you know that your best device responds within X ms.
  **Example**:
  If the best device responds after 100 ms, give a value of 50 ms to the parameter relating to the protocol (Modbus or X-Way).
  If you give this parameter too low a value, you waste the PC's processing time.
- **Max channels**(see *Setting the alias properties, page 105*, *Specific Items, page 194* and *Multi-Channel Feature, page 324*).

# 8.4 The Device overview folder

### Aim of this section

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

### What's 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.**<br>**Note**: You can use the contextual menu by right mouse clicking on top window to create a new device.<br>***Result***: A new device is created and the configuration tool gives a default name that can be immediately changed or later by a "rename".<br>**Note**: you cannot have two identical names. |
| 2 | Assign a name to your device. |
| 3 | Define the device network address *(see page 96)* which includes the network driver, the type and the device address. |
| 4 | Provide a Symbols table file name *(see page 100)* (optional). |
| 5 | Set the alias properties *(see page 105)* which are related to how the server will behave towards the variables created on that alias.<br>**Note**: A device should be associated with a single and unique alias. If two aliases point to the same device and are used simultaneously, the communication will malfunction. The properties will be the same for both and set to the properties of the alias first used to create an item. Using an alias and accessing the same device directly from the address will have similar results. |

## Additionnal 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 194)* #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 324)* 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 105)* specific parameters. They can also be static (configured using the Configuration Tool) or dynamic (configured using a write specific *(see page 194)* item *(see page 115)* and method.

# 8.5 The Default devices folder

### Aim of this Section

The aim of this section is to describe template management.

### What's in this Section?

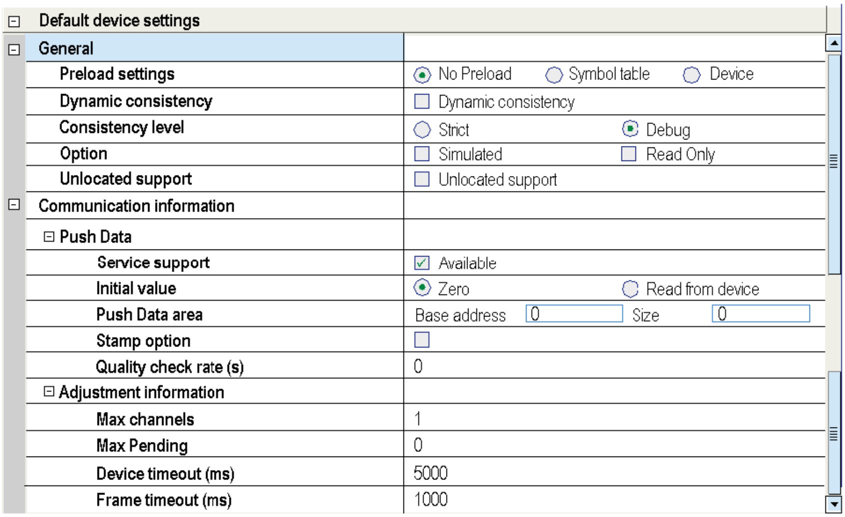
This section contains the following topics:

# 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 | |
| Preload settings | ● No Preload ○ Symbol table ○ Device |
| Dynamic consistency | □ Dynamic consistency |
| Consistency level | ○ Strict ● Debug |
| Option | □ Simulated □ Read Only |
| Unlocated support | □ Unlocated support |
| ⊟ 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 | 1 |
| 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 102)* or the Unity Pro link *(see page 101)* or the Unity Pro file of exported XVM symbols *(see page 247)*.

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.

Thus by using both Concept and OFS or Unity Pro and OFS under Windows Vista, XP or 2000, 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 (this function is available under Windows Vista, XP and 2000).

The automatic loading function can be deactivated (see PLC Software folder *(see page 139)*). In this case, reloading may be performed manually using the OFS Manager *(see page 145)*, 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 *(see page 100)*,
- 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.

## 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 TC/PIP (except ETY 120), Fipway and Ethway networks.

This feature can be enabled and configured PLC by PLC using the device *(see page 105)* 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 105)* properties page).

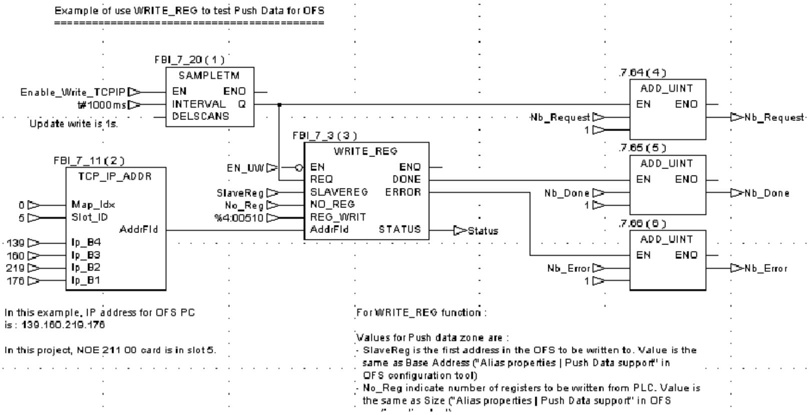
Any number of OPC items can be defined within this range. They are seen as ordinary OPC items.

The device has an option which can be used to send a timestamp *(see page 105)* 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 37h 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:



Example of use WRITE_REG to test Push Data for OFS

Update write is 1s.

In this example, IP address for OFS PC is : 139.160.219.176

In this project, NOE 211 00 card is in slot 5.

For WRITE_REG function :

Values for Push data zone are :
- SlaveReg is the first address in the OFS to be written to. Value is the same as Base Address ("Alias properties | Push Data support" in OFS configuration tool)
- No_Reg indicate number of registers to be written from PLC. Value is the same as Size ("Alias properties | Push Data support" in OFS

Some sample applications are provided on the CD 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).<br>Example: range %MW1000..%MW1500 : base = 1000, size = 500<br>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 CD 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).

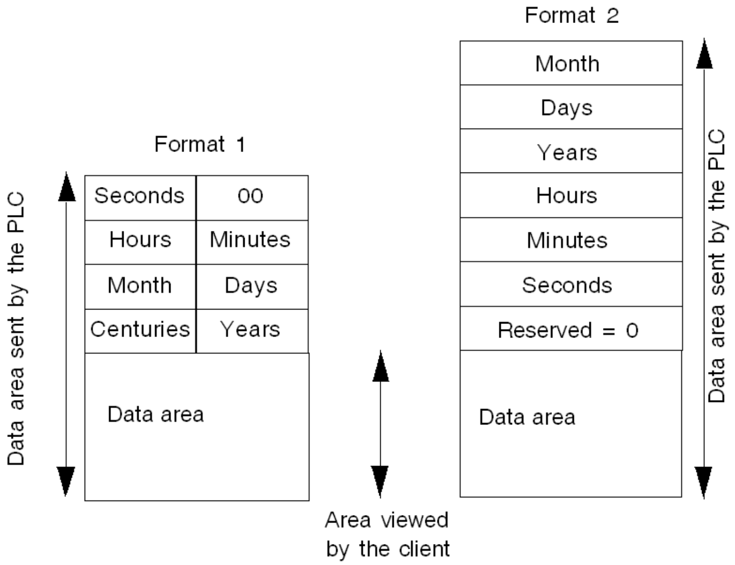
The Timestamp option may be enabled individually for each alias, from the properties page.

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 CD.

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:**

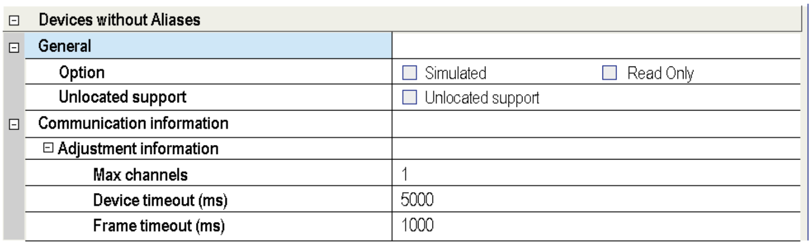
- On a Premium using TCP/IP the PC extension number must be equal or greater than 100 in the configuration of the Ethernet module in order to specify the use of the Modbus/TCP protocol.
- On a Quantum, the communication function blocks use only located variables.

# 8.6 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 |
| 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 113)*. The selections made here are applied only to the devices created without aliases, or aliases created dynamically with OFS Manager, at server run time.

# 8.7 The Deadband folder

### Aim of this Section

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

### What's in this Section?

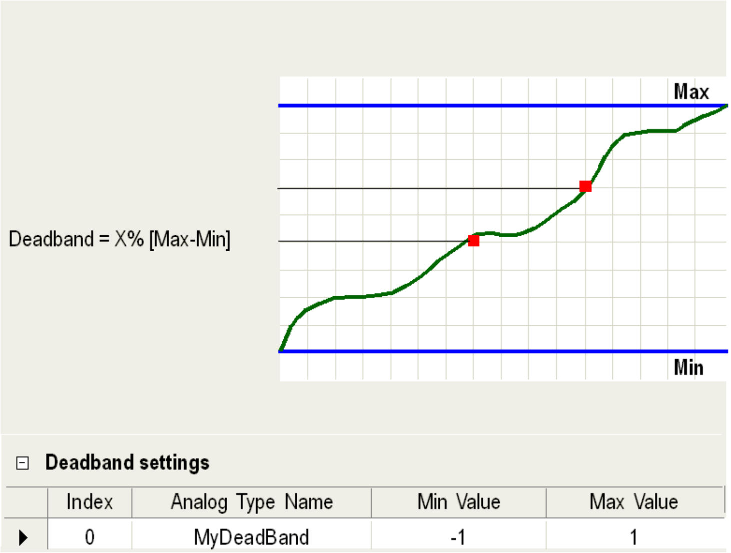
This section contains the following topics:

# 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:



⊟ Deadband settings

| | Index | Analog Type Name | Min Value | Max Value |
|---|---|---|---|---|
| ▶ | 0 | MyDeadBand | -1 | 1 |

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 saves CPU time.

**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. The limits for an analog type cannot be modified by the user once the server has started. This is due to the fact that when it is started, the OFS server learns the analog types defined in the registry. The OFS server must be stopped and then restarted for an alteration of the limits of an analog type to be taken into account.

- 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 *(see page 189)* 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**:

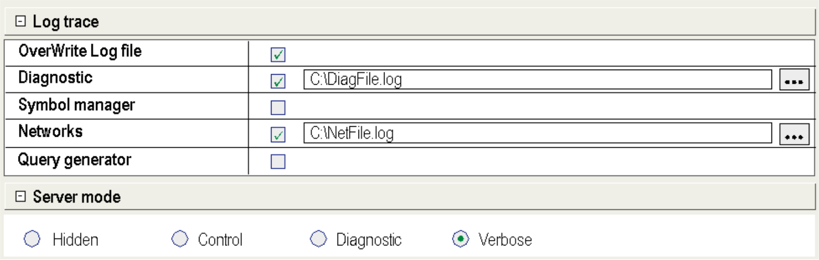
- it is possible to have the same item twice in the same group (e.g. "%MF330") with and without the analog type suffix (" @AnalogType"), so as to be able to compare the effect of dead-banding for notification filtering.
- It is possible to have items of different analog types in the same group (i.e.: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

# 8.8 The Diagnostic folder

## The Diagnostic folder

### Description

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



⊟ Log trace

| | | |
|---|---|---|
| OverWrite Log file | ☑ | |
| Diagnostic | ☑ | C:\DiagFile.log ... |
| Symbol manager | □ | |
| Networks | ☑ | C:\NetFile.log ... |
| Query generator | □ | |

⊟ Server mode

○ Hidden ○ Control ○ Diagnostic ● Verbose

The table below 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. |
| 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. |
| Query generator | Adds data on the generated queries in the networks file. |
| Server mode | **Hidden**: The server is invisible on screen.<br>**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.<br>**Diagnostic**: A complete set of diagnostic windows is displayed when the server is running, including a plotting window displaying warning and/or system messages.<br>**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. |

# 8.9 The Simulator folder

## Aim of this Section

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

## What's in this Section?

This section contains the following topics:

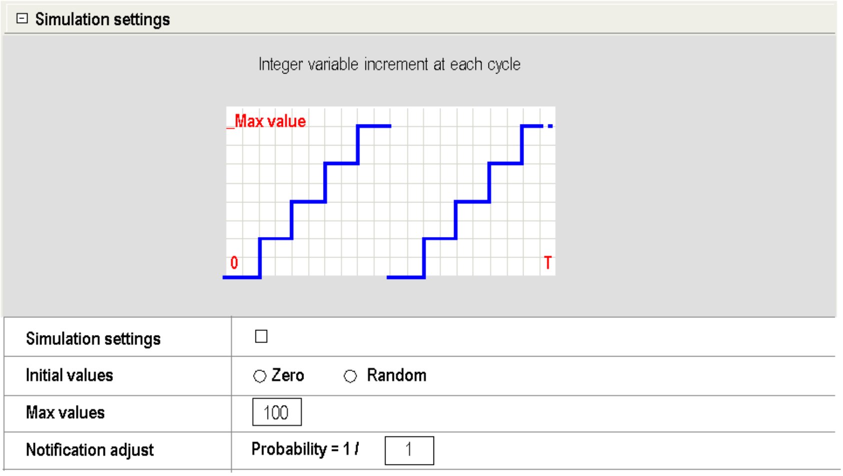
# The Simulator folder

## Presentation

If the alias has been configured with the simulation property *(see page 105)*, 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.



⊟ Simulation settings

Integer variable increment at each cycle

| | |
|---|---|
| Simulation settings | □ |
| Initial values | ○ Zero ○ Random |
| Max values | 100 |
| Notification adjust | Probability = 1 / 1 |

| Field | Description |
|---|---|
| Simulator mode | Allows you to select the simulation mode. |
| Initial values | **Random**: the variables are initialized to random values.<br>**Zero**: all variables are initialized to zero. |
| Max value | Maximum value for the simulated variable. Range: [0..32767].<br>The variable is increased at each cycle, and returns to 0 when the maximum value is reached (cyclically).<br>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)<br>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.<br>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 105)*.

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).

# 8.10 The Symbols folder

## The Symbols folder

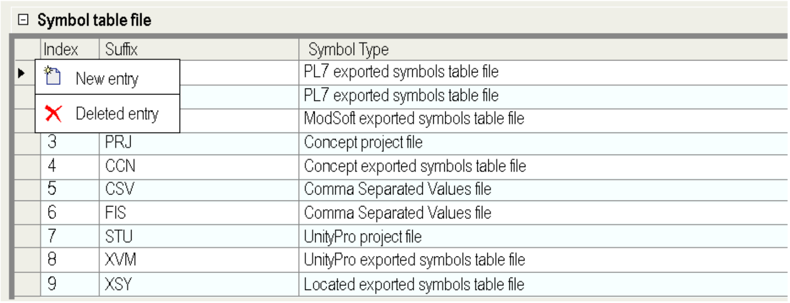
### Description

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:



⊟ Symbol table file

| Index | Suffix | Symbol Type |
|---|---|---|
| | | PL7 exported symbols table file |
| | | PL7 exported symbols table file |
| | | 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:

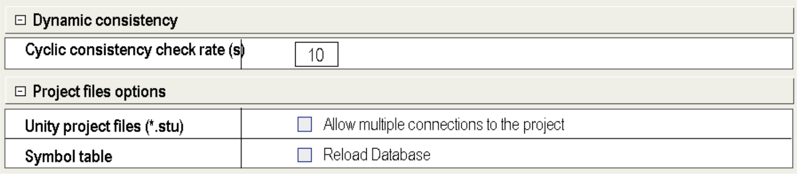
| Step | Action |
|---|---|
| 1 | Create a new entry. |
| 2 | Enter an extension, then press **Return**. |
| 3 | Double click on the corresponding field in the **Symbol Type** column.<br>***Result***: A list appears. |
| 4 | Select a file type. |

# 8.11 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.<br>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).<br>**Symbol table**: If the 'Reload Database' option is checked, the server reloads automatically the new database in case of inconsistency detection. |

# 8.12 The Communication folder

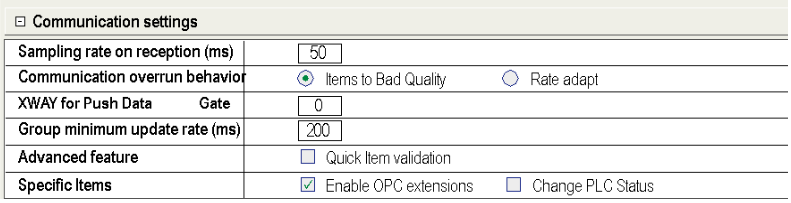
## The Communication folder

### 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

| | |
|---|---|
| Sampling rate on reception (ms) | 50 |
| Communication overrun behavior | ● Items to Bad Quality ○ Rate adapt |
| XWAY for Push Data Gate | 0 |
| Group minimum update rate (ms) | 200 |
| Advanced feature | □ Quick Item validation |
| Specific Items | ☑ Enable OPC extensions □ Change PLC Status |

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

| Field | Description |
|---|---|
| Sampling rate on reception (ms) | Defines the period for checking data reception in milliseconds. Range: [10..32767]<br>Values should be adjusted carefully as they affect the CPU load on your computer. |
| 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.<br>**Set items to Quality Bad** option: The items must be read with the group's update period (default parameters), otherwise the quality of the unrefreshed items is declared **Bad**.<br>**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**.<br>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. |
| X-Way for push data | **Gate**: value of the reception gate on which the server will receive data from the remote device.<br>Range: [0..255] |

| | |
|---|---|
| Group minimum update rate (ms) | Minimum update rate allowed for groups. The update rate that will be set by the client for a given group should also be a multiple of that numerical value in ms. Range:[20..32767] |
| Advanced feature | If the option ‘Quick Item validation’ is checked, item validation makes no physical access to the corresponding device (request emission).<br>If the device is not accessible, this avoids the server having to await communication time out. |
| Specific Items | **Enable OPC extensions**: Enables/disables the specific items *(see page 194)*.<br>**Change PLC status**: If the **Enable OPC extensions** option only is checked, it allows the server to change the PLC operating mode (RUN/STOP) |

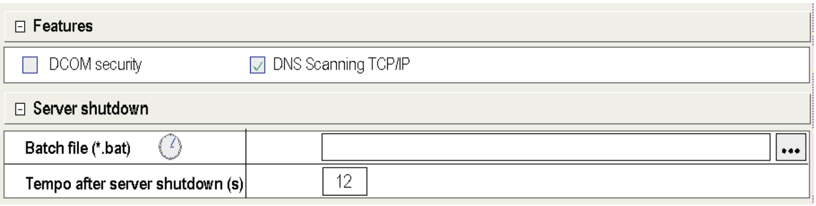
**Note**: If the **Change PLC status** option is not checked, any client application attempting to write on the #PLCStatus item receives an system message.

# 8.13 The Options folder

## The Options folder

### Description

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



⊟ Features

□ DCOM security ☑ DNS Scanning TCP/IP

⊟ Server shutdown

Batch file (*.bat)

Tempo after server shutdown (s) 12

| Field | Description |
|---|---|
| DCOM security | Enables/disables DCOM security<br>*(see page 51)* |
| DNS Scanning TCP/IP | Authorizes the server to use DNS to identify the PLC. |
| 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.<br>If moreover, a .BAT file is indicated, the timeout will be armed after running the batch process (limited to 10 seconds). |

# 8.14 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, a warning 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) a warning 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.

# 8.15 Compatibility with Previous Versions of Configuration Tool

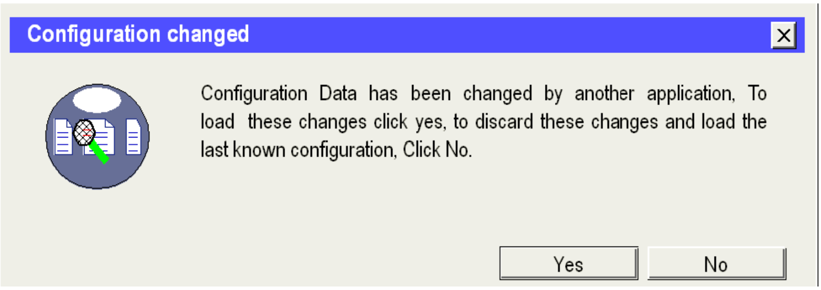
## Compatibility with the Previous Version of the Configuration Tool

### 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:



Configuration changed

Configuration Data has been changed by another application, To load these changes click yes, to discard these changes and load the last known configuration, Click No.

If you answer:

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

# The OFS manager tool

# 9

## The OFS Manager

### Description

The OFS Manager is a tool for troubleshooting and adjustment that works ONLY with OPC Factory Server (locally or remotely) or the OFS simulator. It is not recommended for use for other purposes, for instance during operation.

All troubleshooting functions of the OFS Manager are available from the debugging interface of the server. These functions are particularly useful for troubleshooting or when the OFS operates without its debugging 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 (including the warning and information 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 Vista, XP or 2000 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.

# The OFS Test Clients

# 10

## Aim of this Chapter

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

## What's 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

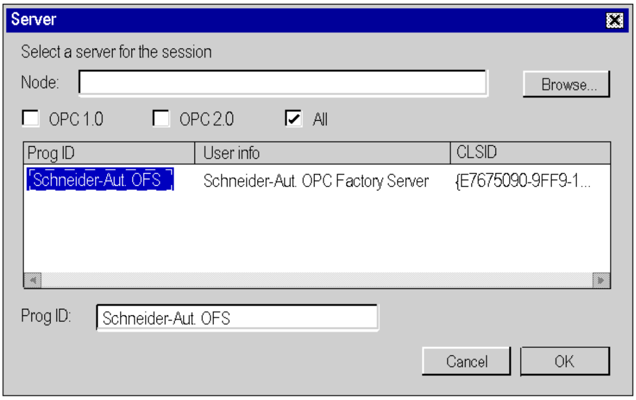
To install it on your machine, select the "Sample OPC client" option during installation of the full extension or the option "OFS server test client" during the installation of the remote extension.

### 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:<br><br>Server<br>Select a server for the session<br>Node:<br>□ OPC 1.0 □ OPC 2.0 ☑ All<br>Prog ID: Schneider-Aut. OFS — User info: Schneider-Aut. OPC Factory Server — CLSID: {E7675090-9FF9-1...<br>Prog ID: Schneider-Aut. OFS<br> |
| 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 150)* 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**

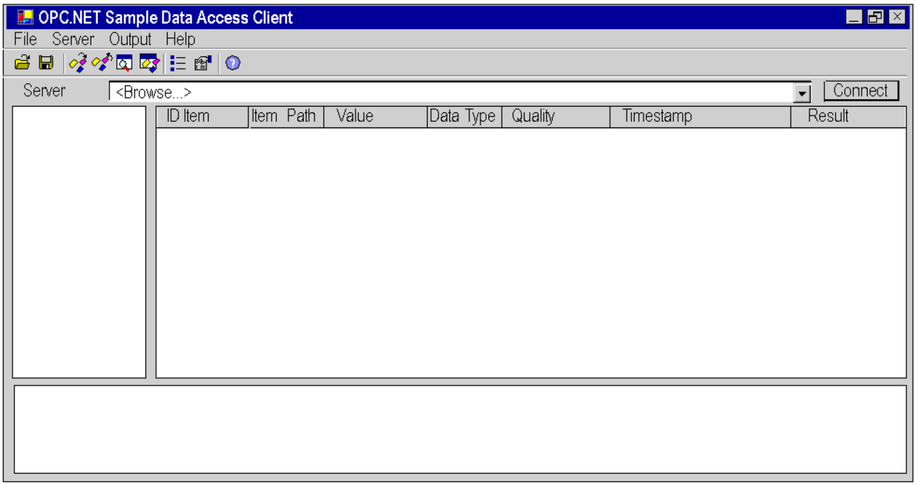
**UNINTENDED EQUIPMENT OPERATION**

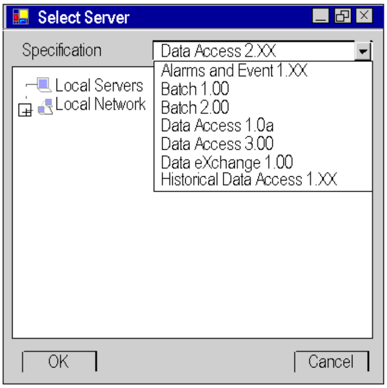
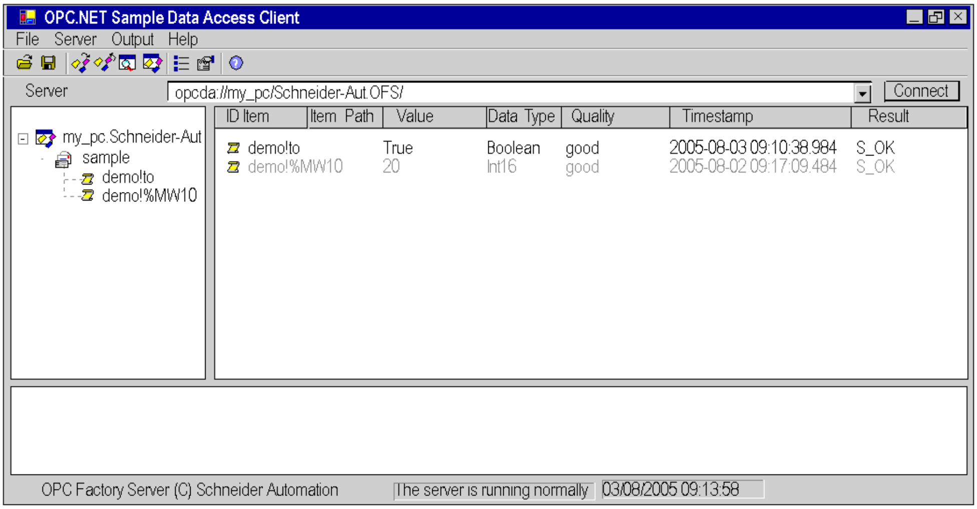
- Limit or restrict access to qualified personnel.
- Provide appropriate and independent protection via your application or process.

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

## Connecting the Client to the OPC Server

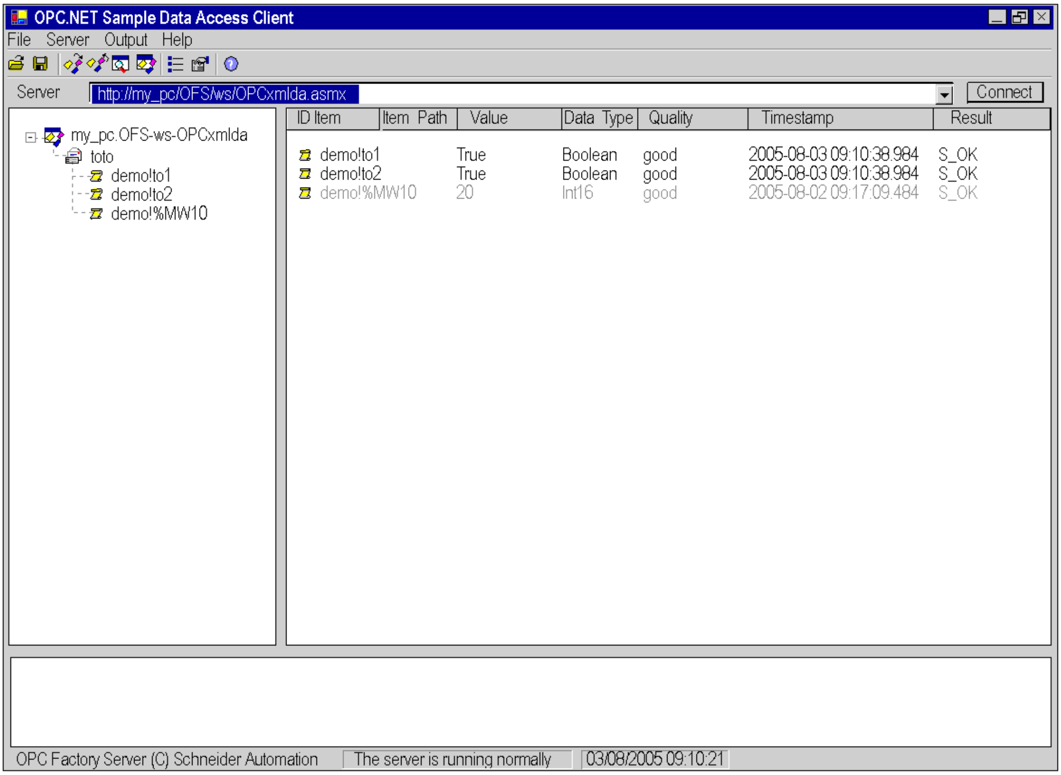
The following table shows how to connect the .NET OPC DA/OPC XML-DA client to the OFS server via OPC DA:

| 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:<br><br> OPC.NET Sample Data Access Client<br>File<br>Server<br>Output<br>Help<br>Server<br><Browse...><br>Connect<br>ID Item<br>Item Path<br>Value<br>Data Type<br>Quality<br>Timestamp<br>Result |

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>3</td><td>Click on &lt;Browse&gt; for the list the servers that may be reached.<br>The following window appears on top of the opening window:<br><br> <br>Specification: Data Access 2.XX</td></tr>
<tr><td>4</td><td>When the .NET OPC DA/OPC XML-DA client is started, the following window appears:<br><br> <br>Server: opcda://my_pc/Schneider-Aut.OFS/
<table>
<tr><th>ID Item</th><th>Item Path</th><th>Value</th><th>Data Type</th><th>Quality</th><th>Timestamp</th><th>Result</th></tr>
<tr><td>demo!to</td><td></td><td>True</td><td>Boolean</td><td>good</td><td>2005-08-03 09:10:38.984</td><td>S_OK</td></tr>
<tr><td>demo!%MW10</td><td></td><td>20</td><td>Int16</td><td>good</td><td>2005-08-02 09:17:09.484</td><td>S_OK</td></tr>
</table>
OPC Factory Server (C) Schneider Automation | The server is running normally | 03/08/2005 09:13:58</td></tr>
</table>

## 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:



Server http://my_pc/OFS/ws/OPCxmlda.asmx

| ID Item | Item Path | Value | Data Type | Quality | Timestamp | Result |
|---|---|---|---|---|---|---|
| demo!to1 | | True | Boolean | good | 2005-08-03 09:10:38.984 | S_OK |
| demo!to2 | | True | Boolean | good | 2005-08-03 09:10:38.984 | S_OK |
| demo!%MW10 | | 20 | Int16 | good | 2005-08-02 09:17:09.484 | S_OK |

OPC Factory Server (C) Schneider Automation | The server is running normally | 03/08/2005 09:10:21

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.

# The diagnostics screens of OPC Factory Server

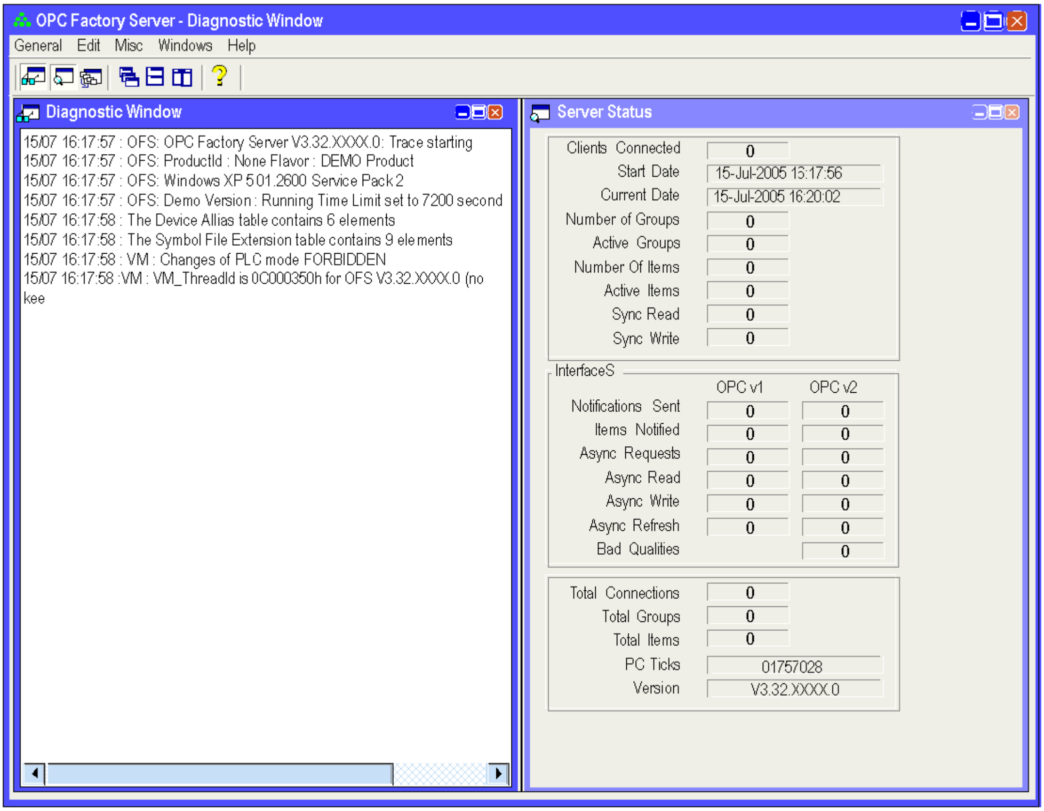
# 11

## 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).



```
15/07 16:17:57 : OFS: OPC Factory Server V3.32.XXXX.0: Trace starting
15/07 16:17:57 : OFS: ProductId : None Flavor : DEMO Product
15/07 16:17:57 : OFS: Windows XP 5.01.2600 Service Pack 2
15/07 16:17:57 : OFS: Demo Version : Running Time Limit set to 7200 second
15/07 16:17:58 : The Device Allias table contains 6 elements
15/07 16:17:58 : The Symbol File Extension table contains 9 elements
15/07 16:17:58 : VM : Changes of PLC mode FORBIDDEN
15/07 16:17:58 :VM : VM_ThreadId is 0C000350h for OFS V3.32.XXXX.0 (no
kee
```

| | |
|---|---|
| Clients Connected | 0 |
| Start Date | 15-Jul-2005 13:17:56 |
| Current Date | 15-Jul-2005 16:20:02 |
| Number of Groups | 0 |
| Active Groups | 0 |
| Number Of Items | 0 |
| Active Items | 0 |
| Sync Read | 0 |
| Sync Write | 0 |

InterfaceS

| | OPC v1 | OPC v2 |
|---|---|---|
| Notifications Sent | 0 | 0 |
| Items Notified | 0 | 0 |
| Async Requests | 0 | 0 |
| Async Read | 0 | 0 |
| Async Write | 0 | 0 |
| Async Refresh | 0 | 0 |
| Bad Qualities | | 0 |

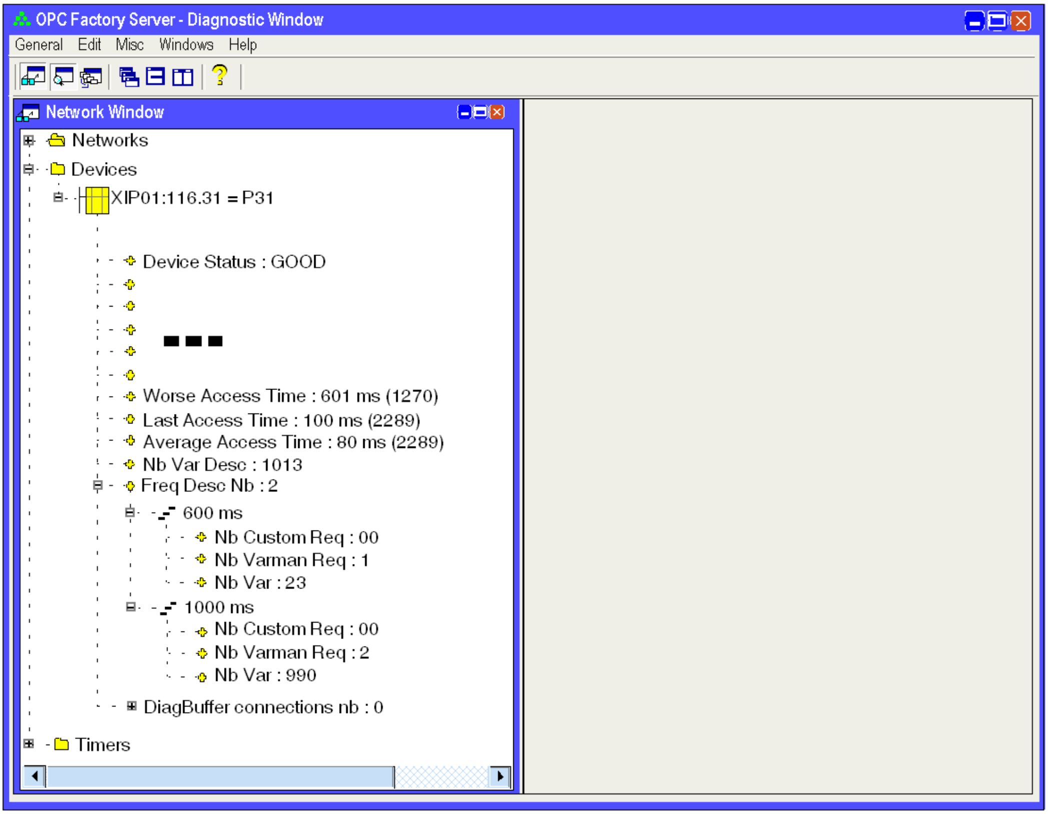
| | |
|---|---|
| Total Connections | 0 |
| Total Groups | 0 |
| Total Items | 0 |
| PC Ticks | 01757028 |
| Version | V3.32.XXXX.0 |

**NOTE:** The characters "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.



```
Networks
Devices
    XIP01:116.31 = P31
        Device Status : GOOD
        ■■■
        Worse Access Time : 601 ms (1270)
        Last Access Time : 100 ms (2289)
        Average Access Time : 80 ms (2289)
        Nb Var Desc : 1013
        Freq Desc Nb : 2
            600 ms
                Nb Custom Req : 00
                Nb Varman Req : 1
                Nb Var : 23
            1000 ms
                Nb Custom Req : 00
                Nb Varman Req : 2
                Nb Var : 990
        DiagBuffer connections nb : 0
Timers
```

# The OFS simulator

# 12

## 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.

# The OFS server WEB site

# 13

## Aim of this Chapter

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

## What's in this Chapter?

This chapter contains the following topics:

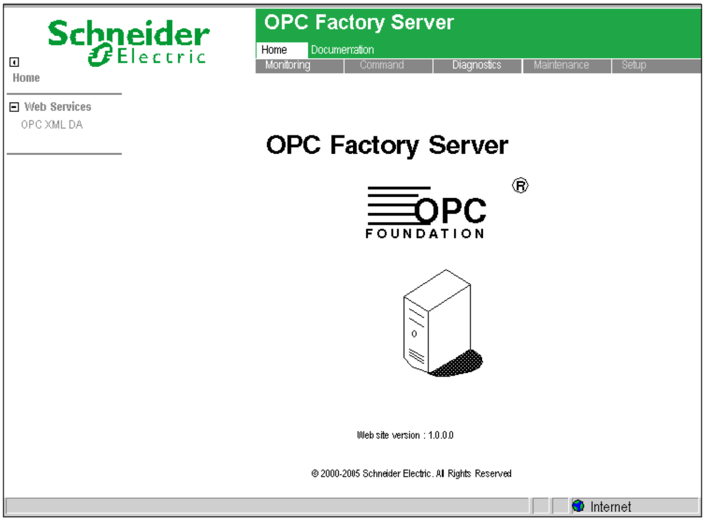
# Home Page of the OFS Web Site

## 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:<br>http://’name of the Web service machine’/OFS.<br>**Result**: The OFS server home page looks like this:<br><br> Schneider Electric<br>Home<br>Web Services<br>OPC XML DA<br>OPC Factory Server<br>Home<br>Documentation<br>Monitoring<br>Command<br>Diagnostics<br>Maintenance<br>Setup<br>OPC Factory Server<br>OPC FOUNDATION<br>Web site version : 1.0.0.0<br>© 2000-2005 Schneider Electric. All Rights Reserved<br>Internet<br>The Documentation link allows you to access and open Web services documentation. |

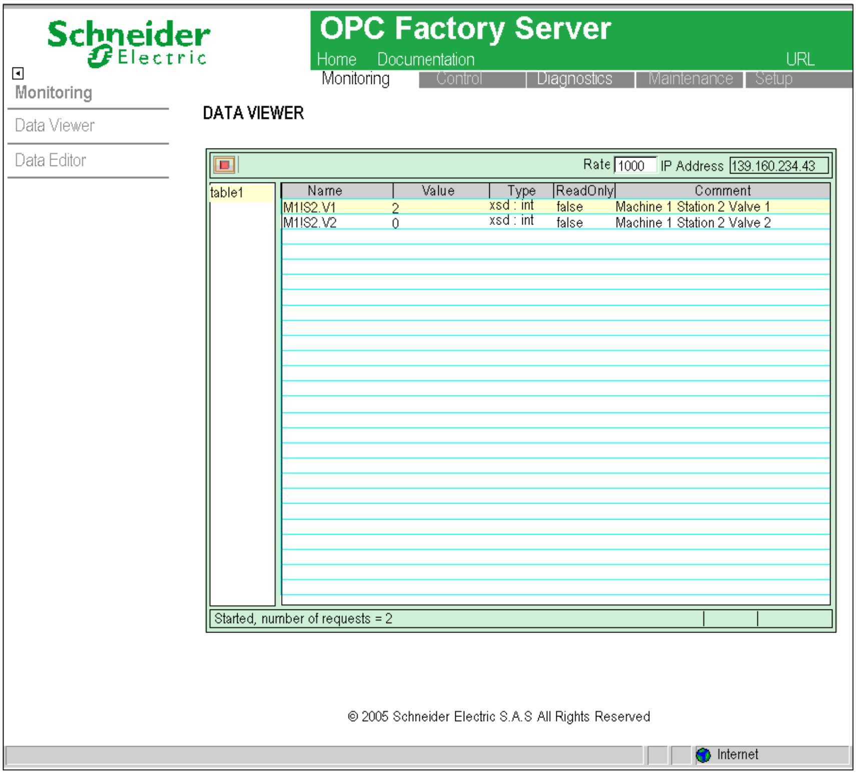
## Read Data Page

### At a Glance

The Read Data page can be used to view animation tables containing lists of device variables configured and communicating with the OFS server. This page cannot be used to create a table, to modify a variable or to force a variable value. The data viewer page uses the tables created by the data editor.

### Illustration

View of the Data Viewer page (read data) from an OFS server:



DATA VIEWER

Rate 1000 IP Address 139.160.234.43

table1

| Name | Value | Type | ReadOnly | Comment |
|---|---|---|---|---|
| M1!S2.V1 | 2 | xsd : int | false | Machine 1 Station 2 Valve 1 |
| M1!S2.V2 | 0 | xsd : int | false | Machine 1 Station 2 Valve 2 |

Started, number of requests = 2

© 2005 Schneider Electric S.A.S All Rights Reserved

**NOTE:** In the left-hand field the window displays the tables created by the editor. The user can select which table to display.

The variable comprises the following elements:

| **FIELD** | **FUNCTION** |
|---|---|
| Name | Name of the variable (mnemonic). |
| Value | Value of the variable or system message. |
| Type | Data type supported by OFS. |
| ReadOnly | If this box is selected the variable cannot be forced by the variable editor.<br>This field does not match the **access right** attribute of the OPC item. |
| Comment | Comment on the variable. |

## Data Editor Page

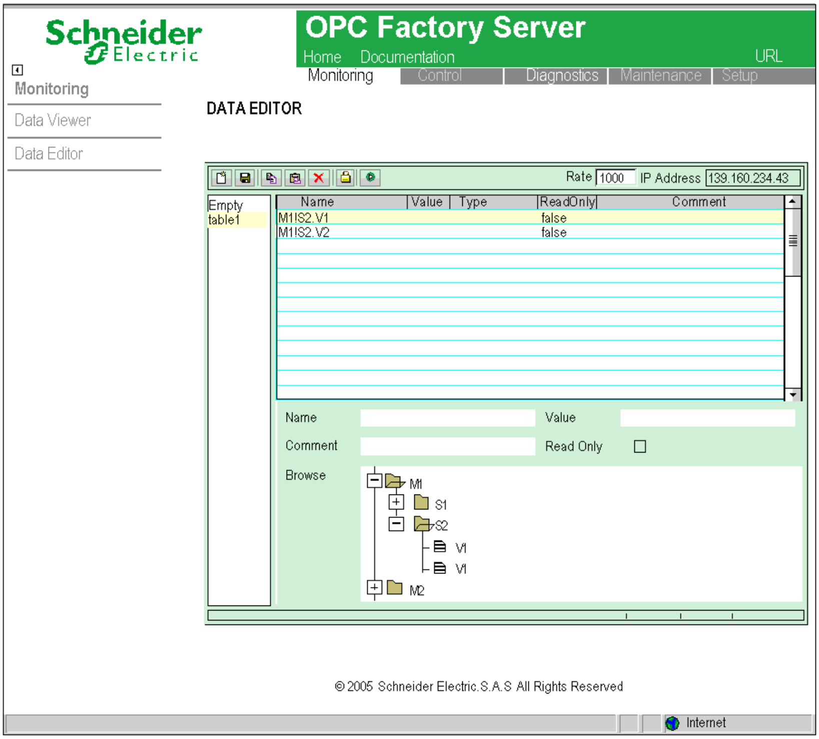
### At a Glance

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 qualified personnel.
> - Provide appropriate and independent protection via your application or process.
>
> **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:

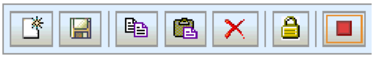


Rate 1000 IP Address 139.160.234.43

| Name | Value | Type | ReadOnly | Comment |
|---|---|---|---|---|
| M1!S2.V1 | | | false | |
| M1!S2.V2 | | | false | |

Name
Value
Comment
Read Only □

Description of the data editor buttons:



In sequential order:

- create a new table of variables,
- save a table, password protected,
- 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 (USER by default),
- start or stop the animation.

**NOTE:** write access to variables is password controlled. The password is the same as the one used to save the tables.

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**.

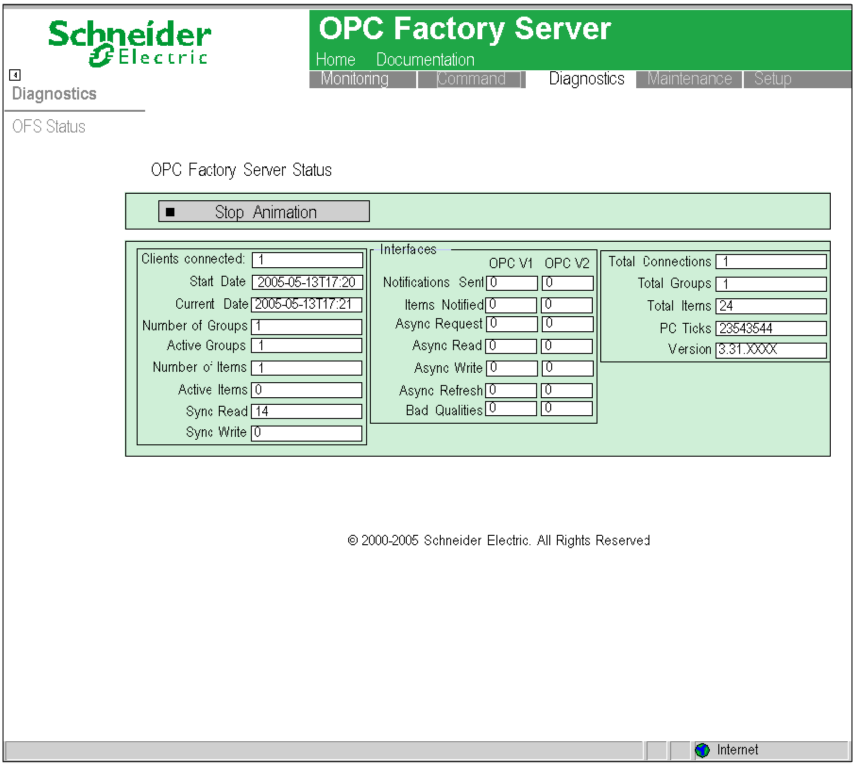
# OFS Diagnostics Home Page

## Diagnostics Page

This page displays the status of the OFS server.

## Illustration

The Diagnostics home page is the following:



OPC Factory Server Status

Clients connected: 1
Start Date 2005-05-13T17:20
Current Date 2005-05-13T17:21
Number of Groups 1
Active Groups 1
Number of Items 1
Active Items 0
Sync Read 14
Sync Write 0

Interfaces

|  | OPC V1 | OPC V2 |
|---|---|---|
| Notifications Sent | 0 | 0 |
| Items Notified | 0 | 0 |
| Async Request | 0 | 0 |
| Async Read | 0 | 0 |
| Async Write | 0 | 0 |
| Async Refresh | 0 | 0 |
| Bad Qualities | 0 | 0 |

Total Connections 1
Total Groups 1
Total Items 24
PC Ticks 23543544
Version 3.31.XXXX

© 2000-2005 Schneider Electric. All Rights Reserved

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

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

# User example

# V

# Example of using OFS

# 14

## 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's 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 87)*. 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

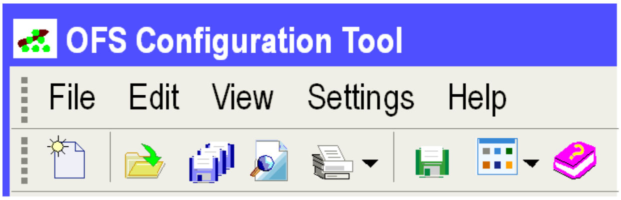
In this example, you will see how to use OFS to read and write a word in a Unity Pro PLC (Quantum or Premium) using TCP IP. You need to completely install OFS (client + server).

### 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:

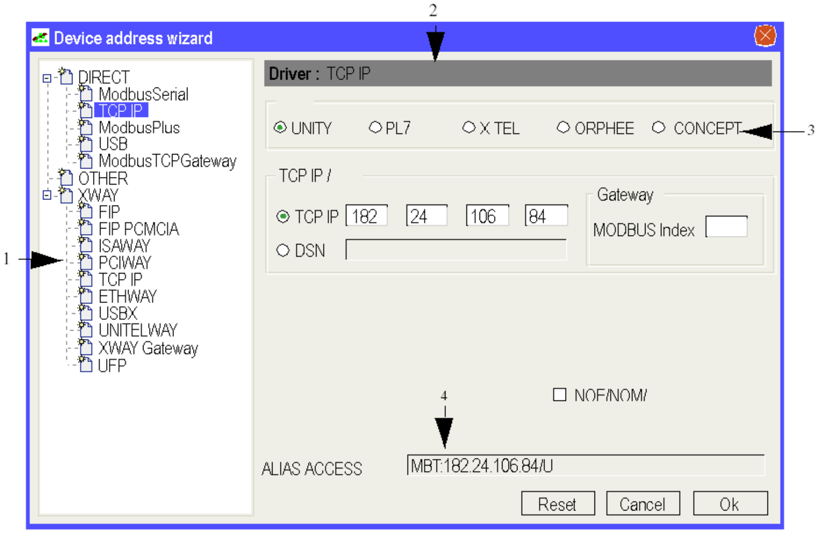
In this step, we are going to create and configure an alias:

- 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:


Device address wizard

Driver : TCP IP

● UNITY ○ PL7 ○ X TEL ○ ORPHEE ○ CONCEPT

TCP IP /

● TCP IP 182 24 106 84

○ DSN

Gateway

MODBUS Index

□ NOF/NOM/

ALIAS ACCESS MBT:182.24.106.84/U

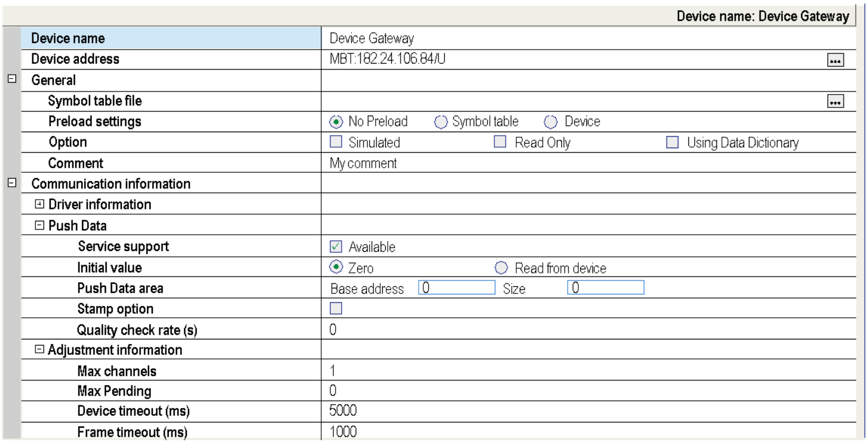
This screen will help you configure the alias. It is divided into 4 zones:
- zone 1 shows all the available drivers,
- zone 2 shows data for each driver,
- zone 3 enables you to identify the type of PLC in use,
- zone 4 displays the final data of the alias' address.

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: Device Gateway

| | |
|---|---|
| Device name | Device Gateway |
| Device address | MBT:182.24.106.84/U ... |
| ⊟ General | |
| Symbol table file | ... |
| Preload settings | ● No Preload ○ Symbol table ○ Device |
| Option | □ Simulated □ Read Only □ Using Data Dictionary |
| Comment | My comment |
| ⊟ Communication information | |
| ⊞ Driver 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 | 1 |
| Max Pending | 0 |
| Device timeout (ms) | 5000 |
| Frame timeout (ms) | 1000 |

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.

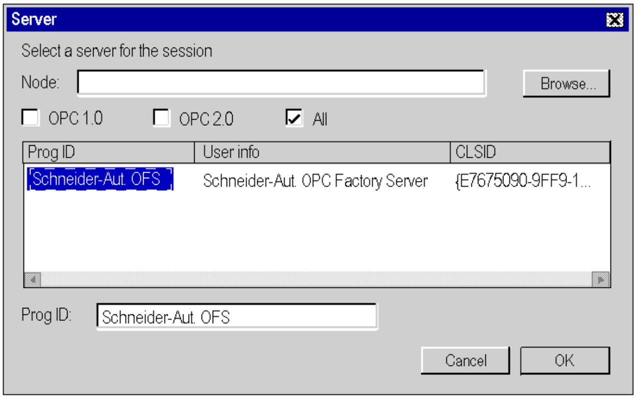
**NOTE:** When you make a change in the OFS configuration tool, you must re-start the server for the modifications to be taken into account.

## 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:



Server

Select a server for the session

Node:

☐ OPC 1.0 ☐ OPC 2.0 ☑ All

| Prog ID | User info | CLSID |
|---|---|---|
| Schneider-Aut. OFS | Schneider-Aut. OPC Factory Server | {E7675090-9FF9-1... |

Prog ID: Schneider-Aut. OFS

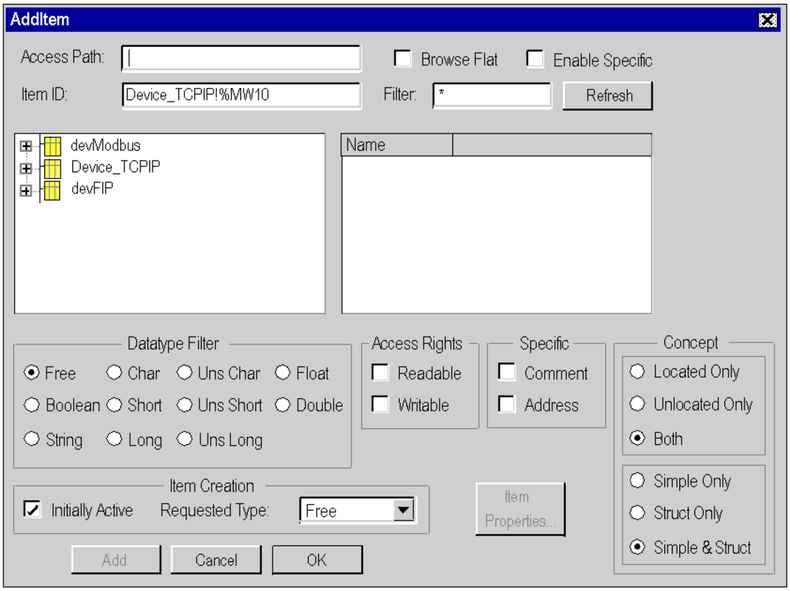
- Select the Schneider-Aut. OFS server in **Prog ID**,
- Enter the name of the computer of the OFS server in **Node**,
- and click on **OK** to connect to the OFS server.

## 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.



AddItem

Access Path: |

□ Browse Flat □ Enable Specific

Item ID: Device_TCPIP!%MW10

Filter: *

devModbus
Device_TCPIP
devFIP

Datatype Filter: ● Free ○ Char ○ Uns Char ○ Float ○ Boolean ○ Short ○ Uns Short ○ Double ○ String ○ Long ○ Uns Long

Access Rights: □ Readable □ Writable

Specific: □ Comment □ Address

Concept: ○ Located Only ○ Unlocated Only ● Both

○ Simple Only ○ Struct Only ● Simple & Struct

Item Creation: ☑ Initially Active Requested Type: Free

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.

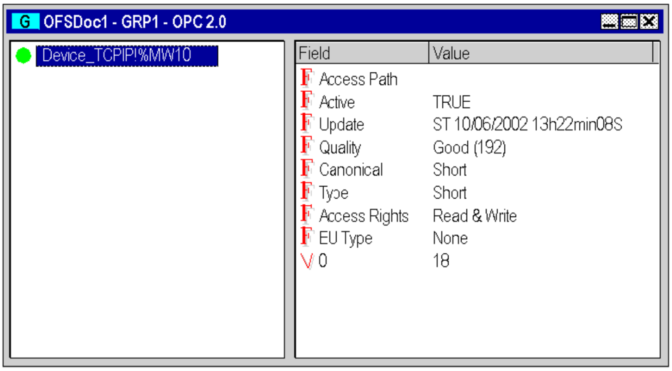
In our example we want to read and write a word. Let us take for example the word 10, or %MW10.

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:



● Device_TCPIP!%MW10

| Field | Value |
|---|---|
| F Access Path | |
| F Active | TRUE |
| F Update | ST 10/06/2002 13h22min08S |
| F Quality | Good (192) |
| F Canonical | Short |
| F Type | Short |
| F Access Rights | Read & Write |
| F EU Type | None |
| V 0 | 18 |

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.

# Advanced User Guide

# VI

## Aim of this section

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

## What's in this Part?

This part contains the following chapters:

# Concepts

15

## Aim of this Chapter

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

## What's 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).

**NOTE:** The conditions ensuring that the items in a group are consistent with one another (read or written in the same PLC scan) are described in read consistency *(see page 240)* and write consistency *(see page 241)*.

## 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 128)*).
  **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.

# Items

16

## Aim of this Chapter

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

## What's in this Chapter?

This chapter contains the following sections:

# 16.1 Items under OFS

## Aim of this Section

The aim of this section is to describe OPC items.

## What's 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**>]

The part <**nom driver**>:<**device address**> /<Device Type> may be replaced by an alias *(see page 105)* 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 * | 0.31.0 | Fipway adapter 01 or 02 |
| FPP2001 * | 0.31.0 | Fipway PCMCIA adapter 01 |
| ISAway01, ISAway02 | 0.5 | ISAway adapter 01 or 02 |
| Ethway01 *, Ethway02 * | 0.5 | Ethway adapter 01 or 02 |
| XIP01 -> XIP05 | 0.5 | X-Way TCP-IP adapter 01 to 05 |
| 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 | 0.6 | PCIway adapter 01 or 02 |
| USB | - | USB |
| USBX | 0.254.0 | USB with X-Way address |
| UFP01, UFP02 | 0.31.0 | 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 104)*.

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 suffix part can be R: R means 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: an item is a variable of any PLC which can be accessed either by their address or by their symbol.

- 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.

## 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 218*).

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 150*).

**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 150)*.

- 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" sub-folder 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 140)*.

| Name | Type | Access | Can be activated |
|---|---|---|---|
| **#AppliName** | VT_BSTR | R | No |

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

| Name | Type | Access | Can be activated |
|---|---|---|---|
| **#AppliVersion** | VT_BSTR | R | No |

**NOTE:** The value of #AppliName and #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).

| 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 will cut communication) to avoid timeouts or others.

If the value written is 1, the items related to that device will become "Bad" immediately since the server will stop 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.

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

This item is related to the multi-channel *(see page 324)* 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 105)*.

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 will be 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.

| 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 324)* configurations.

| 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 239)* 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

| 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 200)* option must have been checked in the "Communication" folder of the configuration tool.

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

| 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 will read the application name and version from the Device.

If this item is read, the server will perform 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.

| 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.

| Name | Type | Acces s | Can be enabled | Value read |
|---|---|---|---|---|
| **#DeviceIdentity** | VT_BSTR | R | No | PLC product reference |

ID: Application signature

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

OMC: Object modification counter

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

This item allows monitoring the communication activity of a device:

| Name | Type | Access | Can be enabled | Value read |
|---|---|---|---|---|
| **#PLCQualStatus** | VT_I2 | R | Yes | ● QUAL_BAD + QUAL_COMM_FAILURE if the device is INCONSISTENT (the SymbolFile is different than the PLC application)<br>● QUAL_BAD + QUAL_DEVICE_FAILURE if no more communication for DEVICE_TO milliseconde<br>● QUAL_BAD if the device is MISSING or UNKNOWN<br>● QUAL_GOOD if communication with the PLC occurs correctly |

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.

The #OFSStatus specific item is reserved for internal use only.

## Specific items supported on PLCs

The table below 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 |
| **#PLCQualStatus** | R | R | R | R | R | R |

## 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 140)*.

Changing the operating can change the system behavior.

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Keep strict 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.

## 16.2 Detected Error management

### Aim of this Section

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

### What's in this Section?

This section contains the following topics:

## 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 351)*.
- **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 warning. 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 interuption | Incorrect value due to a communication interuption 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.

# Variables

# 17

## Aim of this Chapter

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

## What's in this Chapter?

This chapter contains the following sections:

# 17.1 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.

# 17.2 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's 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:**

- Suffix S is used to read and write a character string-type variable in the form of a byte table (OCP-type VT_UI1 table).
- Suffix C is used to read and write a string-type variable in the form of a string of characters (OCP-type VT_BSTR).

### 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

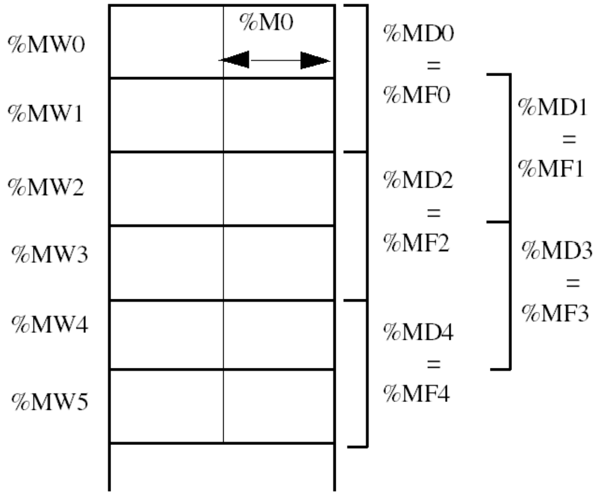
Access syntax:

| | 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 (for Premium PLC)** | | | |
| 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 |
| Word | %IW<@mod>.<c> | %IW4.2.3 | R |
| | %IW<@mod>.<c>.<d> | %IW4.2.3.1 | R |

| | Syntax | Example | Program access rights |
|---|---|---|---|
| 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 (for Premium PLC)** | | | |
| 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.

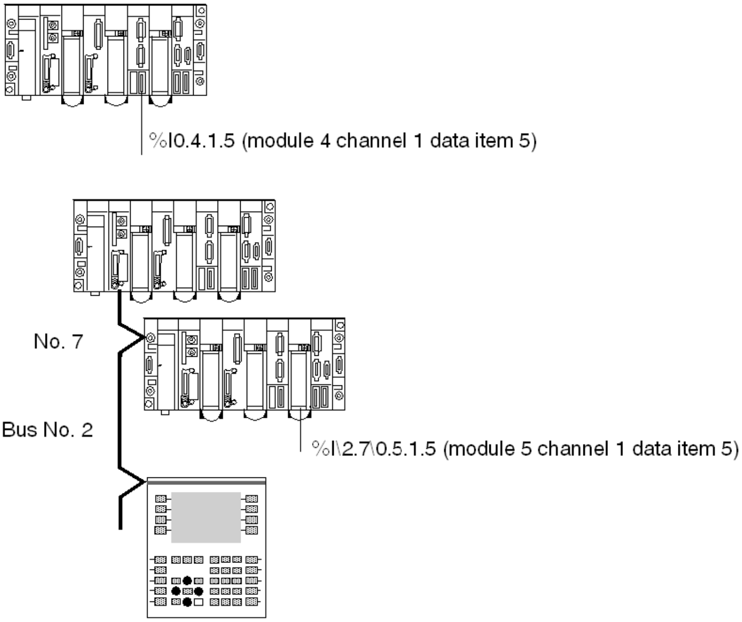
<m> module slot

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

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

**NOTE:** For Quantum PLCs, the syntax of inputs %I and outputs %Q is not managed. Instead you must use flat addresses such as %I *(see page 213)* and %M *(see page 211)*

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)

# 17.3 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's in this Section?

This section contains the following topics:

## 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<br>*%Bi*<br>*%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<br>*%Wi* | R/W | R/W | R/W | R/W |
| Internal double word | %MDi<br>*%DWi* | R/W | R/W | R/W | R/W |
| Floating point (32 bits) | %MFi<br>*%FDi* | R/W | R/W | R/W | R/W |
| Constant word | %KWi<br>*%CWi* | R | - | R | - |
| Constant double word | %KDi<br>*%CDi* | R | - | R | - |
| Constant floating point (32 bits) | %KFi<br>*%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<br>j=station no.<br>k=word no. | R/W | - | - | - |
| Common word on other networks | %NW{i.j}k<br>i=network no.<br>j=station no.<br>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]<br>%I\p.2.c\m.j[.r] | R | - | - |
| Discrete output | %Qi.j[.r]<br>%Q\p.2.c\m.j[.r] | R/W | - | - |
| Input word | %IWi.j[.r]<br>%IW\p.2.c\m.j[.r] | R | R | - |
| Output word | %QWi.j[.r]<br>%QW\p.2.c\m.j[.r] | R/W | R | - |
| Double input word | %IDi.j[.r]<br>%ID\p.2.c\m.j[.r] | R | R | - |
| Double output word | %QDi.j[.r]<br>%QD\p.2.c\m.j[.r] | R/W | R | - |
| Floating input (32 bits) | %IFi.j[.r]<br>%IF\p.2.c\m.j[.r] | R | R | - |
| Floating output (32 bits) | %QFi.j[.r]<br>%QF\p.2.c\m.j[.r] | R/W | R | - |
| Channel error bit | %Ii.j.ERR<br>%I\p.2.c\m.j.ERR | R | - | - |
| Module error bit | %Ii.MOD.ERR<br>%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 256)* 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 |

| Object | Syntax | TSX 37 | PCX/TSX 57 |
|---|---|---|---|
| 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<br>*%Bi:L*<br>*%Mxi:L* | R/W | R/W | R<br>W if module length is 8 | R<br>W if module length is 8 |
| Internal word | %MWi:L<br>*%Wi:L* | R/W | R/W | R/W | R/W |
| Double word | %MDi:L<br>*%DWi:L* | R/W | R/W | R/W | R/W |
| Floating point (32 bits) | %MFi:L<br>*%FDi:L* | R/W | R/W | R/W | R/W |
| Constant word | %KWi:L<br>*%CWi:L* | R | - | R | - |
| Constant double word | %KDi:L<br>*%CDi:L* | R | - | R | - |
| Constant floating point (32 bits) | %KFi:L<br>*%CFi:L* | R | - | R | - |
| Common word on network 0 | %NW{j}k:L<br>j = station no.<br>k = word no. | R/W | R/W | - | - |
| Common word on other networks | %NW{i.j}k:L<br>i = network no.<br>j = station no.<br>k = word no. | R/W | R/W | - | - |
| Character string | %MBi:L<br>*%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.

# 17.4 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's 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 102)*,
- 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.
- Suffix S is used to read and write a variable in the form of a byte table (OCP-type VT_UI1 table).

## 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).

# 17.5 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 suffix allows reading and writing a variable to a byte array (Array of VT_UI1 OPC type).

# 17.6 Variables in general

### Aim of this Section

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

### What's 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 211* | - |
| Word extracted bit | %MWi:Xj<br>*%Wi,j* | R | R/W | | R |
| Double word extracted bit | %MDi:Xj<br>*%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 211* | %MB100:X6<br>%MW100:X3<br>%MD200:X25<br>%SW6:X7<br>%KW100:X0<br>pump :X4 | pump:X5<br>struct1.member: X8<br>tab1[1000]:X4 | W100,3<br>DW200,25<br>CW100,0<br>Pump,4 | %MW100:X3<br>%MD200:X25 | 300500:X11<br>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.

# Symbols

18

## Aim of this Chapter

The aim of this section is to describe symbol management within the OFS product.

## What's in this Chapter?

This chapter contains the following sections:

# 18.1 Symbol operation

### Aim of this Section

The aim of this section is to describe several features concerning symbols.

### What's 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 350)* 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:
  The OFS server guarantees that all 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 *(see page 327)*.
  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 lodgeable items in a write frame is generally less than the lodgeable 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 optimizers 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 guaranteed 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 *(see page 327)* 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.

**NOTE:** Synchronous read and write functions are possible during the periodic read. They are not recommended however, as they may bring about anomalies (notification is not sent, as the value was read before the required sending of notification.)

**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.

# 18.2 Symbol management

### Aim of this Section

The aim of this section is to describe symbol management.

### What's 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 102)* or Unity Pro *(see page 101)* 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 138)* 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).

## 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 105* and *The PLC Software folder, page 139*).

### 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 119)*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 105* and *The PLC Software folder, page 139*).

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 *Setting the alias properties, page 105* and *(see page 139)*. 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 100)* 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 105* and *The PLC Software folder, page 139*). 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 105* and *The PLC Software folder, page 139*).

## Browsing of symbols

### At a Glance

Symbols browsing is supported by the OPC-Browse interface. It has a multi-level hierarchy:

A node for each device declared in the registry (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 100)* associated with the device *(see page 100)* 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.

# 18.3 Symbols and links

### Overview

This section describes the various links.

### What's 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 responses only to the requests, which 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 also another mechanism based on periodic polling of the PLC application stamps in order to check the consistency between the PLC application and the symbols. This mechanism is enabled when the **Dynamic consistency checking** option is checked, allowing to detect any application change, even minor. This mechanism is associated with the 'Strict' level of consistency, and allows linking a SCADA application to a PLC application, disabling automatically the communication of even a minor change detected in the PLC.

This mechanism is not available when the user selects the direct synchronization by the PLC.

### Description

To install the Unity Pro link via the OFS server, you must simply select the .stu project file (see File Table *(see page 100)* and see Symbol management *(see page 244)*) 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 119)*.

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 using Windows Vista, XP & 2000.

**NOTE:** In case of direct UnityPro utilisation, the security of UnityPro must not be set 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 above.

To install the Concept link, you must simply select the prj project file (see *Associating a Symbols Table File, page 100* and see *Symbol management, page 244*) 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 119)*.

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 in Windows Vista, XP & 2000 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, simply 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 envisage using non-located variables:

- the Quantum executable MUST NOT be a "stripped" version,
- IEC runtime must be activated on the PLC,
- the "unlocated support" option must be checked in the properties page. Otherwise, no access to any unlocated variable will be performed.

## 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).

# 18.4 Symbols management through Direct PLC link

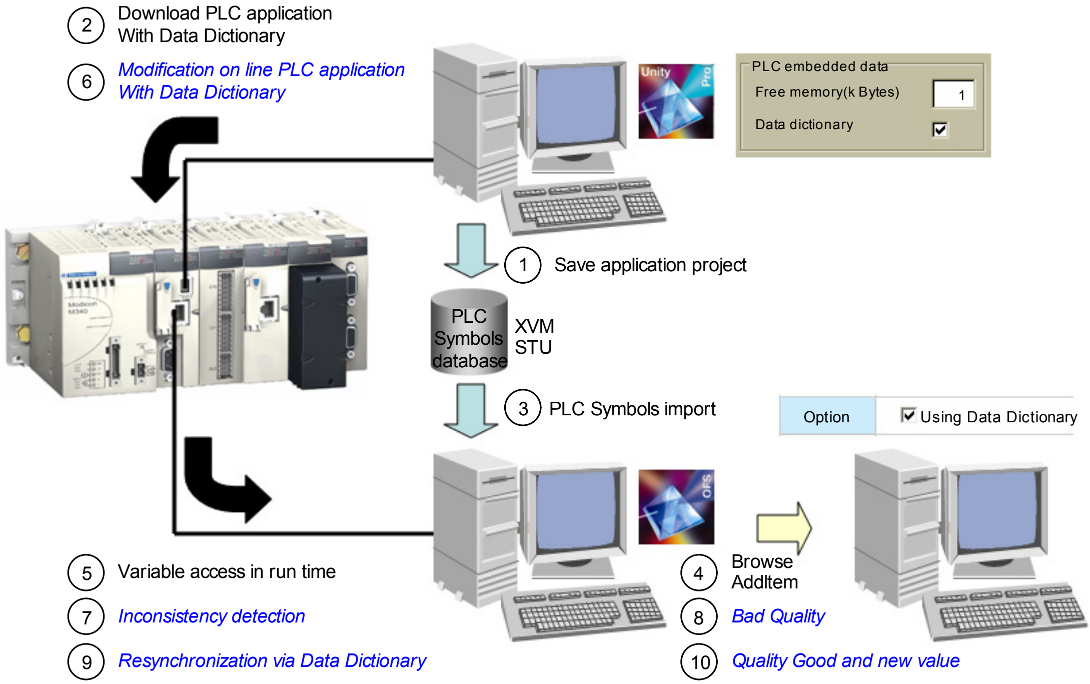
## Direct resynchronization

### Direct resynchronization of PLC symbol database

Thanks to the new application memory block 'Data Dictionary' generated by UnityPro V4.0 and thanks to specific requests allowing to get the address of variable from PLC, OFS is able to resynchronize the changing address of the variable after an online modification of the user of UnityPro connected to the PLC. The browse operation of OFS still performs through the UnityPro project files (STU) or UnityPro variable export file (XVM). When an inconsistency is detected, OFS resynchronizes the addresses of the variable from requests accessing the Data Dictionary. In case of incomplete resynchronization, OFS retrieves the addresses of the variable from UnityPro through STU or XVM files as the previous OFS behavior.

To enable a complete mechanism, the user has to check the 'Data Dictionary' option in the UnityPro project setting and the 'Using Data Dictionary' option in the OFS configuration tool for the corresponding device.

The illustration below shows a resynchronization mechanism:


2 Download PLC application
With Data Dictionary
6 *Modification on line PLC application*
*With Data Dictionary*
Unity Pro
1 Save application project
PLC Symbols database
XVM
STU
3 PLC Symbols import
OFS
5 Variable access in run time
7 *Inconsistency detection*
9 *Resynchronization via Data Dictionary*
4 Browse
AddItem
8 *Bad Quality*
10 *Quality Good and new value*

PLC embedded data

Free memory(k Bytes) 1

Data dictionary ☑

Option ☑ Using Data Dictionary

## Standard Start up sequence description

| Step | Sequence | Description |
|---|---|---|
| 1 | Save application project | After the build of the UnityPro application, it is saved as project file or exported as symbol file in order to be consistent with the binary application downloaded in the PLC. This operation can also be automatically done by checking the 'Auto-saving on download' option in the 'Project setting' of UnityPro. |
| 2 | Download PLC application with Data Dictionary | The binary application is downloaded into the CPU containing the data dictionary memory block. |
| 3 | PLC Symbols import | OFS can transfer the application symbols to save them. |
| 4 | Browse and AddItem from OFS client | The OFS client can browse and add items for this PLC. |
| 5 | Variable access in run time | The symbol file being consistent with the PLC application and the addresses of the PLC variable being correct, the values of the variable can be acquired. |

## Online modification sequence description

| Step | Sequence | Description |
|---|---|---|
| 6 | Online modification of the PLC application with Data Dictionary | Connected to the PLC with UnityPro software, the user can modify the application online. |
| 7 | Inconsistency detection | OFS server detects an inconsistency with the old symbol files if the modification affects the variable mapping. |
| 8 | Bad Quality on variable attribute | The values of the variables are no more animated and the qualities are set to 'Bad'. |
| 9 | Resynchronization according to the new Data Dictionary | In this case, OFS server gets the new addresses from requests of the CPU accessing the data dictionary. |
| 10 | Quality Good and new value | The OFS client retrieves once again in real time all the values of the application variables in 'Good' quality. |

## Versions Compatibilities

The automatic synchronization feature of PLC symbols is available for the following versions:

| | **UnityPro** | **CPU**[(*)] | **OFS** |
|---|---|---|---|
| **M340** | V4.0 | V2.0 | V3.33 |
| **Premium** | V4.0 | V2.6 | V3.33 |
| **Quantum** | V4.0 | V2.6 | V3.33 |

[(*)]not available for Safety CPU

For this feature to work properly, it is mandatory that UnityPro, PLC CPU and OFS have the minimum version level listed above.

If one or more of the concerned items is not at the right version, it is necessary to update the concerned item to get this automatic synchronization feature of PLC symbols.

**NOTE:** Due to communication limitation, and because the Data Dictionary is generated from UnityPro, the automatic resynchronization is available with the devices using ‘TCP IP direct’ driver over Ethernet network allowing communication requests of 1024 bytes length and ‘UNITY’ PLCs. As well, the safety CPU doesn’t implement the data dictionary.

**NOTE:** In case of XVM or STU not up-to-date in comparison with the downloaded PLC application, a variable can still be not found in the symbol export file or in the UnityPro project files and the result rejected even if this is present in the data dictionary embedded in the CPU. For this reason, it is strongly recommended to check the ‘Project autosaving on download’ option through ‘Tool/Option’ menu in UnityPro project.

**NOTE:** When *Using Data Dictionary* is checked, the *Preload settings* is implicitly set to *Device* by the OFS server.

# The Diag Buffer

# 19

## Overview

This chapter describes the Diag Buffer detection tool.

## What's in this Chapter?

This chapter contains the following sections:

# 19.1 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.

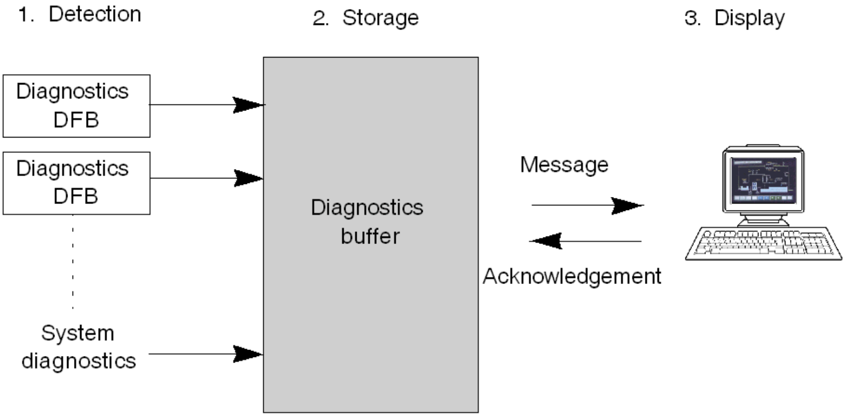
This function is only provided for Premium TSX57/PCX57 PLCs programmed with PL7 and those programmed by Unity Pro that have a minimum software version (see PL7 / Unity Pro documentation for further information).

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 | A buffer memory called diagnostics Buffer saves the detected faults in time-stamped message form. |
| 3 | One or more multi-station viewers (15max) allow you to:<br>● view of one or more areas of a PLC,<br>● view one or more areas of several PLCs,<br>● acknowledge messages,<br>● view the evolution of an item's status. |

For more information on the Diag buffer, see the section on Diag buffer installation *(see page 294)*.

# 19.2 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's in this Section?

This section contains the following topics:

## Operation from an OPC client

### Reminder about the Diag Buffer

The Diag buffer *(see page 266)* 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 diagnositc System or Application must be enabled in the application for Dag Buffer functioning.

### Description of the client interface

Diag Buffer functions authorize access to PLCs using specific items.

The table below shows the specific items:

| Service | Item | Type | Access | Value read | Value to write |
|---|---|---|---|---|---|
| Open connection | #DiagLogon | VT_UI2 | READ/WRITE | Viewer or 0xFFFF identifier | zone number |
| Close connection | #DiagLogout | VT_UI2 | READ/WRITE | Viewer or 0xFFFF 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<br>see *Information retrieved by the Diag buffer at the top of the table, page 288* |
| 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 |

| Service | Item | Type | Access | Value read | Value to write |
|---|---|---|---|---|---|
| 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:

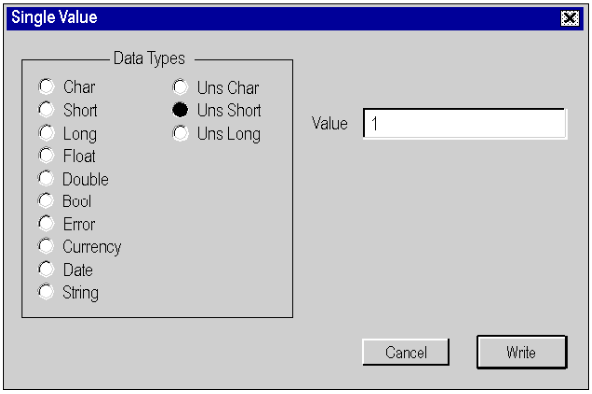
- VT = variable,
- UI1 = unsigned integer on 1 byte,
- UI2 = unsigned integer on 2 bytes,
- UI4 = unsigned integer on 4 bytes,
- ARRAY = table of bytes

### #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:



Single Value

Data Types

○ Char
○ Short
○ Long
○ Float
○ Double
○ Bool
○ Error
○ Currency
○ Date
○ String
○ Uns Char
● Uns Short
○ Uns Long

Value: 1

**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 the value 0040h
  - to monitor zones 2 and 15: write the value 8004h

**Value returned after read**:

- the viewer number is displayed if the connection is open, if not the connection is not established and 0xFFFF 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 |
| 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 the value FFFFh 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, the OxFFFF value 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 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 284)*.

Value returned by the item:

| HRESULT | Comment |
|---|---|
| S_OK | Read successful, no modification is recorded in the 120 byte table, or<br>read successful, modifications recorded in the 120 byte table (the anomaly has been acknowledged or has disappeared), or<br>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, 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_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 287)*.

**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] = 98h, Var[9] = 01h, Var[10] = 76h, Var[11] = 25h
  The value to write in the #DiagReadStatus item is 25760198h or 628490648d.

| Content | Size | Comment |
|---|---|---|
| Copy of Status Address field | 6 bytes | Memory address for PLC |
| Size of Status | 2 bytes | Byte length of status byte 0x00 |

**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 will update 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 |

| HRESULT | Comment |
|---|---|
| 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. User will have to 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 Fault cause |
| 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 will update the alarm, so user will get a new buffer for this alarm on #DiagReadNextError item. But this alarm will still show 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**:

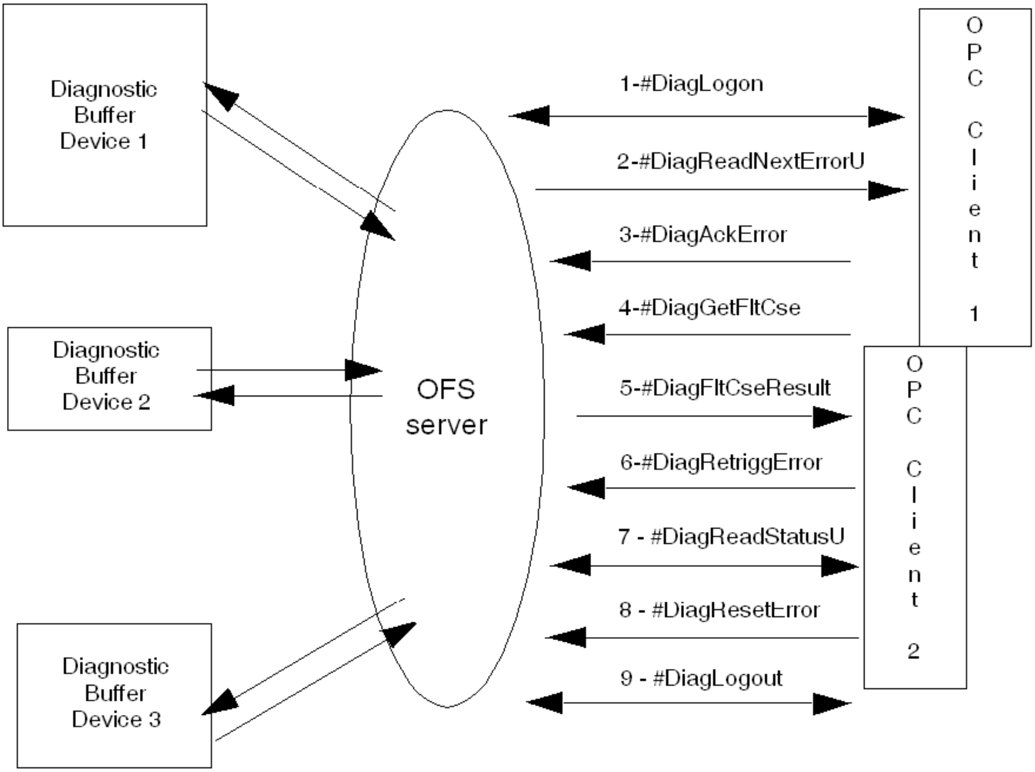
- nothing.

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

The diagram below 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 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 will be deleted from all viewers once it has been acknowledged on one viewer.

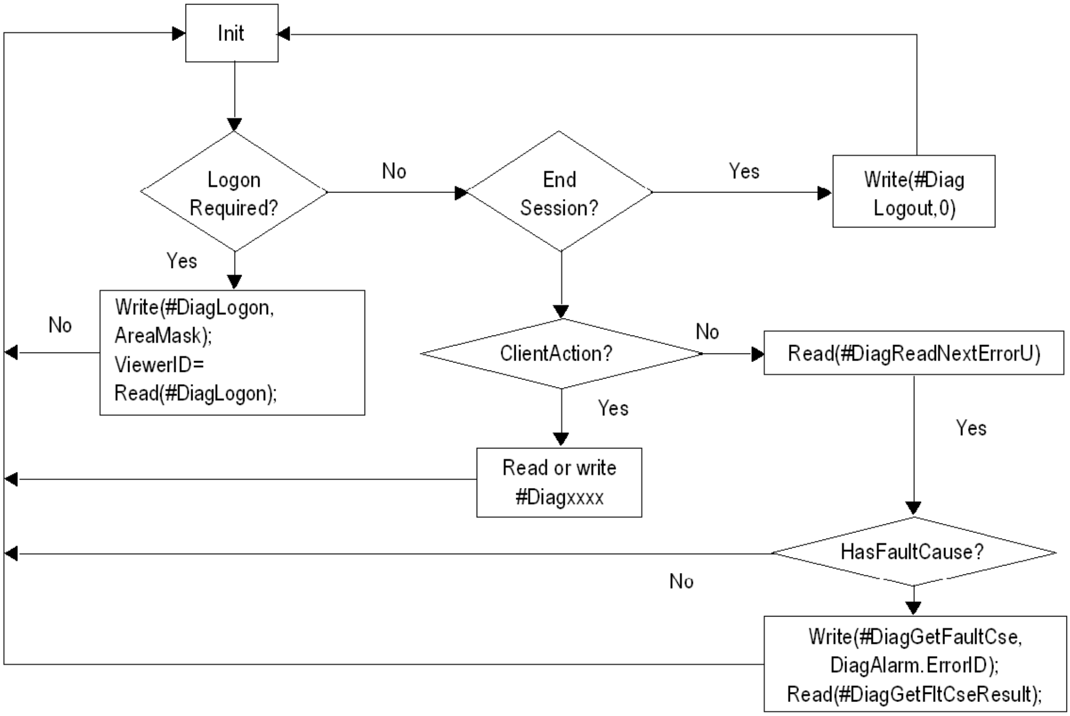
An alarm is deleted from the buffer if:

- 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

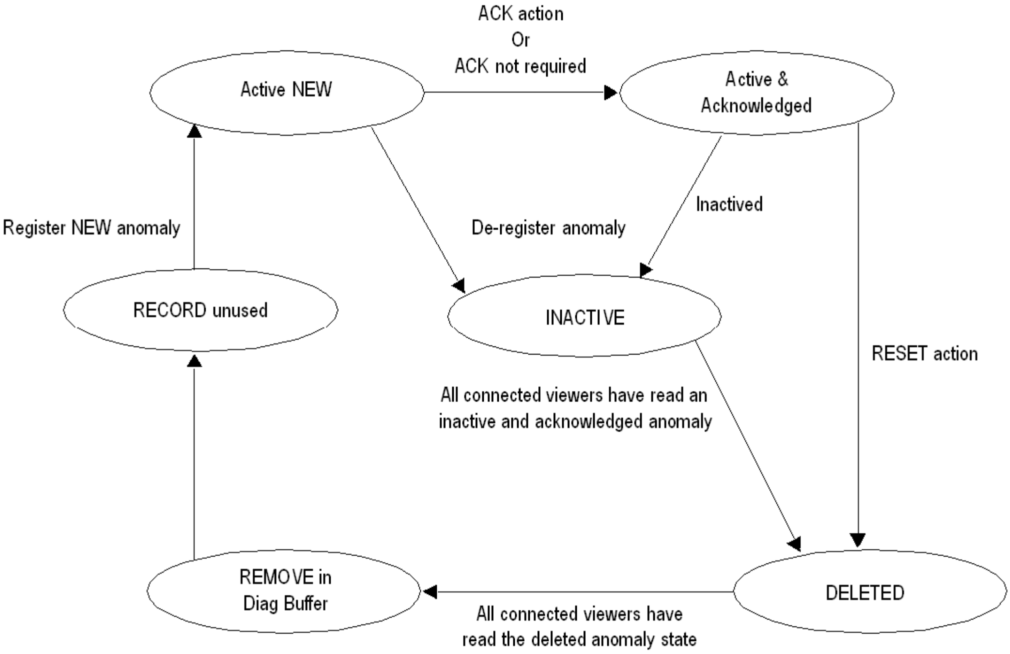
The graph below shows how and in which order an OPC client uses the specific items of Unity Pro Diag Buffer:


Init
Logon Required?
No
Yes
End Session?
Yes
Write(#Diag Logout,0)
Write(#DiagLogon, AreaMask); ViewerID= Read(#DiagLogon);
No
ClientAction?
No
Read(#DiagReadNextErrorU)
Yes
Read or write #Diagxxxx
Yes
HasFaultCause?
No
Write(#DiagGetFaultCse, DiagAlarm.ErrorID); Read(#DiagGetFltCseResult);

**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 will be 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 0xFFFFFFFE). The value 0xFFFFFFFF is reserved.

**Example** of the settings of the handle with the C++ test client supplied on the OPC Factory Server CD:

- 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.

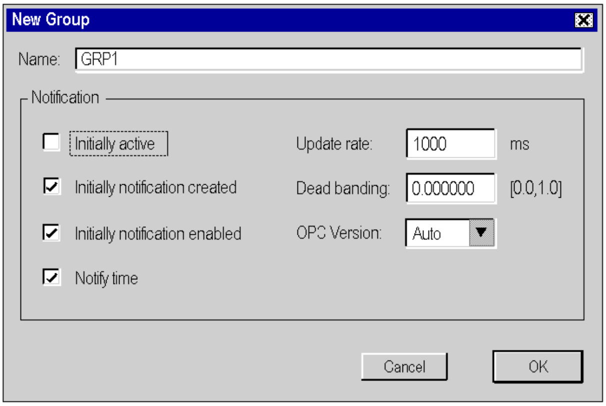
All the examples on the following pages use the test client supplied on the CD.

For more information on the OPC client see the OFS Client *(see page 150)*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**:



Name: GRP1

☐ Initially active

☑ Initially notification created

☑ Initially notification enabled

☑ Notify time

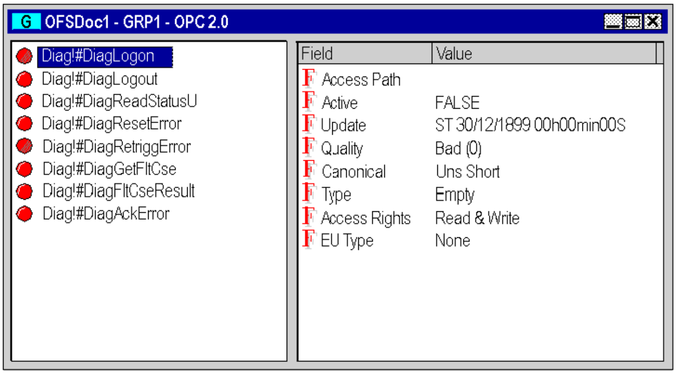
Update rate: 1000 ms

Dead banding: 0.000000 [0.0,1.0]

OPC Version: Auto

1- To connect to the diag buffer, the OPC client must add the #DiagLogon *(see page 270)* 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 272)* specific item to the group. The disconnection will be carried out when the client writes any value in this item.
3- For acknowledgment, the OPC client requires the #DiagAckError *(see page 273)* specific item in the group.
4- To update the Diag Buffer status, the OPC client needs to add the #DiagReadStatusU *(see page 274)* specific item.
5- To get the FaultCause system message, the OPC client requires the #DiagGetFltCse (to write) and the # DiagFltCseResult *(see page 276)* (to read).
6- To retrigger the list of FaultCause, the OPC client requires the #DiagRetriggError *(see page 278)*specific item to the group. The retrigg will be done when the client writes a valid ErrorID in this item.

7- For the deletion, the OPC client requires the #DiagResetError *(see page 274)* specific item to the group. The destruction will be done when the client writes a valid ErrorID in this item.
The following screen shows the installation of specific items:

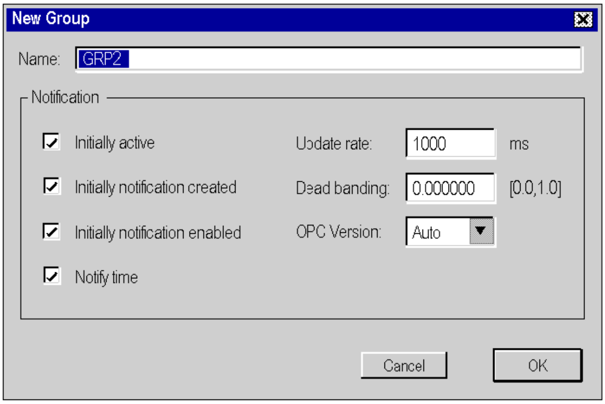


OFSDoc1 - GRP1 - OPC 2.0

- Diag!#DiagLogon
- Diag!#DiagLogout
- Diag!#DiagReadStatusU
- Diag!#DiagResetError
- Diag!#DiagRetriggError
- Diag!#DiagGetFltCse
- Diag!#DiagFltCseResult
- Diag!#DiagAckError

| Field | Value |
|---|---|
| Access Path | |
| Active | FALSE |
| Update | ST 30/12/1899 00h00min00S |
| Quality | Bad (0) |
| Canonical | Uns Short |
| Type | Empty |
| Access Rights | Read & Write |
| EU Type | None |

- **An active group**: this group must be created or activated after actual connection with the #DiagLogon item.



New Group

Name: GRP2

Notification

☑ Initially active

☑ Initially notification created

☑ Initially notification enabled

☑ Notify time

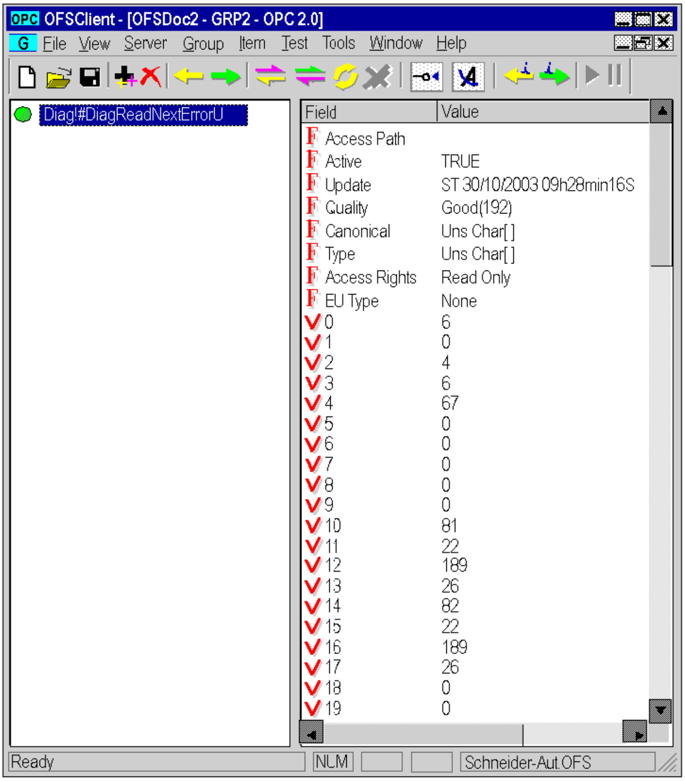
Update rate: 1000 ms

Dead banding: 0.000000 [0.0,1.0]

OPC Version: Auto

To reset the alarms coming from diag buffer, the client needs to add the #DiagReadNextErrorU *(see page 272)* specific item to the group.

The screen below shows a 550 bytes table *(see page 287)* in which the detected error *(see page 287)* code of the activated alarm can be found. Each byte represents a specific piece of information :


OPC OFSClient - [OFSDoc2 - GRP2 - OPC 2.0]
G File View Server Group Item Test Tools Window Help
Diag!#DiagReadNextErrorU
Ready
NUM
Schneider-Aut.OFS

| Field | Value |
|---|---|
| Access Path | |
| Active | TRUE |
| Update | ST 30/10/2003 09h28min16S |
| Quality | Good(192) |
| Canonical | Uns Char[ ] |
| Type | Uns Char[ ] |
| Access Rights | Read Only |
| EU Type | None |
| 0 | 6 |
| 1 | 0 |
| 2 | 4 |
| 3 | 6 |
| 4 | 67 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 81 |
| 11 | 22 |
| 12 | 189 |
| 13 | 26 |
| 14 | 82 |
| 15 | 22 |
| 16 | 189 |
| 17 | 26 |
| 18 | 0 |
| 19 | 0 |

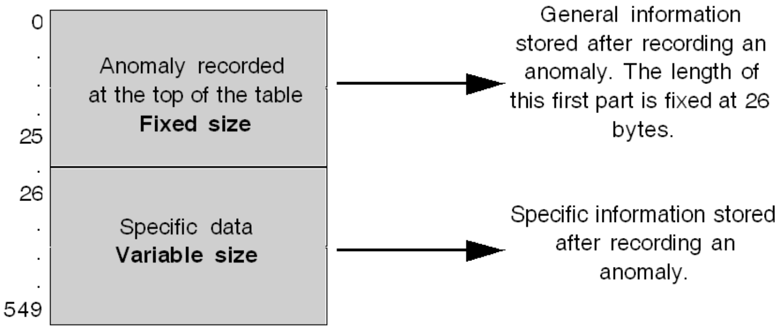
Example of translation, the bytes 2 and 3 represent the identifier of the anomaly.To acknowledge it, the client will write the value 0x0604 (decimal value is 1540) in #DiagAckErr.

## Diag buffer table formats

### Description

The 550 byte table *(see page 283)* (alarm reset after a read on #DiagReadNextError) is structured in the following way:

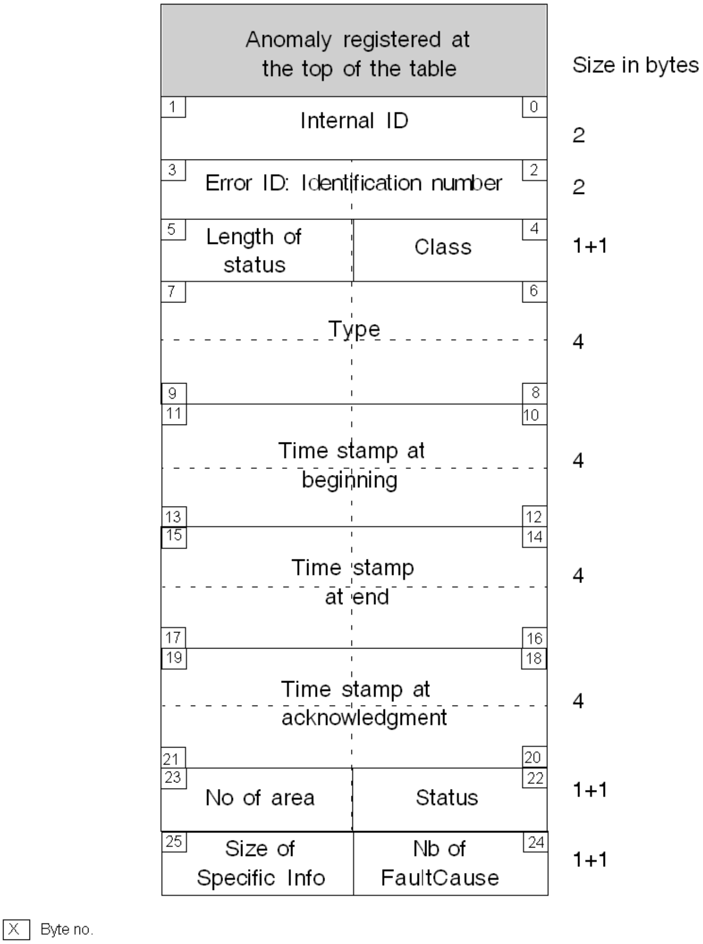
Illustration of the structure of the bytes table:


0
.
.
.
25
.
26
.
.
.
549
Anomaly recorded at the top of the table
Fixed size
Specific data
Variable size
General information stored after recording an anomaly. The length of this first part is fixed at 26 bytes.
Specific information stored after recording an anomaly.

## Information retrieved by the Diag buffer at the top of the table

### Description

The illustration below 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 below gives the definition of the code retrieved in this byte:

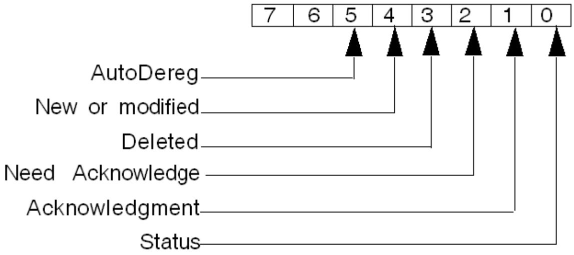
| Symbol | Value | Comment |
|---|---|---|
| OFS_DIAGU_CLASS_SFC_MIN_TIME | 0x00 | A SFC step exited too early |
| OFS_DIAGU_CLASS_SFC_MAX_TIME | 0x01 | A SFC step exited too late |
| OFS_DIAGU_CLASS_SFC_SNS_SYNC | 0x02 | Simultaneous synchronization error |
| | | |
| OFS_DIAGU_CLASS_FB_GEN_FB | 0x40 | Generic DFB |
| OFS_DIAGU_CLASS_FB_STD_EFB | 0x64 | Standard diagnostic EFB |
| OFS_DIAGU_CLASS_FB_USR_EFB | 0x65 | User diagnostic EFB |
| OFS_DIAGU_CLASS_FB_STD_DFB | 0x66 | Standard diagnostic DFB |
| OFS_DIAGU_CLASS_FB_USR_DFB | 0x67 | User diagnostic DFB |
| OFS_DIAGU_CLASS_FB_EXT | 0x68 | Extended EF/EFB/DFB |
| OFS_DIAGU_CLASS_FB_IO | 0x69 | IO detected errors reported |
| | | |
| OFS_DIAGU_CLASS_GEN_SYS | 0x80 | System generic detected error |
| OFS_DIAGU_CLASS_SYS_CMN | 0x85 | Common system IO detected error |
| OFS_DIAGU_CLASS_SYS_LIO | 0x86 | Local IO detected error (not used) |
| OFS_DIAGU_CLASS_SYS_RIO | 0x87 | Remote IO detected error (not used) |
| OFS_DIAGU_CLASS_SYS_CPU | 0x88 | CPU detected error |
| OFS_DIAGU_CLASS_SYS_IO | 0x89 | I/O detected error |
| OFS_DIAGU_CLASS_SYS_ARITH | 0x94 | Arithmetic detected error |
| OFS_DIAGU_CLASS_SYS_TSK | 0x95 | Task detected error |
| OFS_DIAGU_CLASS_SYS_DGB_FULL | 0x96 | Diagnostic buffer full |
| OFS_DIAGU_CLASS_SYS_FFB | 0xA8 | 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 acknwoledgement of the alarm (coded on 4 bytes): PLC time and date when the alarm acknwoledgment,

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 fault 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

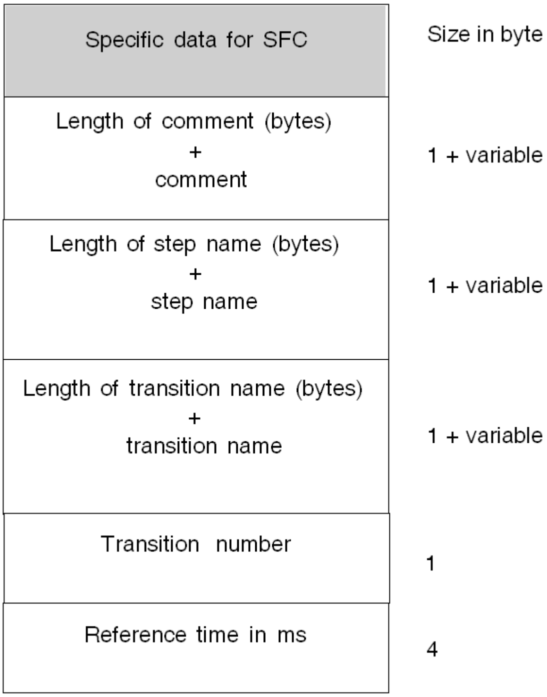
### Specific data types

There are three types of specific data:

- SFC specific data,
- FB specific data,
- System specific data.

### SFC specific data

The diagram below describes the Variable Size Specific data section for SFC:



| Specific data for SFC | Size in byte |
|---|---|
| Length of comment (bytes)<br>+<br>comment | 1 + variable |
| Length of step name (bytes)<br>+<br>step name | 1 + variable |
| Length of transition name (bytes)<br>+<br>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 below describes the Variable Size Specific data section for Function Block:

| Specific data for FB | Size in byte |
|---|---|
| Length of comment (bytes)<br>+<br>comment | 1 + variable |
| Length of instance name (bytes)<br>+<br>instance name | 1 + variable |
| Length of FB type name (bytes)<br>+<br>FB type name | 1 + variable |
| Length of Faulty PIN name (bytes)<br>+<br>Faulty PIN name | 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 Faulty PIN name + PIN name:
  The first part gives the length of the name and then the name of the faulty PIN,
- Status address: an 6 bytes structures for Status address,
- Extra data: not documented part data,

### System specific data

The diagram below describes the Variable Size Specific data section for System:

| Specific data for System | Size in byte |
|---|---|
| Length of comment (bytes)<br>+<br>comment | 1 + variable |
| Length of instance name (bytes)<br>+<br>instance name | 1 + variable |
| 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,
- Extra data: not documented part data,

# 19.3 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's in this Section?

This section contains the following topics:

## Operation from an OPC client

### Reminder about the Diag Buffer

The Diag buffer *(see page 266)* 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 below shows the specific items:

| Service | Item | Type | Access | Value read | Value to write |
|---|---|---|---|---|---|
| Open connection | #DiagLogon | VT_UI2 | READ/WRITE | Viewer or 0xFFFF identifier | zone number |
| Close connection | #DiagLogout | VT_UI2 | READ/WRITE | Viewer or 0xFFFF identifier | not important |
| Read next detected error | #DiagReadNextError | VT_UI1+VT_ARRAY | READ | Error | |
| Error acknowledgment | #DiagAckError | VT_UI2 | WRITE | | Detected error ID number<br>see *Information retrieved by the Diag buffer at the top of the table, page 308* |
| Evolution Status | #DiagReadStatus | VT_UI4 | READ/WRITE | Status0 + Status1 | Status handle |

Type corresponds to OPC standards:

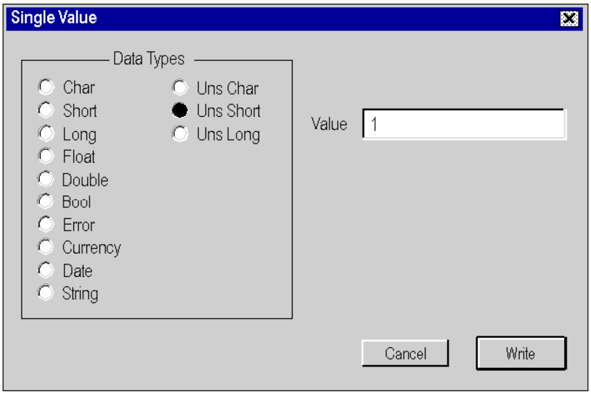
- VT = variable,
- UI1 = unsigned integer on 1 byte,
- UI2 = unsigned integer on 2 bytes,
- UI4 = unsigned integer on 4 bytes,
- ARRAY = table of bytes

## #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:



Single Value

Data Types

○ Char
○ Short
○ Long
○ Float
○ Double
○ Bool
○ Error
○ Currency
○ Date
○ String
○ Uns Char
● Uns Short
○ Uns Long

Value: 1

**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 the value 0040h
  - to monitor zones 2 and 15: write the value 8004h

**Value returned after read**:

- the viewer number is displayed if the connection is open, if not the connection is not established and 0xFFFF 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 |
| 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 the value FFFFh 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, the OxFFFF value 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 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 307)*.

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] = 04h, Var[1] = 05h. The value to write in the item #DiagAckError is 0504h.

**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 307)*.

**Value to write**:

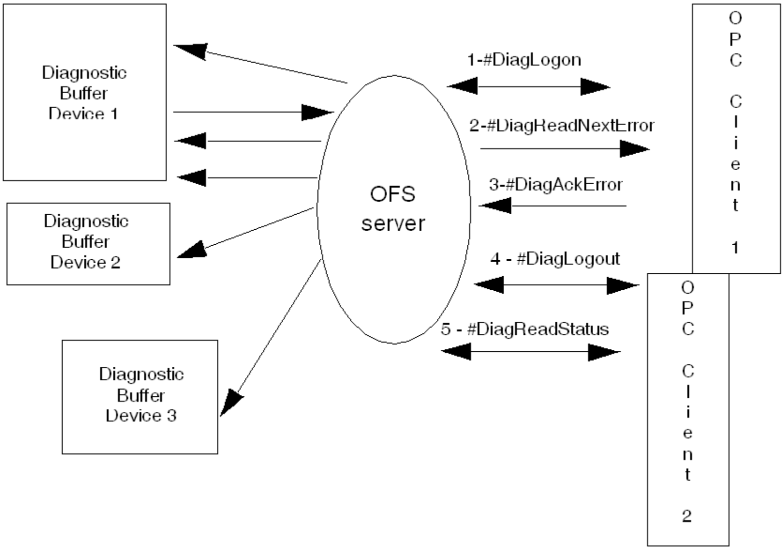
- the value on 4 bytes corresponding to the "Status Handle" zone starting by a read of the highest ranking byte.
  E.g. the value returned in the "Status Handle" zone of the #DiagReadNextError item table is such that:
  Var[8] = 98h, Var[9] = 01h, Var[10] = 76h, Var[11] = 25h
  The value to write in the #DiagReadStatus item is 25760198h or 628490648d.

**Value returned after read**:

- the values of status 0 + status1, taking into account the value of the words from right to left.
  E.g. the value returned is 0010001Dh; status0 value is 001Dh; status1 value is 0010h.

## Description of client operation

The diagram below 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 will be deleted from all viewers once it has been acknowledged on one viewer.

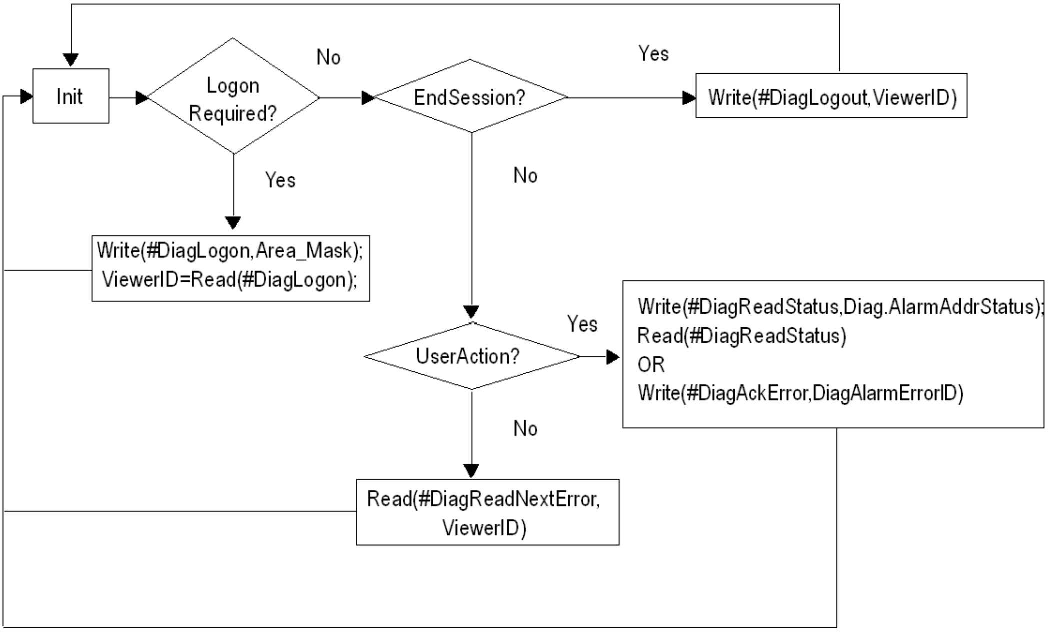
An alarm is deleted from the buffer if:

- the alarm no longer exists,
- all viewers have read the alarm,
- the alarm has been acknowledged (after an acknowledgment request).

# Description of client sequencing

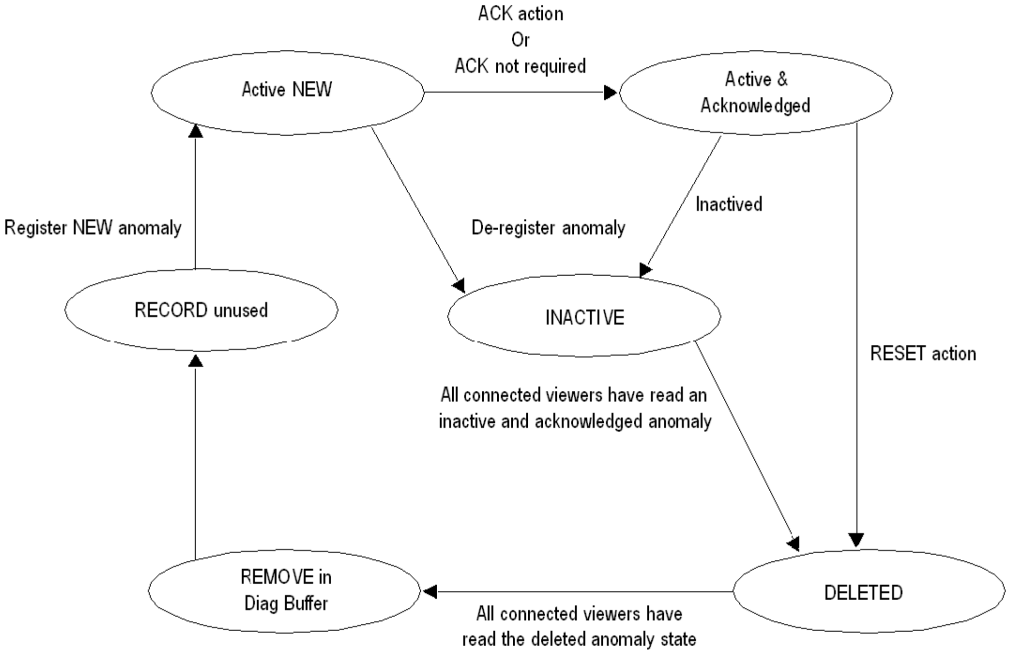
## Description of client sequencing

The graph below shows how and in which order an OPC client uses the specific items for PL7 Diag buffer:


Init
Logon Required?
No
Yes
EndSession?
Yes
No
Write(#DiagLogout,ViewerID)
Write(#DiagLogon,Area_Mask);
ViewerID=Read(#DiagLogon);
UserAction?
Yes
No
Write(#DiagReadStatus,Diag.AlarmAddrStatus);
Read(#DiagReadStatus)
OR
Write(#DiagAckError,DiagAlarmErrorID)
Read(#DiagReadNextError,
ViewerID)

## 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 will be 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 0xFFFFFFFE). The value 0xFFFFFFFF is reserved.

**Example** of the settings of the handle with the C++ test client supplied on the OPC Factory Server CD:

- 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 CD.

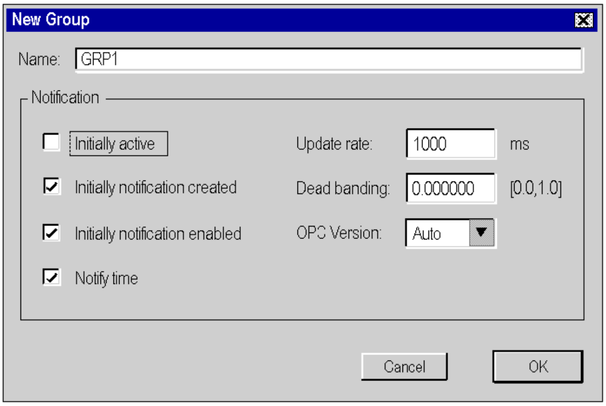
For more information on the OPC client see the OFS Client *(see page 150)*section.

## 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**:



New Group

Name: GRP1

Notification

□ Initially active

☑ Initially notification created

☑ Initially notification enabled

☑ Notify time

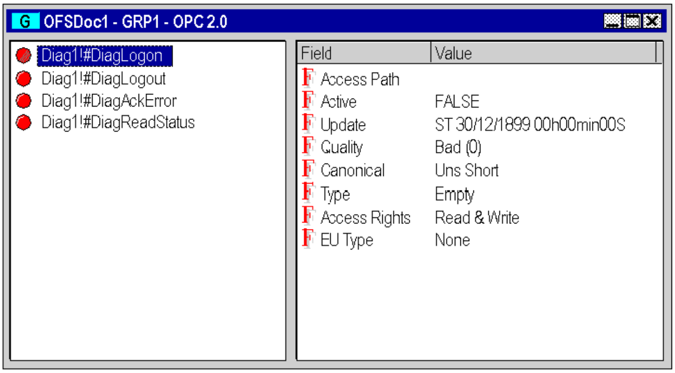
Update rate: 1000 ms

Dead banding: 0.000000 [0.0,1.0]

OPC Version: Auto

1- To connect to the diag buffer, the OPC client must add the #DiagLogon *(see page 296)* 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 297)* specific item to the group. The disconnection will be carried out when the client writes any value in this item.
3- For acknowledgment, the OPC client requires the #DiagAckError *(see page 298)* specific item in the group.
4- To update the Diag Buffer status, the OPC client needs to add the #DiagRead-Status *(see page 299)* specific item.

The following screen shows the installation of specific items:

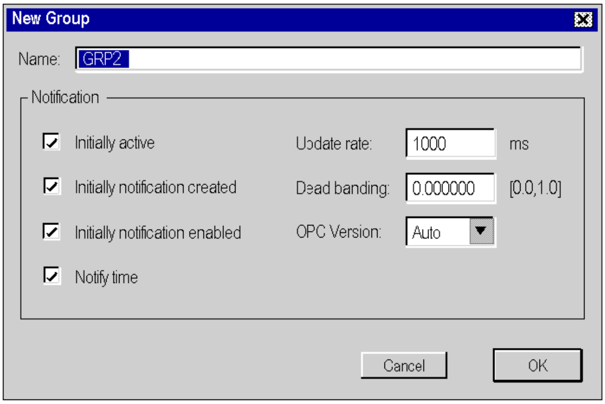


OFSDoc1 - GRP1 - OPC 2.0

- Diag1!#DiagLogon
- Diag1!#DiagLogout
- Diag1!#DiagAckError
- Diag1!#DiagReadStatus

| Field | Value |
|---|---|
| Access Path | |
| Active | FALSE |
| Update | ST 30/12/1899 00h00min00S |
| Quality | Bad (0) |
| Canonical | Uns Short |
| Type | Empty |
| Access Rights | Read & Write |
| EU Type | None |

- **An active group**: this group must be created or activated after actual connection with the #DiagLogon item.



New Group

Name: GRP2

Notification

☑ Initially active — Update rate: 1000 ms

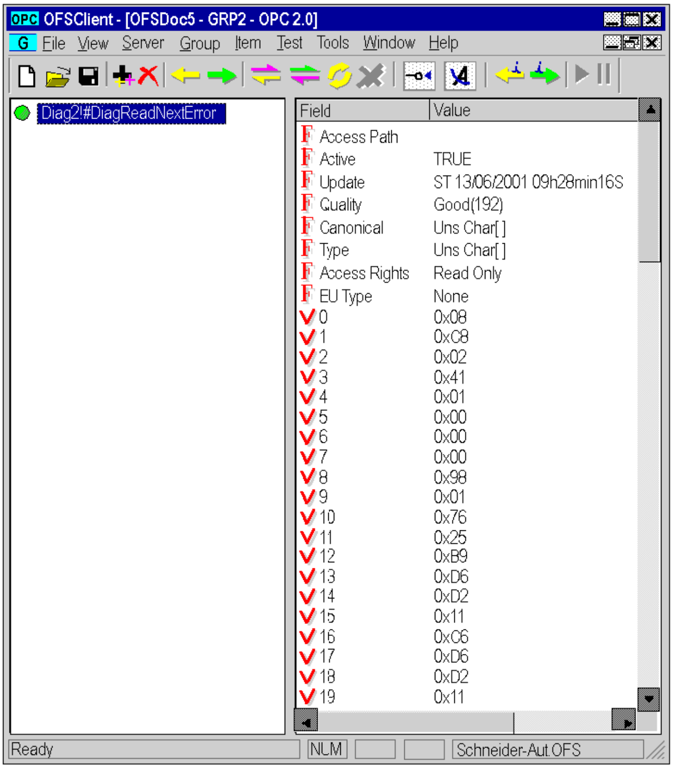
☑ Initially notification created — Dead banding: 0.000000 [0.0,1.0]

☑ Initially notification enabled — OPC Version: Auto

☑ Notify time

To reset the alarms coming from diag buffer, the client needs to add the #DiagReadNextError *(see page 298)* specific item to the group.

The screen below shows a 120 byte table *(see page 307)* in which the detected error *(see page 307)* code of the activated alarm can be found. Each byte represents a specific piece of information:



| Field | Value |
|---|---|
| F Access Path | |
| F Active | TRUE |
| F Update | ST 13/06/2001 09h28min16S |
| F Quality | Good(192) |
| F Canonical | Uns Char[ ] |
| F Type | Uns Char[ ] |
| F Access Rights | Read Only |
| F EU Type | None |
| V 0 | 0x08 |
| V 1 | 0xC8 |
| V 2 | 0x02 |
| V 3 | 0x41 |
| V 4 | 0x01 |
| V 5 | 0x00 |
| V 6 | 0x00 |
| V 7 | 0x00 |
| V 8 | 0x98 |
| V 9 | 0x01 |
| V 10 | 0x76 |
| V 11 | 0x25 |
| V 12 | 0xB9 |
| V 13 | 0xD6 |
| V 14 | 0xD2 |
| V 15 | 0x11 |
| V 16 | 0xC6 |
| V 17 | 0xD6 |
| V 18 | 0xD2 |
| V 19 | 0x11 |

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=11h, V14=D2h, V13=D6h, V12=B9h.

| | 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

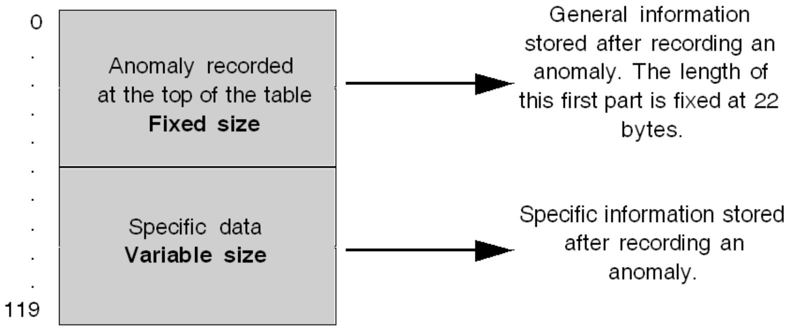
The result is therefore 13h26min57s, 9/07/2001.

## Diag buffer table formats

### Description

The 120 byte table *(see page 304)* (alarm reset after a read on #DiagReadNextError) is structured in the following way:

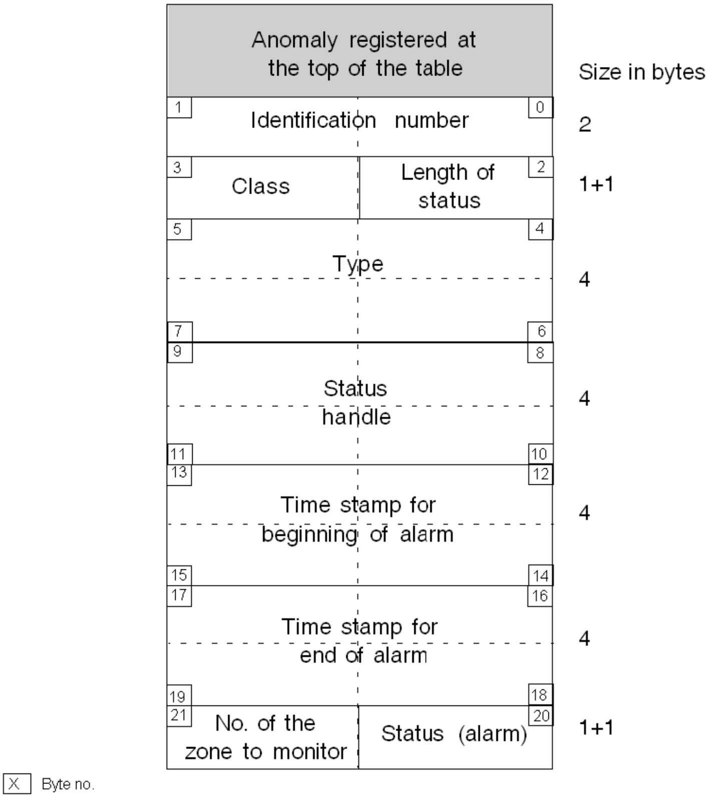
Illustration of the structure of the bytes table:


0
119
Anomaly recorded at the top of the table
**Fixed size**
Specific data
**Variable size**
General information stored after recording an anomaly. The length of this first part is fixed at 22 bytes.
Specific information stored after recording an anomaly.

# Information retrieved by the Diag buffer at the top of the table

## Description

The illustration below 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 below gives the definition of the code retrieved in this byte:

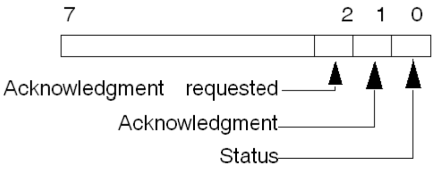
| **Symbol** | **Value** | **Comment** |
|---|---|---|
| OFS_DIAG_CLASS_DFB_EV_DIA | 0x40 | EV_DIA detected error |
| OFS_DIAG_CLASS_DFB_MV_DIA | 0x41 | MV_DIA detected error |
| OFS_DIAG_CLASS_DFB_NEPO_DIA | 0x42 | NEPO_DIA detected error |
| OFS_DIAG_CLASS_DFB_ALARM | 0x43 | ALRM detected error |
| OFS_DIAG_CLASS_DFB_USERA | 0x4A | USER DFB detected error |
| OFS_DIAG_CLASS_DFB_USERB | 0x4B | USER DFB detected error |
| OFS_DIAG_CLASS_DFB_USERC | 0x4C | USER DFB detected error |
| OFS_DIAG_CLASS_DFB_USERD | 0x4D | USER DFB detected error |
| OFS_DIAG_CLASS_DFB_USERE | 0x4E | USER DFB detected error |
| OFS_DIAG_CLASS_DFB_USERF | 0x4F | USER DFB detected error |
| | | |
| **System detected error class** | | |
| OFS_DIAG_CLASS_DFB_SYSTEM_ASI0 | 0x80 | STGENE of ASI_DIA detected error |
| OFS_DIAG_CLASS_DFB_SYSTEM_ASI1 | 0x81 | STSLABS of ASI_DIA detected error |
| OFS_DIAG_CLASS_DFB_SYSTEM_ASI2 | 0x82 | STSLKO of ASI_DIA detected error |
| OFS_DIAG_CLASS_DFB_SYSTEM_ASI3 | 0x83 | STSLNC of ASI_DIA detected error |
| OFS_DIAG_CLASS_DFB_SYSTEM_IO | 0x84 | IO_DIA detected error |
| | | |
| **New characteristic of the PL7v4** | | |
| OFS_DIAG_CLASS_DIAGSYSTEM | 0x85 | system detected error (Task, Arithm) |
| OFS_DIAG_CLASS_SYT_LOCALIO | 0x86 | LOCAL IO detected error |
| OFS_DIAG_CLASS_SYT_REMOTIO | 0x87 | REMOTE IO detected error |
| OFS_DIAG_CLASS_SYT_BUFFERFULL | 0x88 | 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 #DiagRead-Status 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 309*):

| Specific DFB | Size in bytes |
|---|---|
| Length of comment (bytes)<br>+<br>comment | 1 + variable |
| Length of "instantiated" name (bytes)<br>+<br>“instantiated” name | 1 + variable |
| Length of the type of DFB (bytes)<br>+<br>type of DFB | 1 + variable |
| Length of the program address<br>+<br>program address | 1 + variable |

## Definition of the contents of the table

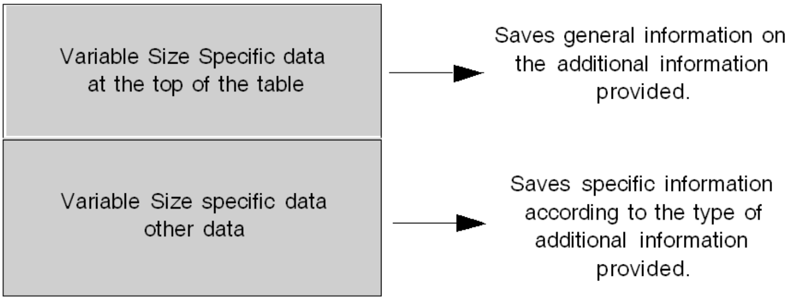
- Length of comments + comments:
  The first part gives the length of the comments and then the DFB message.
- Length of the "instantiated" name + "instantiated" name:
  The first part gives the length of the "instantiated" name then the DFB "instantiated" name.
- Length of file name + file name:
  The 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 309*).

Specific data gives more information according to the class recorded.

Illustration:


Variable Size Specific data at the top of the table
Saves general information on the additional information provided.
Variable Size specific data other data
Saves specific information according to the type of additional information provided.

### 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:

<table>
<tr><th colspan="2">Additional information at the top of the table</th><th>Size in bytes</th></tr>
<tr><td colspan="2">Length of comment<br>(bytes)<br>+<br>comment</td><td>1 + variable</td></tr>
<tr><td colspan="2">Length of "instantiated"<br>name (bytes)<br>+<br>“instantiated” name</td><td>1 + variable</td></tr>
<tr><td>Type</td><td>Size information</td><td>1 + 1</td></tr>
</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.

# Communication

# 20

## Aim of this Chapter

The aim of this chapter is to describe the product’s communication methods.

## What's in this Chapter?

This chapter contains the following sections:

# 20.1 Communication

### Aim of this Section

The aim of this section is to describe the type of communication used by the OFS server.

### What's in this Section?

This section contains the following topics:

# 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 202)* describes the Quality attribute associated with an item.

The OFS server allocates for:

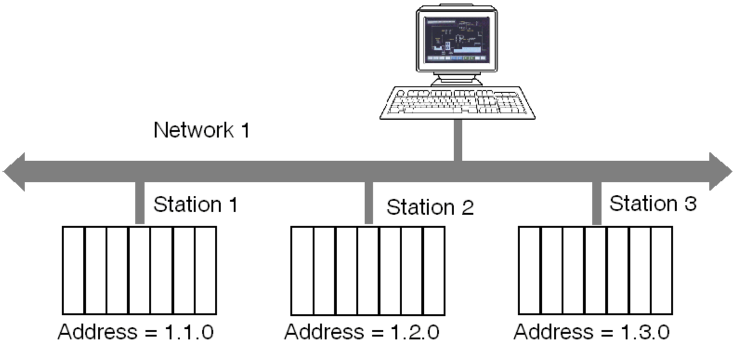
- X-Way: a socket (communication channel) for each driver for PL7 or UNITY-type devices, and up to 16 sockets for XTEL or ORPHEE-type devices
- Modbus Plus: a path to each device (PM), or up to 16 paths to each device (DM)
- TCP-IP (non-XIP) and USB: Up to 16 sockets to each device *(see page 324)*.

**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

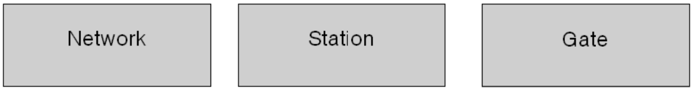
Example of access through a network:



**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.

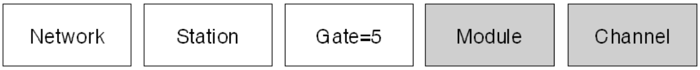
The value "Gate" refers to the communication entity within the station: system server (Gate 0, the most common), the terminal port (Gates 1,2,3), 1K asynchronous server (Gate 7), etc.

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:

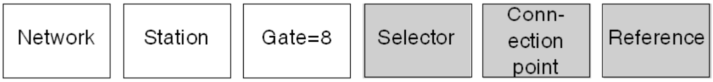


- Module: physical location of the communication module in the rack. Its value must be defined as follows: (Master rack number * 16) + Number of master module.
- Channel: address of the device connected to the communication module. Its value must be defined as follows: (Master channel number * 100) + 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).

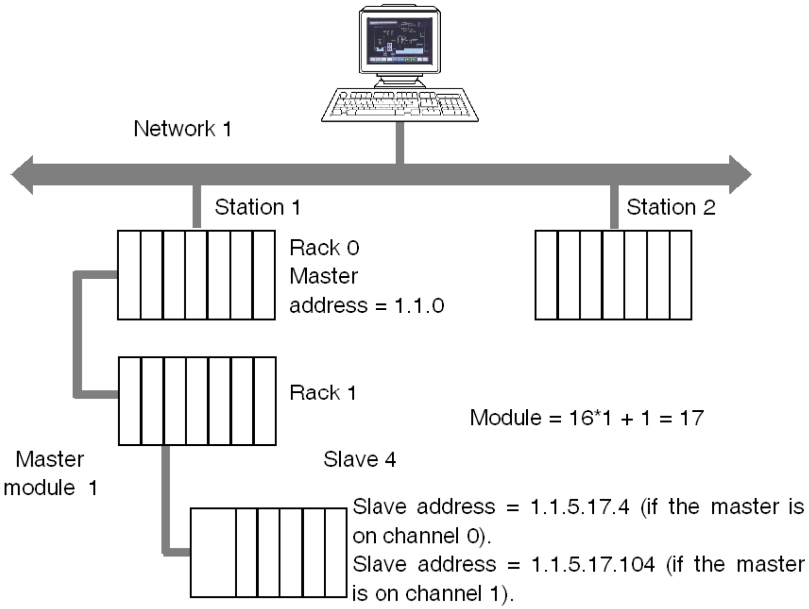
Illustration:



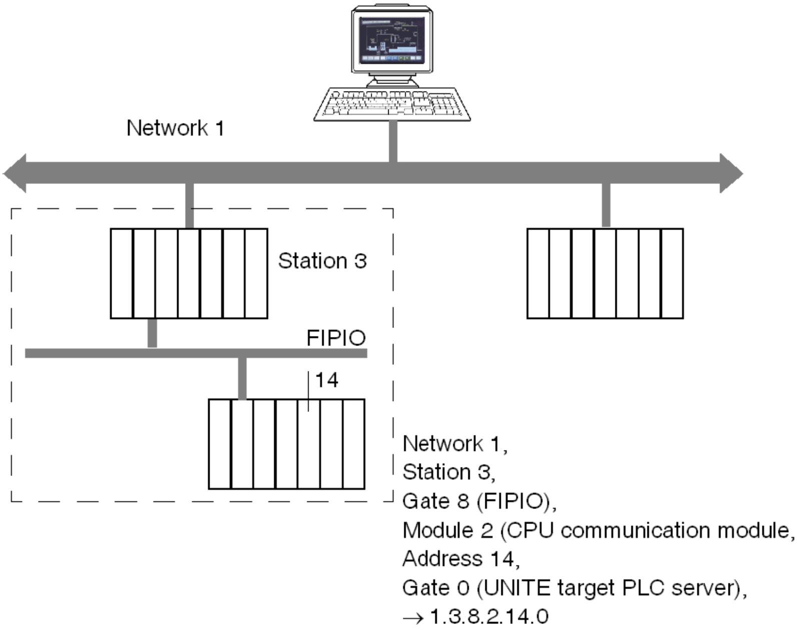
- **Selector**: designates a communication module on the CPU (2) or in a separate module (1).
- **Connection point**: device address, if the destination module is FIPIO. Physical positioning in the PLC rack, if the destination module is a PLC card.
- **Reference**: communication entity in the device (similar to the Gate number).

**Examples**:

5 level addressing:


Network 1
Station 1
Station 2
Rack 0
Master
address = 1.1.0
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:



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 "1K 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 142)*.
- 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.

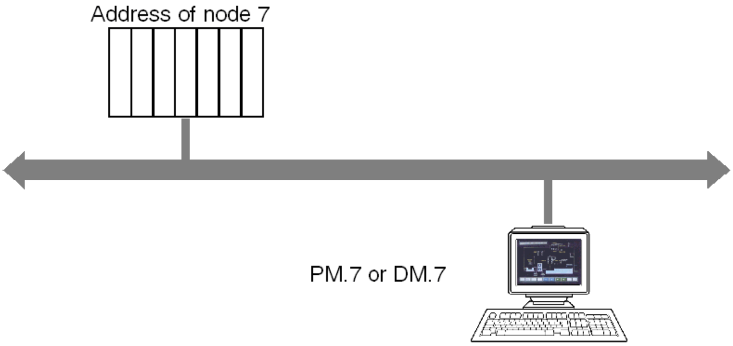
The node number should be used to specify the full path. To access a device without a bridge, only access mode and node are required.

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 96)*.

For example:



# 20.2 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 *(see page 105)* 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.

# Performance

# 21

## 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's in this Chapter?

This chapter contains the following sections:

## 21.1 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's 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 1K (1) | 1016 | 1012 |
| Ethway | 248 | 244 |
| Fipway | 120 | 116 |
| 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:

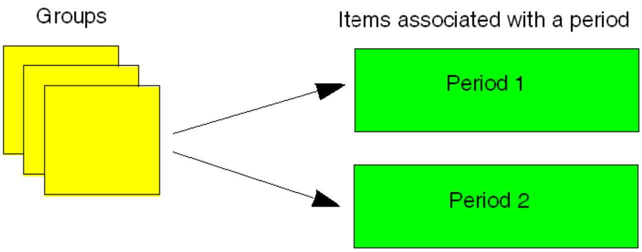
| Communication medium | Read | Write |
|---|---|---|
| Ethway | 1022 | 1016 |

# 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 *(see page 329)* 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

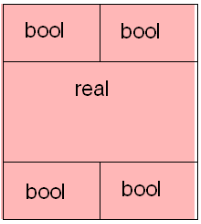
Concept gives you the opportunity to build data structures, made up of members of different types.

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.

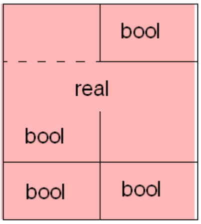
Writing non-aligned fields: when the members are not aligned on the 16 bit boundaries, the writing of one of these members, where this cannot be carried out with a single request, is forbidden.

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 Premium and M340 devices

## General

The addressing of discrete Input/Output modules concerns the following devices: TSX DEY, TSX DSY, TSX DMY, TSX DEZ, TSX DSZ, TSX DMZ as well as devices of the TBX and Momentum families.

The "Read operation" and "Write operation" sections use a type of optimization known as "module optimization".

## 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 Premium and M340 devices

### General

This section concerns all modules not dealt with in the section on addressing of discrete *(see page 332)* Input/Output modules and mainly concerns TSX AEY, TSX ASY, TSX AEZ, TSX ASZ devices as well as periodic exchanges of 64-word table objects from the Fipio agent.

The "Read operation" and "Write operation" sections use a type of optimization known as "channel optimization".

### 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<br>⋮ | 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 332)*" is not performed.

As a result, the channels of discrete I/O modules are addressed using "channel optimization" *(see page 334)*, 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 319)* 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.

# 21.2 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 324)*, 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 194)* 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.

# Developer Guide

# VII

# Advice

# 22

## Aim of this Chapter

The aim of this chapter is to provide developers tips on how to use the OFS product.

## What's 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.

# Appendices

# VIII

# Appendices

# 23

## Aim of this Chapter

The aim of this chapter is to introduce the appendices for this book.

## What's in this Chapter?

This chapter contains the following sections:

# 23.1 Compatibility of the OFS server

## OFS server compatibility

### Definition

OFS is compatible with OPC 1.0A 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 239*).

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.

# 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".

# 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 | Function name | Max. length used | OFS features using the request code |
|---|---|---|---|
| 0x01 | Read Coil Status | 2000 | Read of 0x items |
| 0x02 | Read Input Status | 2000 | Reading 1x items |
| 0x03 | Read Holding Registers | 125 | Read of 4x items and device detection (with 0 reg) |
| 0x04 | Read Input Registers | 125 | Read of 0x items |
| 0x05 | Force Single Coil |  | Write of a single 0x item |
| 0x0F | Force Multiple Coil | 800 | Write of several 0x items |
| 0x10 | 16 Preset Multiple Registers | 100 | Write of any number of 4x items |
| 0x11 | Report Slave ID |  | Read of device operating mode |
| 0x16 | Mask Write 4X registers |  | Write of a byte item located in the 4x area |
| 0x2A | IEC Runtime FC |  | Access to unlocated Concept variables (read/write) |
| 0x5A | Internal Unity request |  | Access to Unity PLC |
| 0x7E | Modsoft FC |  | Starting / Stopping the device |

### UNITE request codes used by OPC Factory Server:

| Request code | Function name | OFS features using the request code |
|---|---|---|
| 0x0F | Identify | Device detection |
| 0x24 | Start | Starting the device |
| 0x25 | Stop | Stopping the device |
| 0x33 | Initialize | Initializing the device |
| 0x36 | Read Object | Read of all items on TSX Series 7, S1000 |
| 0x37 | Write Object | Write of all items on TSX Series 7, S1000 |
| 0x38 | Read Object List | Read of all items on Premium, Micro |
| 0x4F | Read CPU | Device detection and read of the device's operating mode on Premium, Micro |
| 0x5A | Internal Unity request | Access to Unity PLC |
| 0x83 | Write Generic Object | Write of all items on Premium, Micro |
| 0xFA | Mirror | Detection of max PDU size |

# 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 CD 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:<br>If the configuration tool still starts up, retrieve the last alias file using the "get archive" menu.<br>If the configuration tool does not start up, try to copy the back-up file saved in "alias2K.mdb" in the configuration tool directory manually. This operation will overwrite the working database, which is probably corrupted. Then try to restart the program.<br>If you have not backed-up:<br>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. |

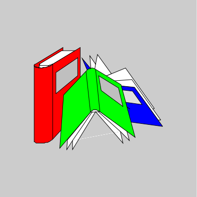
| Component | Anomaly | Remedy |
|---|---|---|
| **Configuration tool On-line help** | On-line help does not work. | On-line help requires at least Internet Explorer v3.02. |
| **Configuration tool** | My new configuration tool parameters are not taken into account. | Confirm the configuration tool and 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.<br>If it isn't: See either the driver or the connections (see appropriate manuals),<br>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 the compatible drivers for OFS V3.33 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 CD). |
| **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:<br>● that you have not inadvertently checked the "X-Tel" option in the properties page.<br>● 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).<br>The parameters which may have an influence on communication performance are:<br>● the number of requests sent in parallel,<br>● 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 CD. |
| **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 running Windows Vista, XP or 2000. | 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 with Windows Vista, XP or 2000. | In order to be used remotely by an OPC DCOM client, the OFS server must be started up using an ADMINISTRATOR account or as an Vista, XP or 2000 utility. 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:<br>Dim WithEvents OpcFactoryServer As OPCServer.<br>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 registry.<br>If you have not defined this option, check that the C:\OPC_SYMB directory exists and that it contains your file.<br>If you have put a space after "=" in the group name, then it must be removed.<br>For example: "grpName= symb.scy" becomes "grpName=symb.scy". |

| Component | Anomaly | Remedy |
|---|---|---|
| **Server**<br>Target device:<br>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**<br>**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**<br>**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**<br>**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**<br>**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**<br>**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**<br>**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**<br>**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:<br>● create the items in an inactive group, then activate the group,<br>● check that 'Verbose' server mode is not selected in order to keep traces to a minimum,<br>● 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

**A**ctive **S**erver **P**age allows a Web site builder to dynamically create web pages. It supports the code written in compiled languages such as byVisual Basic, C++, C #, etc.

## C

### Client application

Software using the primitives provided by a server application, via mechanisms (interfaces) implemented by OLE.

### CLR

**C**ommon **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**

**C**ommon **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**

**C**omponent **O**bject **M**odel: foundations of the OLE 2.0 standard.

## D

**DCOM**

**D**istributed **COM**: COM model distributed over a TCP-IP network.

## F

**FIP**

**F**actory **I**nstrumentation **P**rotocol.

**FTP**

**F**ile **T**ransfer **P**rotocol is the standard internet protocol that is used for exchanging files between computers and the internet.

## G

**GAC**

**G**lobal **A**ssembly **C**ache contains all assemblies necessary for .NET and manages different versions of assemblies.

## H

**Handle**

Single value identifying the object.

**HTML**

**H**yper**T**ext **M**ark-up **L**anguage is the language used to describe Web pages.

**HTTP**

**H**yper**T**ext **T**ransfert **P**rotocol is the protocol used for transferring HTML pages.

## I

**IDE**

**I**ntegrated **D**evelopment **E**nvironment is a program that includes a code editor, a compiler, a detected error analyzer and a graphic interface.

**IIS**

**I**nternet **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**

**J**ava **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**

**L**anguage **C**ode **ID**entifier.

## M

**Multi-clients**

Several client applications simultaneously access the same server application.

## O

**OFS**

**OPC F**actory **S**erver: OLE server for exchanging data with the PLC.

**OLE**

**O**bject **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**

**P**rogrammable **L**ogic **C**ontroller: programmable controller (industrial).

**Primitive**

OPC function

## R

**RCW**

**R**untime **C**allable **W**rapper: The main function is to group calls between .Net client and the non-managed COM object.

**RDE**

**R**ead **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**

**S**imple **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.

**SP**

**S**ervice **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**

**U**niversal **N**aming **C**onvention.

## V

**VB**

**V**isual **B**asic: consumer language supporting OLE Automation.

**VBA**

**V**isual **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**

**W**eb **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