Interoperability
for your devices
and software applications.

OpcDbGateway and SAEAUT Universal OPC Server User's guide

Application integration, data collecting, monitoring, control, reporting, alarm handling and data archiving

by SAE-Automation, s.r.o. (Ltd.)

OPcDbGateway - configuring and executing of data exchange between various sources - OPC servers, OPC client applications, device communication drivers, data handling, monitoring, storage into process databases, trends, alarms, logging,
reports, integrating to SOA.

SAEAUT Universal OPC Server - creating of own applications with the OPC server communication interfaces more easily as by standard SDK's for development of OPC servers.

Interfaces: OPC DA (Data Access) 3.0x, 2.05, 1.0 a OPC AE (Alarm and Events) 1.10., OPC UA 1.01 and OPC XML DA 1.01.

OpcDbGateway and SAEAUT Universal OPC Server

http://www.saeautom.sk/products/opcdbgateway/
sae-automation@saeautom.sk

Table of Contents

OpcDbGateway and SAEAUT Universal OPC Server

Part I

1 Products overview

Products OpcDbGateway and SAEAUT Universal OPC Server have common core parts, for they are described in common user's guide.

SAEAUT Universal OPC Server was derived from OpcDbGateway by removing of functionality to connect to external OPC servers and all functionalities related to process databases.

Both OpcDbGateway and SAEAUT Universal OPC Server are delivered with a DDE client which has independent user's guide.

OpcDbGateway is delivered with the product SAEAUT SMS Service that has also independent user's guide.

OpcDbGateway runtime and configuration applications are components of the SCADA HMI product SAEAUT SCADA™ that has own user's guide.

Other software products from SAE - Automation, s.r.o., (Ltd.) can be also used in integrated applications. Please see a table overview.

Above mentioned products can be bought also trough e-shop.

1.1 OpcDbGateway

OpcDbGateway enables **integration of software applications** to **collect process and visualise data** from external devices, applications and data sources.

It enables parallel running of tasks for data collecting and processing, working with one or more process databases using built-in database commands or SQL queries, periodical or single-shot launching of external programs, scripts and database stored procedures at specified time or according to specified conditions, generating of reports, alarms, log-files, sending of SMS and E-mails, enabling of OPC tunnelling and OPC redundancy. It can be used also as universal OPC client/server.

It consists of a **configuration and a runtime application**.

The **configuration application** increases integrator's productivity by substituting of laborious programming with configuring. In the same time, it keeps flexibility as the customer programming modules-dll's (e.g. digital filters, communication drivers, regulators) can be used as an **enhancement of the configurable functionality**. Productivity is leveraged also by **debugging tools and configuration wizards.**

The **runtime application** has three main parts:

- OPC DA client to communicate with external OPC servers.
- OPC server to provide processed data over OPC DA, AE, OPC UA binary and OPC UA, OPC XML DA web services for external applications or a graphical user interface for different client applications.
- Soft controller that provides executing of configurable commands and uses functionality

implemented in enhancing custom dll's for data and tasks processing

Functionality of the runtime application can be enhanced with:
- DDE client (licensed separately).
- SAEAUT SMS Service (licensed separately) - to configure and provide alarming and messaging over SMS and short E-mails
- User's enhancing DLL's

Product installation package contains also:
- OPC UA 1.01 a o OPC XML DA 1.01 wrappers
- web application SAEAUT OPC WebView to see address space of the internal OPC server in web browser.

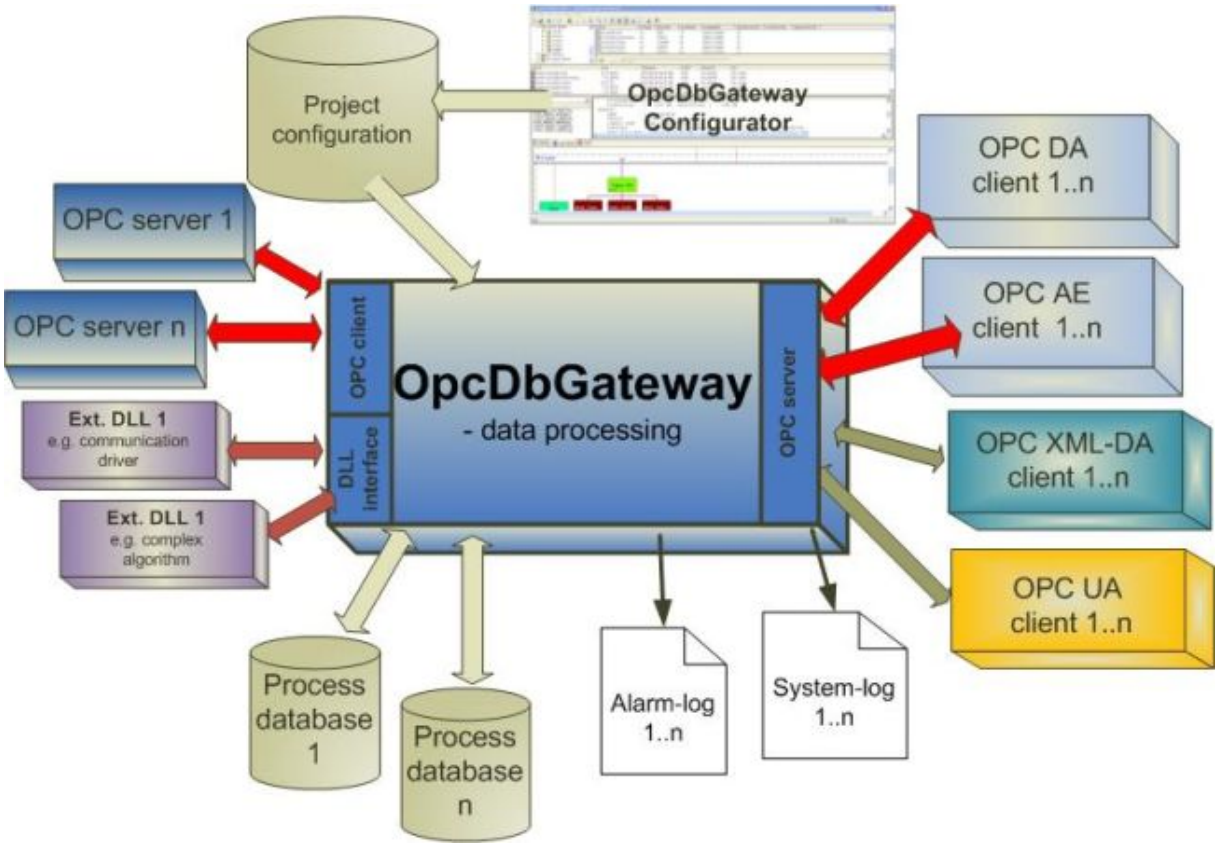


Figure: *OpcDbGateway - architecture and usage.*

Related links on the web

OpcDbGateway on the web
OpcDbGateway blog
Buy on line
On line help
OpcDbGateway - documents, videos, white papers and downloads
SAEAUT UNIVERSAL OPC Server - documents, videos, white papers and downloads
DDE client for OpcDbGateway and SAEAUT UNIVERSAL OPC Server

Related articles

The first start of OpcDbGateway
New features of OpcDbGateway
The OpcDbGateway server architecture
.

1.1.1 How it functions

Using **configuration application** you can configure connections with external data source, devices and applications. **Mapping** of connections to them is easy and efficient due to different **software wizards**. Configuration application enables configuring also data processing, logging and alarming. You can enhance configurable functionality for different application domains using **own program modules (enhancing DLL's)**. **Configuration application** is used also for debugging, checking the created configuration and viewing log files, alarms and content of connected databases. By configuring, a control flow diagram is created that can be used by editing of the configuration. Configuration can be also exported / imported as XML file.

Runtime application (without user interface) executes activities according to configuration created by configuration application. (In case that you have created more configurations, the one that has to be used must be activated using configurator) .
Runtime application is implemented as an OPC server that can run **local**, as **Windows NT Service**, or ***remote***. It provides **periodic functionality (like in PLC) and also event based functionality.**

1.1.2 Usage - overview

OpcDbGateway can be used for:

- integration of applications,
- data logging to files and databases (messages parameterised by actual values of memory operands),
- alarming
  - proprietary (with alarm history) based on memory operands,
  - acording to the OPC AE standard
- creating of own configurable applications with OPC Server/client interfaces
- reporting,
- scheduled starting external tasks amd applications
- data bridging
- tunneling
- data preprocessing

1.1.3 Integration of Applications

OpcDbGateway enables integration of applications as follows:

- providing of **internal memory space** where data from external devices, applications and databases can be **mapped** and that can be used for **configured or programed internal data processing**,
- providing of means and **software wizards for fast and easy mapping** of big amounts of data from external devices, applications and databases to internal memory space:
  - wizards for mapping of address spaces of external OPC servers to memory operands and to OPC items of the internal OPC server
  - wizard to connect to databases,
  - configuring of database tables and creating them on connected databases,
  - wizard for mapping existing database tables to the configurations,

  - historic trend wizard
- providing of one shot or periodical starting of external applications with posibility to configure their command line parameters,
- scheduling of **SQL queries** and **database procedures,**
- **integrating own programing modules** or third party modules - dll's, ActiveX so that they become part of configurable functionality
- alarming from different external sources **over memory operands** and **according to OPC AE standard**
- logging from different external sources over memory operands as well as logging of internal functionality

Related links
Integration of applications effectively.OpcDbGateway – configuring and programming, overview

1.1.4 Communication and data processing

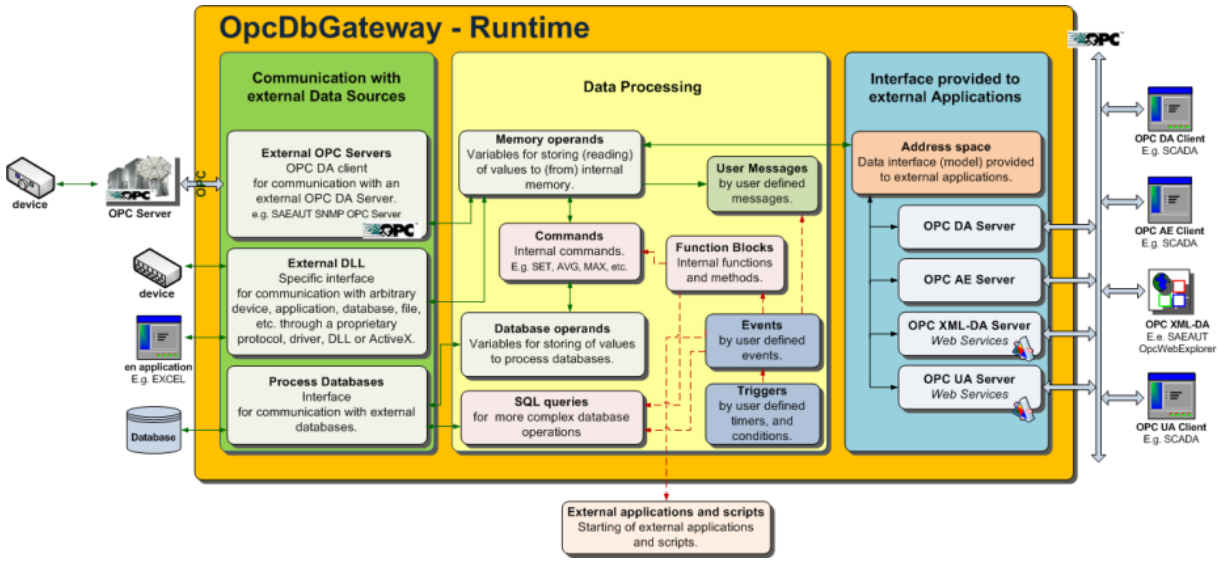


Figure: *Communication with external devices and applications and data processing in the OpcDbGateway runtime application.*

The figure above shows that OpcDbGateway runtime application provides
- *Communication with external Data Sources*.
- *Data Processing*.
- Client *Interface provided to external Applications*.

Communication with external Data Sources
- OPC DA Servers - according to OPC DA 3.0 or OPC DA 2.05. standards
- External DLL - it enables to create a specific DLL library which functionality and data can be shared with OpcDbGateway and SAEAUT Universal OPC Server.
- Process Databases - it enables to connect a process databases via universal interface ODBC or other database drivers installed on the same computer as runtime application.

Data Processing
OpcDbGateway enables preprocessing off data. Data from external devices, applications and program modules (enhancing DLL's) are mapped to ***memory operands***. Memory operands are used as arguments for **configurable *commands***. In case that application is stopped, values of memory

operands are saved to configuration database and can be used by restart of runtime application (see data persistency).

Data from databases (table fields and table columns) can be **mapped to *database operands*** that can also be used as arguments for configurable database commands. When working with databases using only ***SQL queries***, mapping to database operands is not necessary.
Commands are organized in ***function blocks***. Function blocks can be called conditionally or without a condition from cyclically (alike as in PLC) executed special function block *MAIN*, one time executed function blocks *START, STOP, RESTART* or as ***events*** initiated by ***triggers***. Triggering conditions for triggers can be a **value of memory operand** or **time**. Events can be synchronous – executed synchronously (synchronised with the period of the sync. controller and executed in the same thread as the function block MAIN) or asynchronous running in distinct asynchronous thread. External applications can be started as asynchronous events.

Interface provided to external Applications
The OpcDbGateway and SAEAUT Universal OPC Server provides various standard interfaces for accessing data as follows:

- OPC DA Server - it provides interface according to the following standards OPC DA 3.0 and OPC DA 2.05,
- OPC AE Server - it provides interface according to the following standard OPC AE 1.10,
- OPC XML-DA Server - it provides interface according to the following standard OPC XML-DA 1.01,
- OPC UA Server - it provides interface according to the following standard OPC UA 1.01,

OpcDbGateway runtime provides also functionality of two alarm systems:

According to the OPC AE standard – alarming is bound to the OPC items of the internal OPC server Proprietary alarm system with alarm history bound to memory operands.

1.1.5 Installed software and examples

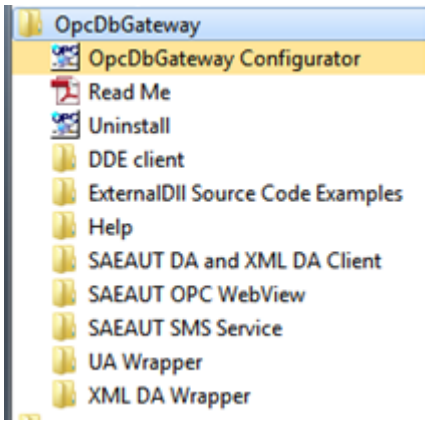


Figure: Start menu for OpcDbGateway

Not all parts of the product OpcDbGateway are installed within initial installation process. In the first step, following parts are installed:
- OpcDbGateway - runtime
- OpcDbGateway - configuration application
- DDE client for OpcDbGateway and SAEAUT UNIVERSAL OPC Server - delivered as dll's for runtime and configuration applications (licensed independently)

Next applications can be installed from start menu (figure above):

- OPC UA Wrapper
- OPC XML DA Wrapper
- SAEAUT OPC DA, XML DA client
- SAEAUT SMS Service - (enables sending of alarms and parameterizable messages using individual and group SMS and E-mails)
- SAEAUT OPC WebView™ is a web application enabling to visualize data from different devices and data sources in a web explorer by unified way. It can be used as a template for complete visualisation application.

The folder **\MyDocuments\OpcDbGateway\Examples** contains complete projects source codes (*Example1* for Visual Studio 2005 and *Example1* for Visual Studio 2012) of the enhancing **enhancement Dll's.**

After installing OPC UA Wrapper, in the root of the start menu will be installed - OPC Foundation\UA SDK1.01\UA Configuration Tool and also SAEAUT OPC UA access client\ SAEAUT Data access client. In case that you have installed OPC UA functionality together with SAEAUT Universal OPC Server or SAEAUT SNMP OPC Server it is not necessary to install it again. It is enough to set UA wrapping of the OpcDbGateway using UA Configuration Tool.

1.2 SAEAUT UNIVERSAL OPC Server

It enables **creating of own applications with OPC server interface** according to the OPC DA (Data Access) 3.0x, 2.05, 1.0 a OPC AE (Alarm and Events) 1.10. more easily comparing to current SDK's for development of OPC servers. Detailed knowledge of the OPC technology is not necessary. Also, a development of own configuration application for the OPC server is not needed.

Product installation package contains also OPC UA 1.01 a o OPC XML DA 1.01 wrappers. Both standards enable communication using of web services. Newest OPC UA standard enables also more powerful communication using binary communication over TCP/IP. The OPC UA advantage is also that it does not use proprietary DCOM technology.

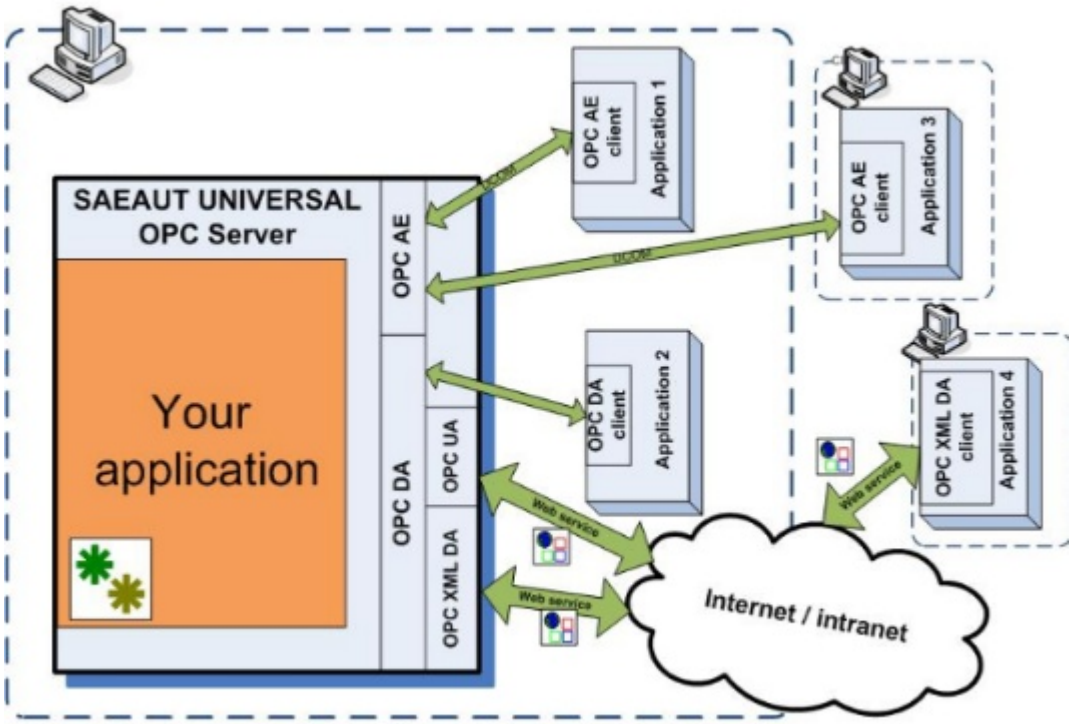


Figure: *Usage of SAEAUT UNIVERSAL OPC Server*

Own application specific functionality can be put into the product by combination of configuring and programming.

Your application cooperates with the ready made OPC server over shared memory by using so-called memory operands. Different functions using these operands can be configured using delivered configuration application. Product can be completed with dynamic linked libraries. They can be programmed using standards software development tools (eg. MS Visual Studio 2012). This way created dll's can be later used as part of the standard configurable functionality of the product.

Creating of applications by configuring enables higher productivity by application creating as programming. It is possible to use configurable arithmetic, logical and comparison commands as well as command for creating of parameterizable messages saved to log-files.

Configuring application offers a few possibilities to verify created configuration and to debug runtime application. It contains also built-in OPC client.

Likeness with OpcDbGateway

Those who already worked with our application OpcDbGateway will be immediately able to work also with the SAEAUT UNIVERSAL OPC Server. From the point of view of configuration and integration there is difference that OPC UNIVERSAL Server runtime **does not contain internal OPC client and none functionality related to process databases**. The functionality to connect external dll's has been slightly enhanced. New function to access the memory operands area and functionality to notify start and ending of the runtime application has been added.

Related articles

SAEAUT Universal OPC Server - Inter operability for your applications

The first start of OpcDbGateway
New features of OpcDbGateway
The OpcDbGateway server architecture
www.saeautom.sk/en/products/opcdbgateway

1.2.1 SAEAUT Universal OPC Server - concepts

OPC servers intermediate access to data from different data sources, applications and devices over OPC items placed in their address spaces to OPC client applications. An application integrator uses a **configuration application** for defining of name, data type and other information for OPC data items and places them within a tree structure of the address space. Saved configuration of the address space uses **runtime application** of the OPC server. OPC client can have possibility to browse address space for OPC items and choose some for its needs.

In the SAEAUT UNIVERSAL OPC Server, data that are accessible as OPC items are provided by your application. It saves them to a non-structured shared memory of so-called memory operands. Operations above memory operands can be configured using configuration application of the SAEAUT UNIVERSAL OPC Server, eventually, their values can be affected from your program modules which are connected to the SAEAUT UNIVERSAL OPC Server runtime as dynamically linked libraries (enhancing dll).

Your application is usually created as **combination of the programmed and configured functionality**. For easier applications, the whole application functionality can be only configured. If you create own dynamically linked libraries according to the defined rules, they can be used also in future as configurable modules of the system. As the configuring is usually less laborious than programming, the productivity of new application creation is higher.

1.2.2 Features

- OPC DA (Data Access) 3.0, 2.05, 1.0 a OPC AE (Alarm and Events) 1.10 specifications implemented
- Installation package is enhanced with OPC UA 1.01 wrapper, enabling possibility to access data from OPC server according to the newest OPC Unified Architecture (UA) standard using web services or binary data over TCP/IP.
- Installation package is enhanced with OPC XML-DA 1.01 wrapper enabling communication over web services
- Your configuration can be created by user friendly configuration application containing:
  - built-in OPC DA client,
  - built-in graphical browser (editor) of the configuration
  - browser of the system log-files
  - built-in configuration verifier with context dependant finding of errors
  - installation package contains easy immediately running demo
  - runtime application enables logging of internal functionality with definable depth
  - runtime application enables monitoring of actual status using system variables over OPC DA interface
  - the „Find" function enabling looking for text expressions in an actual configuration by user friendly way.

1.2.3 Usage examples

1. OPC DDE server
Although OPC is thought of more powerful communication technology, the DDE communication is still used as well. It can be used for example for communication with the application MS Excel. Using SAEAUT UNIVERSAL OPC Server, you can create OPC DDE Server functioning as gateway between

applications with DDE and OPC communication. DDE communication driver will be placed within your dll. Functions from this dll will be called from the SAEAUT UNIVERSAL OPC Server runtime core using configurable commands Call. The sequence of the function calls can be controlled by configured status automat.

2. I/O OPC Server
Vendors of the I/O modules for PC delivers often a communication ActiveX or dll to their modules. Using this software within your enhancing dll, you can create OPC server for this module easily.
A communication with this module will be then possible trough OPC XML DA or OPC UA also over Internet. Except of this, data from this module can be processed on different ways, for example to filter, evaluate, generate events and alarms, to create parameterised messages containing actual values, or to start external applications depending on actual values of the variable from I/O module. If relatively stable communication functionality will be placed within enhancing dll and the functionality of a next data processing will be configured, you can gain very flexible and easily configurable system working with data from the I/O module. You can use many configurable functions, The writing of actual values to log-files within parameterised messages is one of them.

1.2.4 Installed software and examples

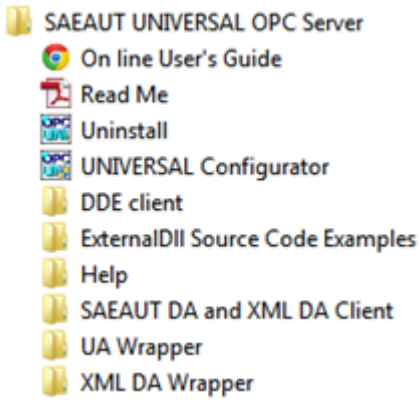


Figure: *Start menu of the SAEAUT UNIVERSAL OPC Server*

Not all parts of the product SAEAUT Universal OPC Server are installed within initial installation process. In the first step, following parts are installed:
- SAEAUT Universal OPC Server - runtime
- SAEAUT Universal OPC Server - configuration application

Next applications can be installed from start menu:
- OPC UA Wrapper
- OPC XML DA Wrapper
- OPC DA, XML DA client

The folder **\MyDocuments\SAEAUT Universal OPC Server\Examples** contains complete projects source codes (*Example1* for Visual Studio 2005 and *Example1* for Visual Studio 2012) of the enhancing **enhancement Dll's.**

Installed examples

- **Enhancing Dll examples**
  - There are two functionally identical example projects for the enhancing Dll for MS Visual Studio 2005 and MS Visual Studio 2005. They enable testing of functionality implemented within function DoProcessIO and called configurable

command CALL DLL and in the same time testing of functionality of coordinating runtime and enhancing dll over commonly used memory operands.

- **Configurations**
  - Common - ExampleConfiguration.ODG – if DDE functionality is **not installed** this configuration is opened by the first start
  - For DDE client for OpcDbGateway and SAEAUT UNIVERSAL OPC Server:
    - DDETestBook.ODG – it enables test the DDE functionality
    - SystemToExcel.ODG – it enables (if DDE functionality is installed this configuration is opened by the first start)
- **Databases** – used with configuration ExampleConfiguration.ODG
  - DatabaseExample.mdb for ver. x_64 uses access over MS JET driver
  - DatabaseExample.accdb for ver. x_86 uses access over MS ACE.OLEDB driver
- **XLS files –** they are used for testing of the DDE client functionality
  - **OPC_Server.xls –** used with SystemToExcel.ODG
  - **DDETestBook.xlsx -** used with DDETestBook.ODG

1.3 Versions - overview

VER 5.

Version 5.03.0.3

- **Configuring application**
  - Added manifest – related to correct working with UAC
- Runtime application
  - Changes related to using as SAEAUT SCADA Server
- Installation
  - **Common installation file for x86 an x64 OS platforms**
  - Common installation script for more products
  - Improvements related to working with UAC

Version 5.01.0.8

- **Configuring application**
  - Changed alarm related dialogs
- Examples
  - Added alarms to the configuration for ExampleConfiguration.ODG
  - Added event to start web explorer with page containing info about demo configuration and other links
  - Added example of historical trend

Version 5.01.06

- **Configuring application**
  - New dialogs for configuring of access to the external OPC servers **– more OPC groups can be created for every external OPC server**
  - Better performance by configuring of OPC items of non-responsive or slow responsive external OPC servers
  - Monitoring view enables not only reading but also writing of the OPC items
  - Mapping of tables on process databases to the OpcDbGateway configuration
  - Creating of database tables on process database according to the table in the OpcDbGateway configuration
  - Output view is opened in individual window – to be able to watch log files in on line mode better. The Checker view and Find view are still in the main window
  - New tabs in Output View: Alarm viewer and Table query viewer
  - New configuration structure (some elements as e.g. triggers cannot be automatically updated from older configurations but can be updated by hand.)

- New enhanced trigger functionality and new configuring dialog box
- New dialog box for events
- New dialog box for commands
- New dialog for the Sync. Controller features
- New dialog box for memory operands – settings for „Memory size and „Persist data" have been moved there from the dialog box for Sync. Controller
- Interconnecting of OPC Item with Memory operand using common ID enables easier changes in configuration
- Commands within tree view of the function block are numbered and ordered according to the command Nr.
- New configuration database is created with structured folder for system variables in the address space of the internal OPC server to gain better overview

- **Runtime application**
  - Better keeping of the synchronous controller period
  - Improved trigger functionality
  - Improved performance – reduced time for executing of some commands
  - Better cooperation of the runtime core with enhancing dll's
  - Asynchronous logging – reduced impact on the synchronous controller period
- **Examples**
  - **Enhancing Dll examples**
    - There are two functionally identical example projects for the enhancing Dll for MS Visual Studio 2005 and MS Visual Studio 2005. They enable testing of functionality implemented within function DoProcessIO and called configurable command CALL DLL and in the same time testing of functionality of coordinating runtime and enhancing dll over commonly used memory operands.
  - **Configurations**
    - Common - ExampleConfiguration.ODG – if DDE functionality is **not installed** this configuration is opened by the first start
      - Added alarms to the configuration for ExampleConfiguration.ODG
      - Added event to start web explorer with info about example configuration and some links
    - For DDE client for OpcDbGateway and SAEAUT UNIVERSAL OPC Server:
      - DDETestBook.ODG – it enables test the DDE functionality
      - SystemToExcel.ODG – it enables (if DDE functionality is installed this configuration is opened by the first start)
  - **Databases** – used with configuration ExampleConfiguration.ODG
    - DatabaseExample.mdb for ver. x_64 uses access over MS JET driver
    - DatabaseExample.accdb for ver. x_86 uses access over MS ACE.OLEDB driver
  - **XLS files –** they are used for testing of the DDE client functionality
    - **OPC_Server.xls –** used with SystemToExcel.ODG
    - **DDETestBook.xlsx -** used with DDETestBook.ODG
- **Help**
  - Access to the online help was added to start menu of the application

Ver. 4.

- OpcDbGateway can run also as Windows service
- Added DDE client functionality with easy and advanced configuring

Ver. 3.17

- Added On/Off switch possibility to display properties by browsing address space. It can be set in Configurator/Tools/Options.
- Delivered with **SAEAUT OPC WebView™.**

OpcDbGateway and SAEAUT Universal OPC Server

Part II

2 First steps

Application activation

The applications will be run in demo mode till software license key(s) will not be entered. To see how to activate applications (for running in full functionality mode), please lunch the *Read Me* from the start menu. You can test full applications functionality also in demo mode, but they must be restarted always after 1 hour.

Launching of configuration applications

After installing OpcDbGateway or SAEAUT UNIVERSAL OPC Server, please open Start menu and launch UNIVERSAL Configurator or OpcDbGateway Configurator.
An active demo configuration will be opened and checked.

OpcDbGateway and SAEAUT Universal OPC Server are delivered with 3 demo configurations that can define their demo functionality.
In case that you have chosen installing DDE client, the easy configuration *SystemToExcel.ODG is* activated and so runtime application will run according to this configuration. If you have not chosen installing DDE client, a more comprehensive configuration *ExampleConfiguration.ODG* will be activated. Of course, you can chose which configuration has to be activated using Configurator from START menu and chose from main menu *File->Open*. You can create also own configuration choosing *File->New*.

Launching of runtime applications

The runtime application can be installed as Windows NT service or as a standard executable. The runtime application is implemented as OPC DA server and so, if not running as Windows NT service, it must be launched by an OPC DA client.

There is a **built in OPC client in the Configurator** that can be used **for starting** (in real deployments it will be another application e.g. HMI with the OPC client interface) in both cases. However, to stop it using Configurator, the runtime must be installed as standard executable. Windows service must be stopped by standard means of the operating system or from start menu using right mouse taste click on e.g. *SAEAUT UNIVERSAL OPC Server->UNIVERSAL Server as Windows Service->Stop service(as administrator)* and choosing *Run as Administrator* from context menu.

In the Configurator, start / stop of the OPC client can be done from main menu View ->Monitor View or from toolbar using icon with eyeglasses as shown in the Figure 1.

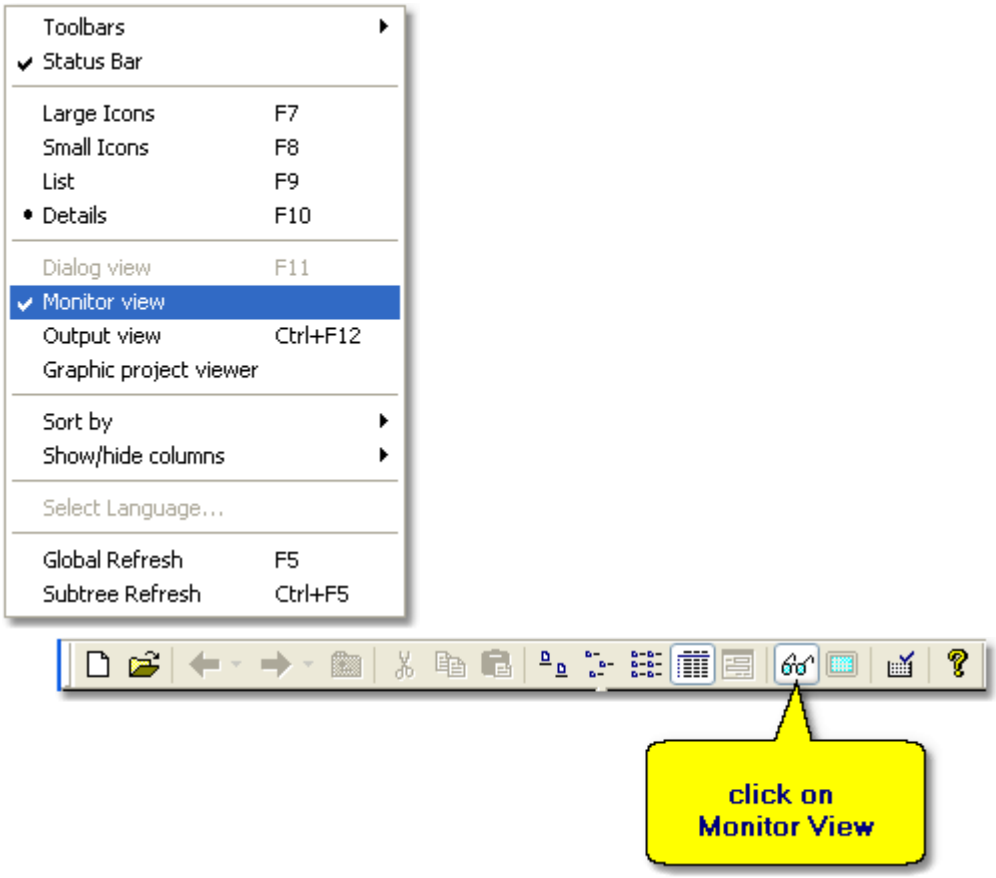


Figure 1 *Launching Monitor View (view for OPC DA client in Configurator)*

Reading/writing system variables

After that, monitor view Figure 2 will be started. First you will see none OPC items in the monitor view. To watch OPC items from a folder of the OPC server address space, please, click on the folder that you want to watch in the tree view. In every demo configuration (and even in new created configuration) you can find folder System where system variables for monitoring and control of the runtime application functionality.

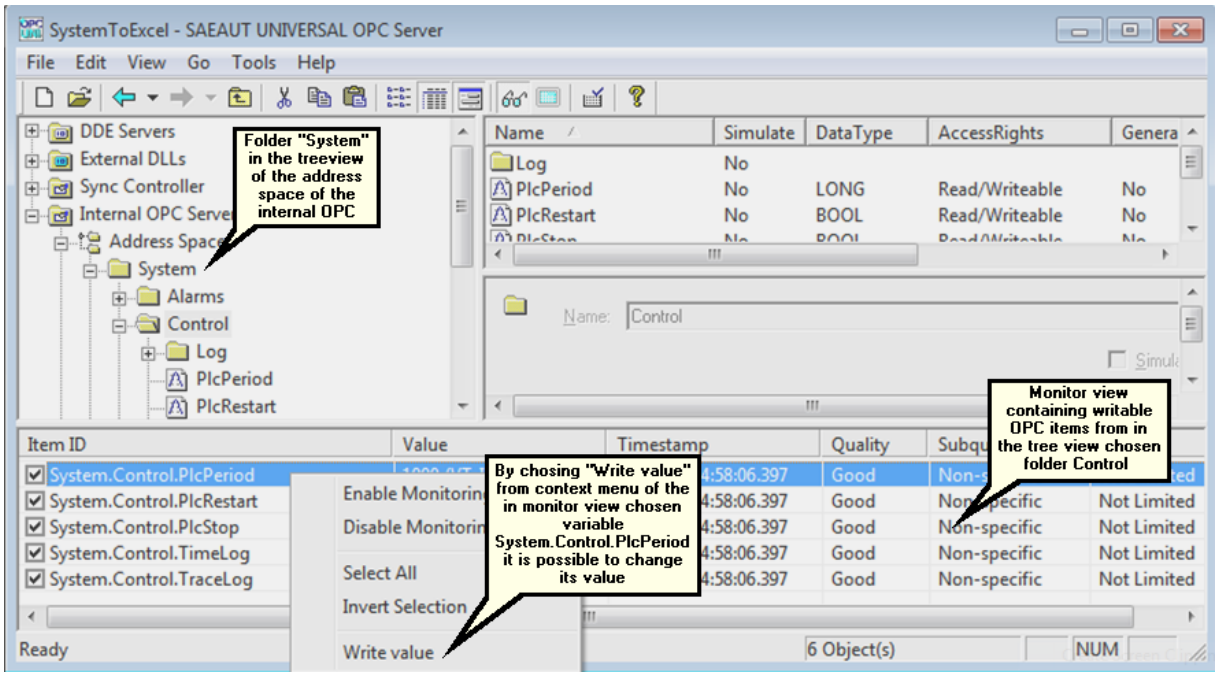


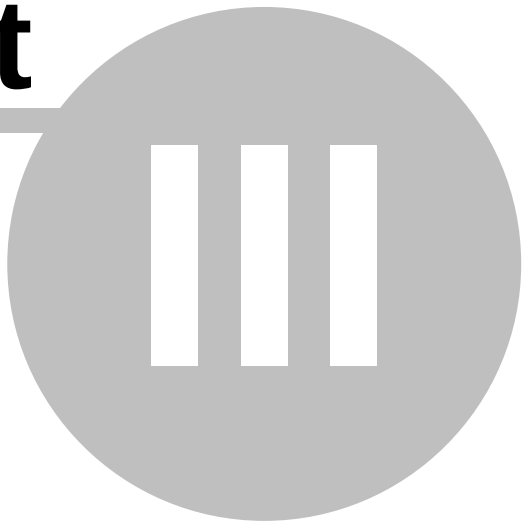
***Figure 2**. Monitor view with variables from the folder System->Control of the address space of the internal OPC server of runtime application*

In the Figure 2, it is shown how to change a value of the system variable *PlcPeriod* from the folder *Control.* If we overwrite this value e.g. from 1000 to 2000ms, the period of the synchronous controller will be changed. Real measured period can be seen in the variable System.Status.PlcPeriod.
Alike way, we can change e.g. also level of the system logging using variables *System.Control.TimeLog* and *System.Control.TraceLog.*

OPC items of the internal OPC server are mostly mapped to memory operands. In case of system variables, mapping is provided only if user needs to use system variables in own application (configuration).

OpcDbGateway and SAEAUT Universal OPC Server

Part



3 Technical parameters

- OPC Interface,
- System requirements,
- Database access,

3.1 System requirements

- Computer/Processor: Pentium II -compatible CPU
- Memory: at least 128 MB (recommended 512 MB)
- OS x86 or x64: MS Windows XP, Vista, Windows 7, Windows 8, Windows Server 2003, 2008, 2012
- Disk space: 70 MB for full installation

OPC UA Wrapper
Microsoft .NET Framework 3.5,

OPC XML DA and OPC WebView
Microsoft .NET Framework 4.0 or 4.5

In addition, if you want to use XML-DA access or SAEAUT OPC WebView, then the Internet Information Services (IIS) server is needed too. For more information please see the How to access OPC data from Internet/Intranet through Web Service.

Related articles
How to use access through Web Service, OPC XML-DA Wrapper

3.2 Database access

The OpcDbGateway enables to access databases through the following drivers:
- Microsoft OLE DB Provider for SQL Server,
- Microsoft OLE DB Provider for Microsoft Jet,
- Microsoft OLE DB Provider for ODBC,

- To access databases, OpcDbGateway uses various database drivers as for example ODBC or other drivers that are available in the operating system. Actually by OpcDbGateway used data access paths are marked on the figure below with red connection lines. There is possibility to add another access pathes in the figure (or other not shown there) according to the customers requirements. There is e.g.possibility to connect to MySQL database using OLE DB Provider for ODBC,

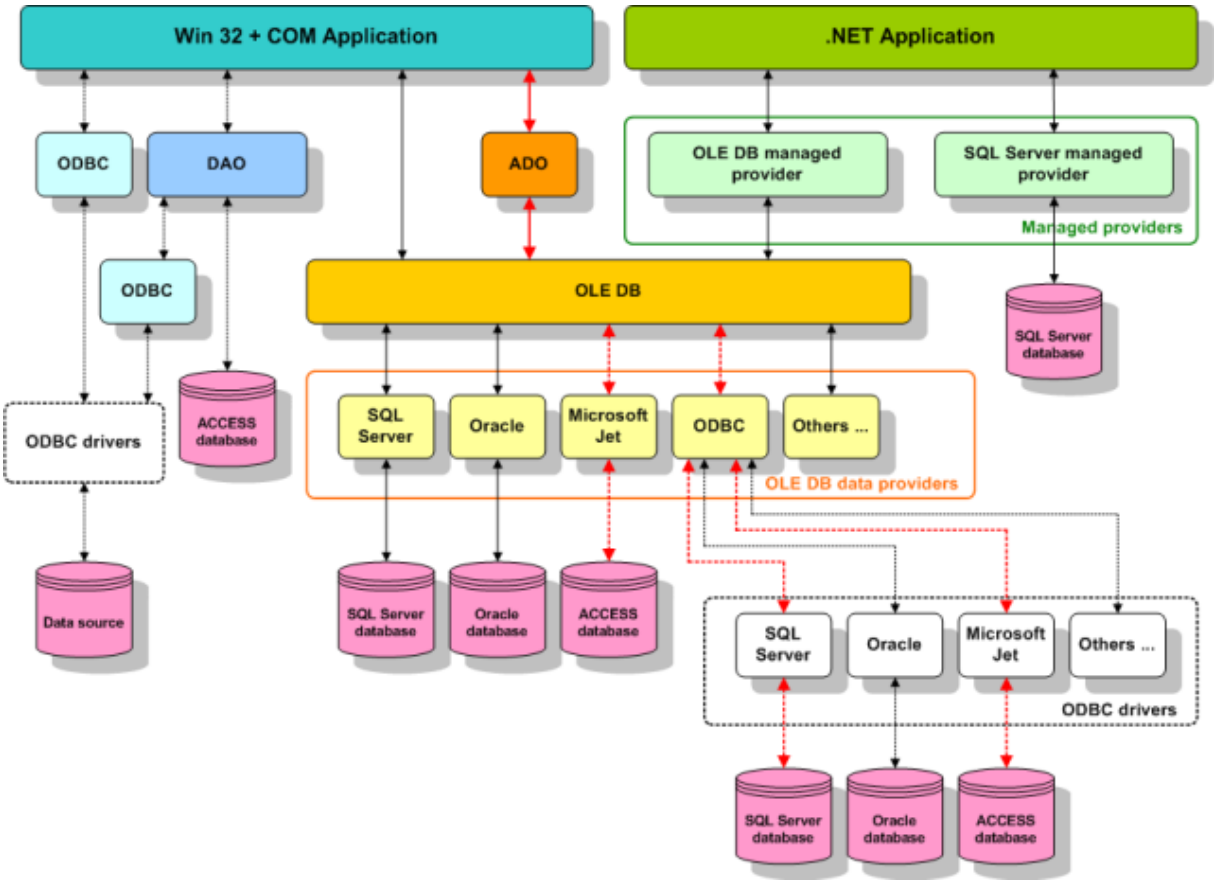


Figure: *Database connestions*

Access to different databases can be configured using connection strings as shown in the figure bellow.

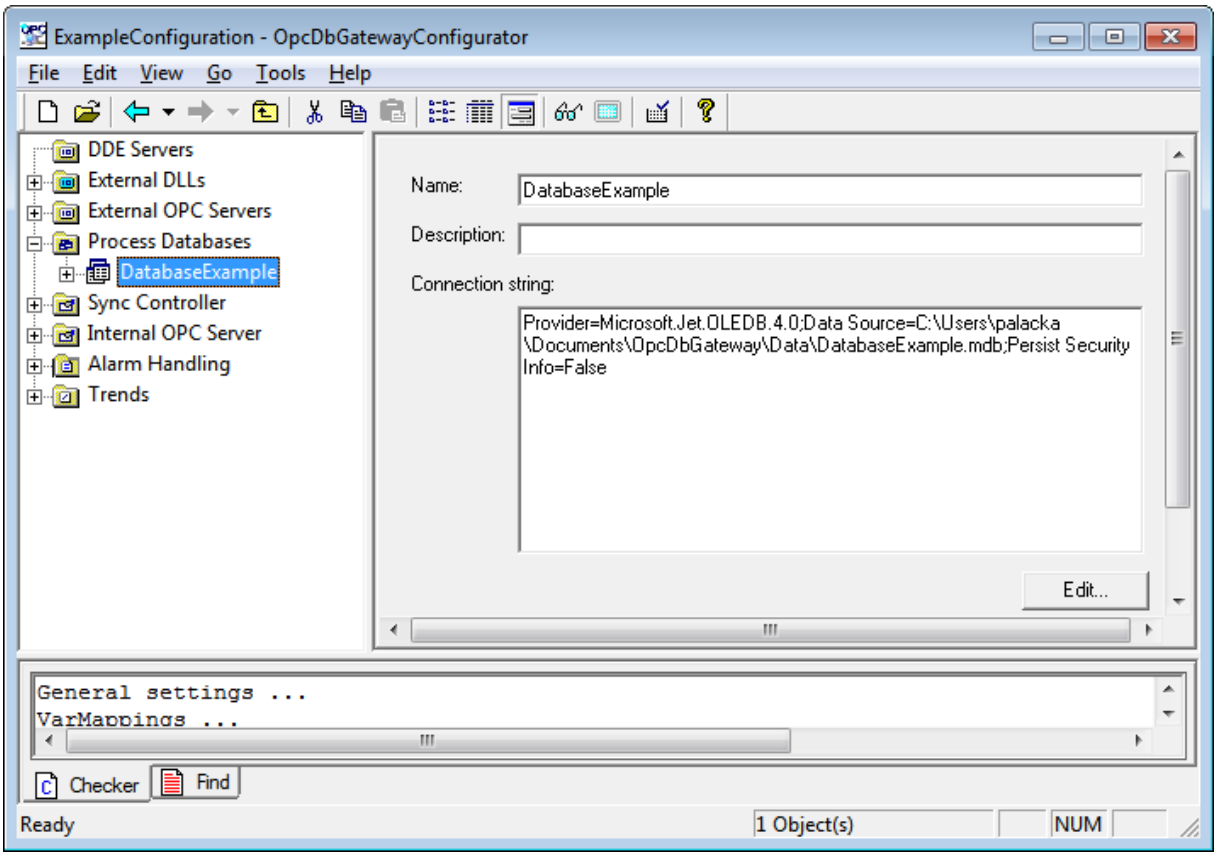


Figure: *Configuring o database access using connection string.*

3.3 OPC Interface

The **OpcDbGateway and SAEAUT Universal OPC Server** are provided with the following OPC Interface:

SERVER (Runtime)

- **OPC DA** (Data Access) **3.0**, **2.05**, **1.0**,
- **OPC AE** (Alarm and Events) **1.10** specifications implemented
- **OPC UA** (Unified Architecture) **1.01** - Installation package is enhanced with OPC UA 1.01 wrapper, enabling possibility to access data from OPC server according to the newest OPC Unified Architecture (UA) standard using web services or binary data over TCP/IP.
- **OPC XML-DA 1.01** - Installation package is enhanced with OPC XML-DA 1.01 wrapper enabling communication over web services.

CLIENT (only Runtime OpcDbGateway)

- **OPC DA** (Data Access) **3.0**, **2.05**,

CLIENT (Configurator - Monitor View)

- **OPC DA** (Data Access) **3.0**,

CLIENT (SAEAUT UA Data Access Client)

- **OPC UA** (Unified Architecture) **1.01**,

CLIENT (SAEAUT OPC DA and XML DA Client)

- **OPC DA** (Data Access) **3.0**, **2.0**
- **OPC XML-DA 1.01**

Related articles

OPC Unified Architecture (OPC UA)
OLE for Process Control (OPC)

3.3.1 OPC API for internal OPC server

When creating own OPC client applications able to communicate with internal OPC server following API functions can be used:

Data Access Server Required Interfaces	1.0	2.0	3.0	OpcDbGateway
OPCServer				
IUnknown	Required	Required	Required	Supported
IOPCServer	Required	Required	Required	Supported
IOPCCommon	N/A	Required	Required	Supported
IConnectionPointContainer	N/A	Required	Required	Supported
IOPCItemProperties	N/A	Required	N/A	Supported
IOPCBrowse	N/A	N/A	Required	Supported
IOPCServerPublicGroups	Optional	Optional	N/A	N/A
IOPCBrowseServerAddressSpace	Optional	Optional	N/A	Supported
IOPCItemIO	N/A	N/A	Required	Supported
OPCGroup				
IUnknown	Required	Required	Required	Supported
IOPCItemMgt	Required	Required	Required	Supported
IOPCGroupStateMgt	Required	Required	Required	Supported
IOPCGroupStateMgt2	N/A	N/A	Required	Supported
IOPCPublicGroupStateMgt	Optional	Optional	N/A	N/A
IOPCSyncIO	Required	Required	Required	Supported
IOPCSyncIO2	N/A	N/A	Required	Supported
IOPCAsyncIO2	N/A	Required	Required	Supported
IOPCAsyncIO3	N/A	N/A	Required	Supported
IOPCItemDeadbandMgt	N/A	N/A	Required	Supported
IOPCItemSamplingMgt	N/A	N/A	Optional	Supported
IConnectionPointContainer	N/A	Required	Required	Supported
IOPCAsyncIO	Required	Optional	N/A	Supported
IDataObject	Required	Optional	N/A	Supported

Following table summarizes the OPC Alarms and Events Server Interfaces interfaces supported by the OpcDbGateway:

OPC Alarms and Events Server Interfaces	Version 1.0	Version 1.10	OpcDbGateway
OPCEventServer			
IOPCCommon	Required	Required	Supported
IOPCEventServer	Required	Required	Supported
IOPCEventServer	Required	Required	Supported
IOPCEventServer2	N/A	Optional	Supported
IConnectionPointContainer	Required	Required	Supported
OPCEventAreaBrowser	Optional	Optional	Supported
IOPCEventAreaBrowser	Optional	Optional	Supported
OPCEventSubscription			
IOPCEventSubscriptionMgt	Required	Required	Supported
IOPCEventSubscriptionMgt2	N/A	Optional	Supported
IConnectionPointContainer	Required	Required	Supported

Details please see in OPC standards descriptions from OPC Foundation.

OpcDbGateway and SAEAUT Universal OPC Server

Part IV

4 OPC via Internet

How to transfer OPC data over Internet

The OpcDbGateway and SAEAUT Universal OPC Server are provided with two standardized OPC interfaces which enable to transfer data in both Internet and Intranet networks. More details about these standards you can read in the following topics:

- OPC Unified Architecture (OPC UA) (the latest OPC standard),
- OPC XML-DA standard.

Related articles

OLE for Process Control (OPC),
Benefits of OPC UA for End Users

4.1 OPC Unified Architecture (OPC UA)

What is OPC UA?

The Unified Architecture (UA) is the next generation OPC standard that provides a cohesive, secure and reliable cross platform framework for access to real time and historical data and events. The goal for this project is to provide a path forward from the original OPC communications model (namely COM/DCOM) to a cross-platform service-oriented architecture (SOA) for process control, while enhancing security and providing an information model.

OPC UA supports two protocols. This is visible to application programmers only via changes to the URL. The binary protocol is **opc.tcp://Server** and **http://Server** is for Web Service. Otherwise OPC UA works completely transparent to the API.

The binary protocol offers the best performance/least overhead, takes minimum resources (no XML Parser, SOAP and HTTP required which is important for embedded devices, offers best interoperability (binary is explicitly specified and allows fewer degrees of freedom during implementation) and uses only TCP port 4840 communication easing tunneling or easy enablement through a firewall.

The Web Service (SOAP) protocol is best supported from tools as e.g., from JAVA or .Net environments, and is firewall-friendly, using standard http/https ports.

Benefits of OPC UA for End Users

OPC-UA provides a way to connect clients and servers in a secure manner, without relying on Microsoft DCOM. This is a big advantage because it means that you are no longer saddled with the headaches associated with having to configure DCOM. It can also allow users to make secure connections through firewalls and over VPN connections.

www.opcfoundation.org - Benefits of OPC UA for End Users

How does OPC UA Work in OpcDbGateway and SAEAUT

Universal OPC Server

Interface OPC UA in OpcDbGateway and SAEAUT Universal OPC Server is provided by OPC UA Wrapper. This OPC UA Wrapper is not installed immediately together with OpcDbGateway and SAEAUT Universal OPC Server. But, you can additionally install it from start menu.

More details about OPC UA Wrapper you can find in the following topics:

- OPC UA Wrapper - Installation,
- OPC UA Wrapper - Installation on diffrent computer,
- OPC UA Wrapper - Start,
- OPC UA Wrapper - Stop.

After installation of OPC UA Wrapper, you can start this wrapper according to the OPC UA Wrapper - Start topic. If OPC UA Wrapper has been started, the OpcDbGateway and SAEAUT Universal OPC Server data are available via OPC UA interface and you can monior and present them in installed SAEAUT UA Data Access Client (Figure 1). You can start this client from start⇨All Programs⇨SAEAUT UA Data Access Client ⇨SAEAUT Data Access Client. On the Figure 1 below are data from OpcDbGateway (or SAEAUT Universal OPC Server) presented via OPC UA interface in SAEAUT UA Data Access Client.

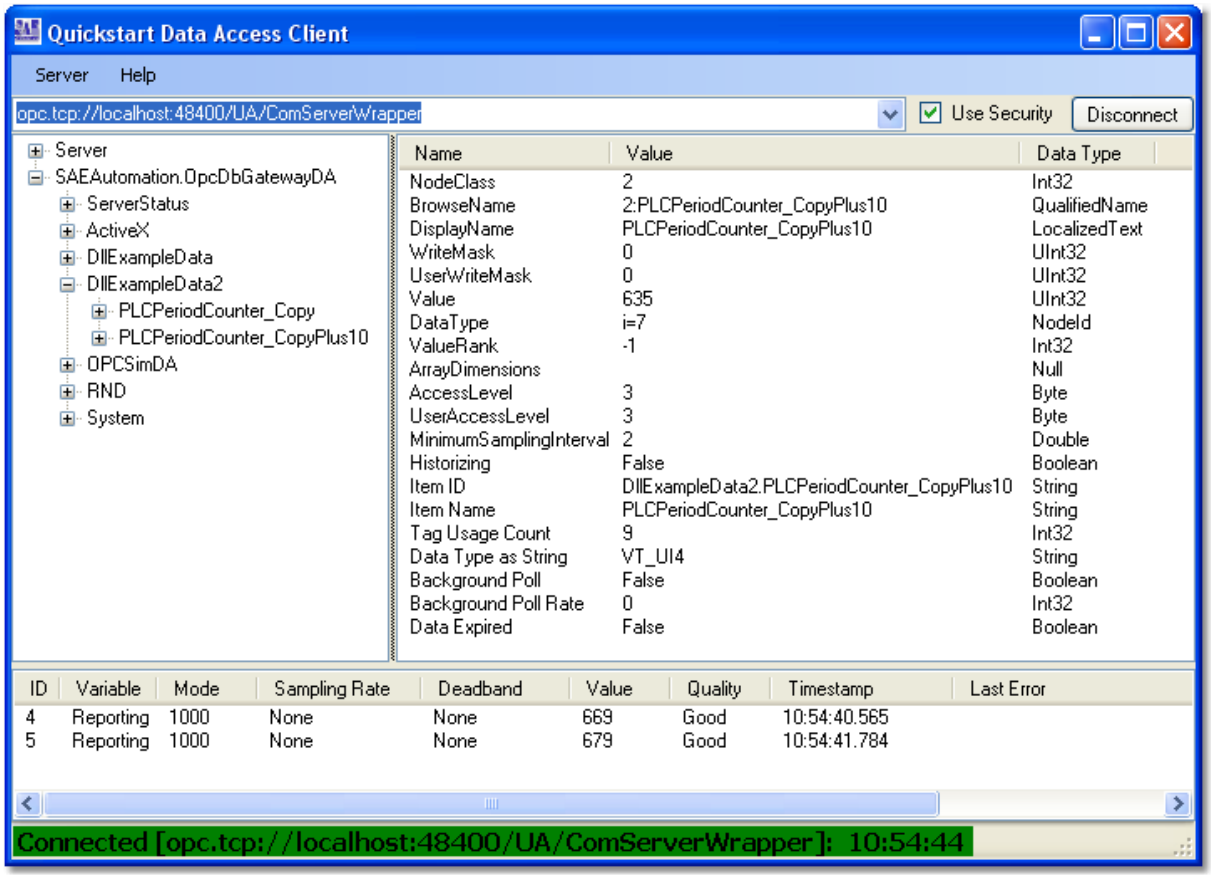


Figure: *SAEAUT UA Data Access Client presenst data from OpcDbGareway (or SAEAUT Universal OPC Server).*

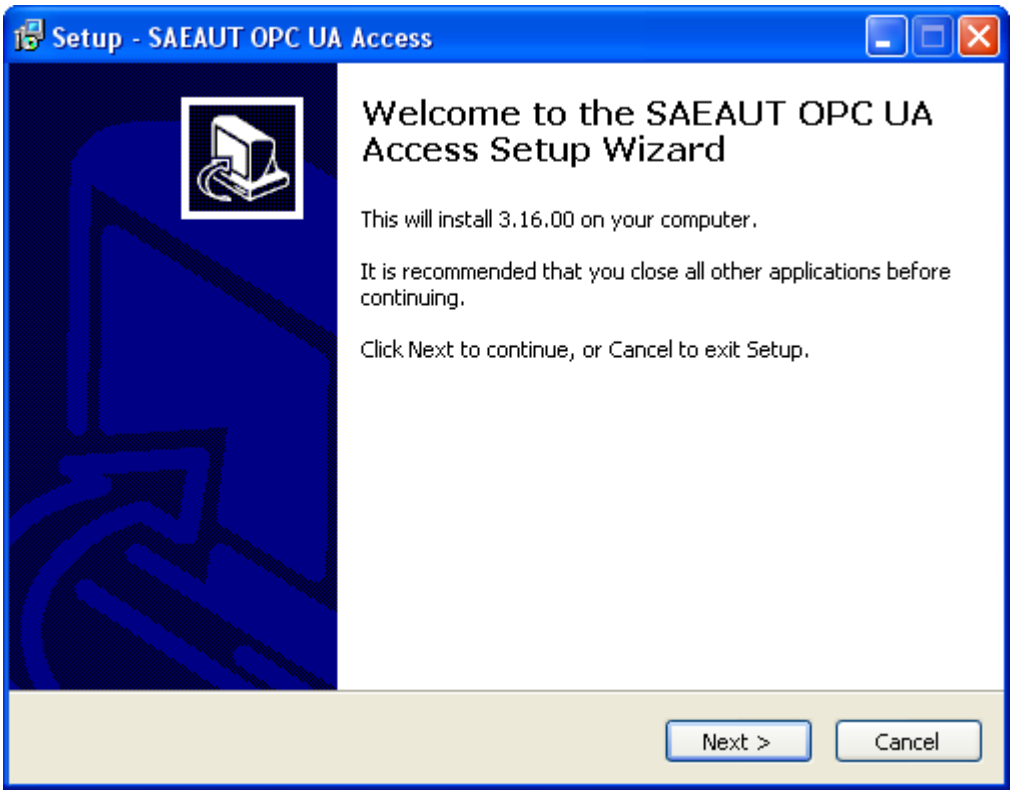
Related articles

OLE for Process Control (OPC),
Benefits of OPC UA for End Users

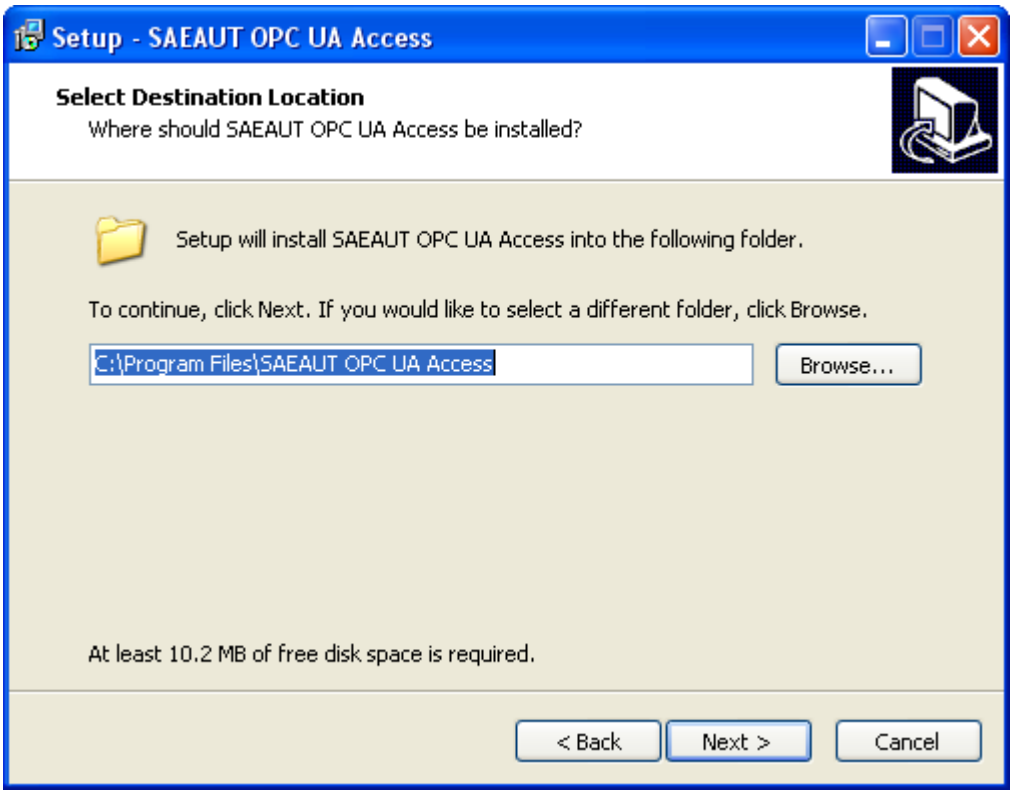
4.1.1 OPC UA Wrapper - Installation

INSTALLATION

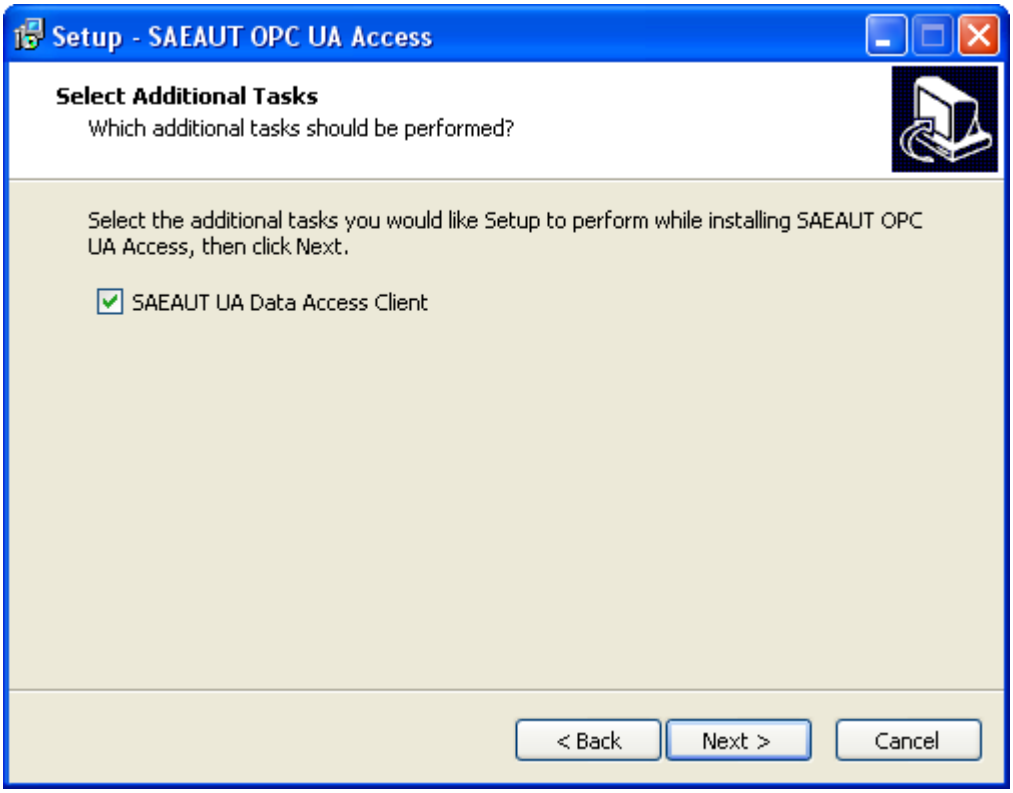
1. Go to directory ..\Program Files\OpcDbGateway\OPC UA Wrapper\ or ..\Program Files\SAEAUT Universal OPC Server\OPC UA Wrapper\.
2. Click on Setup.exe
3. Click the Next button.



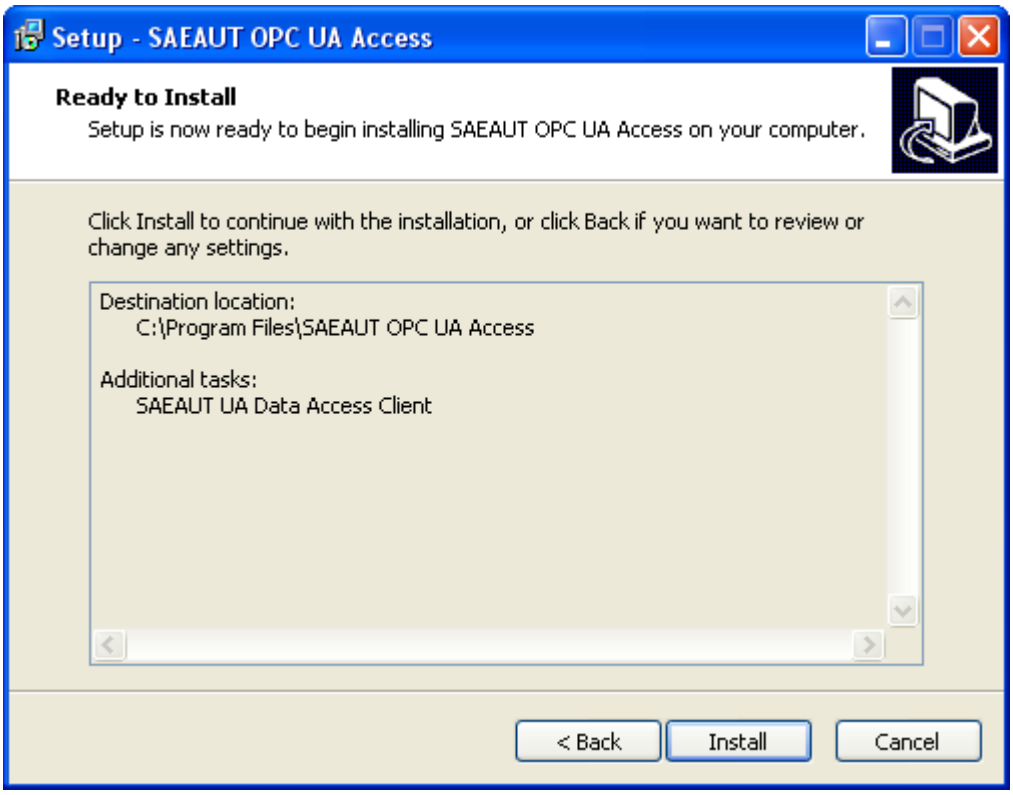
4. Select destination folder (we recommend it). Click the Next button.



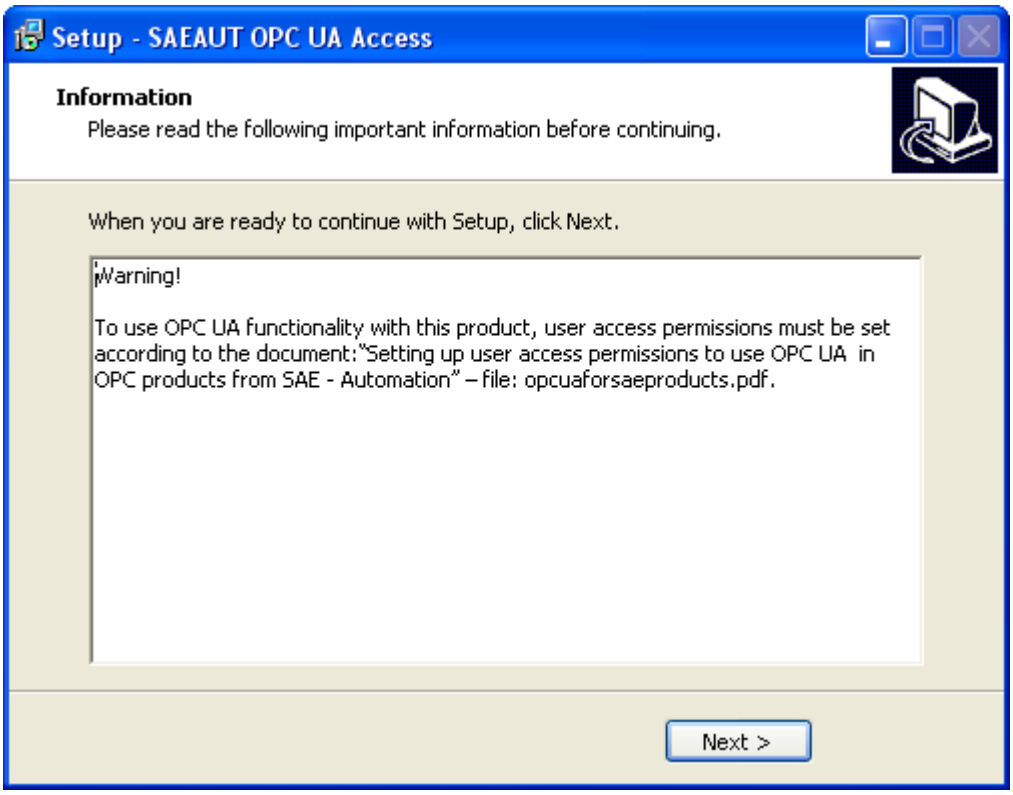
5. You can choose installation of SAEAUT UA Data Access Client (we recommend it). Click the Next button.



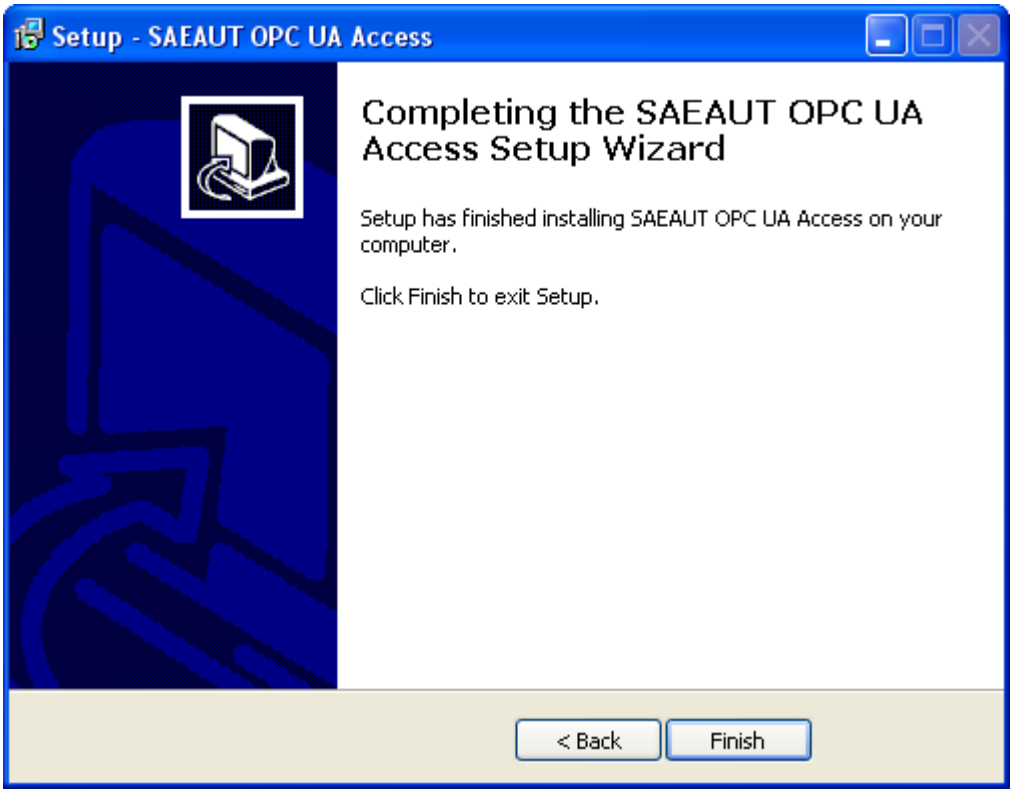
6. Click the Install button.



7. Click the Next button.



8. Click the Finish button.



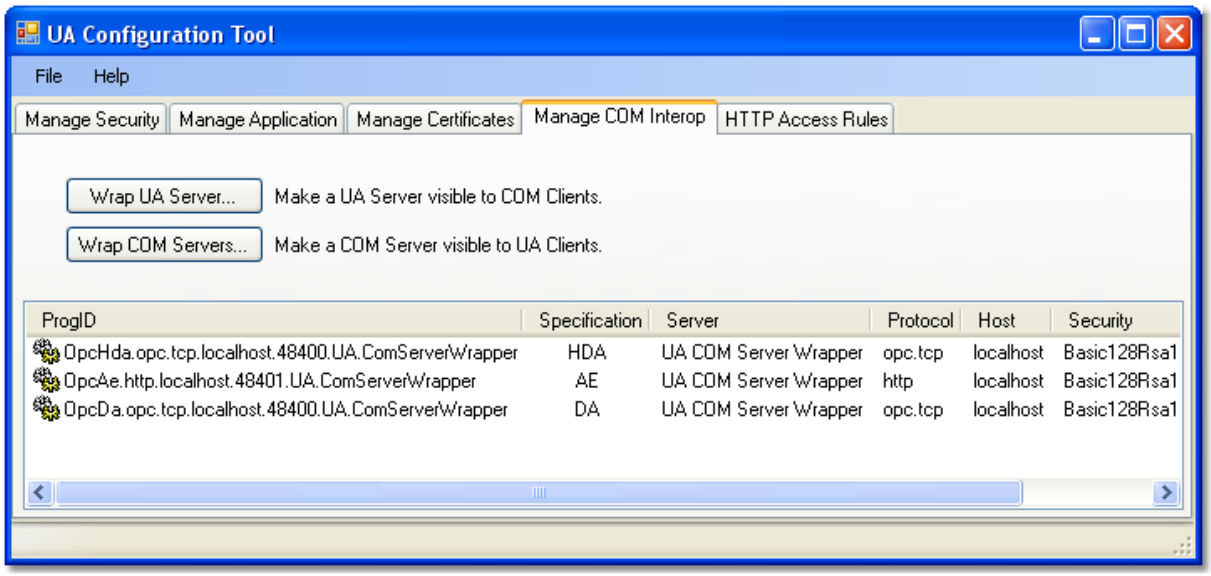
Related articles
OPC UA Wrapper - Installation on diffrent computer
OLE for Process Control (OPC),
Benefits of OPC UA for End Users

4.1.2 OPC UA Wrapper - Installation on another computer

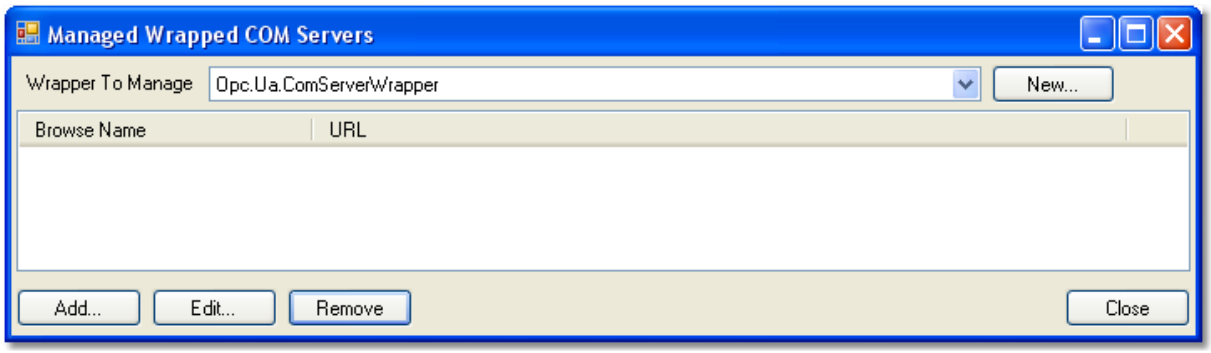
The OPC UA Wrapper can be installed on diffrent compuer as OpcDbGateway and SAEAUT Universal OPC Server. In order to run and access the OpcDbGateway (or SAEAUT Universal OPC Server) from remote computer, you have to configure the wrapper through UA Configuration Tool.

INSTALLATION

1. Install wrapper according to OPC UA Wrapper - Installation topic.
2. Click on start⇨All Programs⇨OPC Foundation⇨UA SDK 1.01⇨UA Configuration Tool.

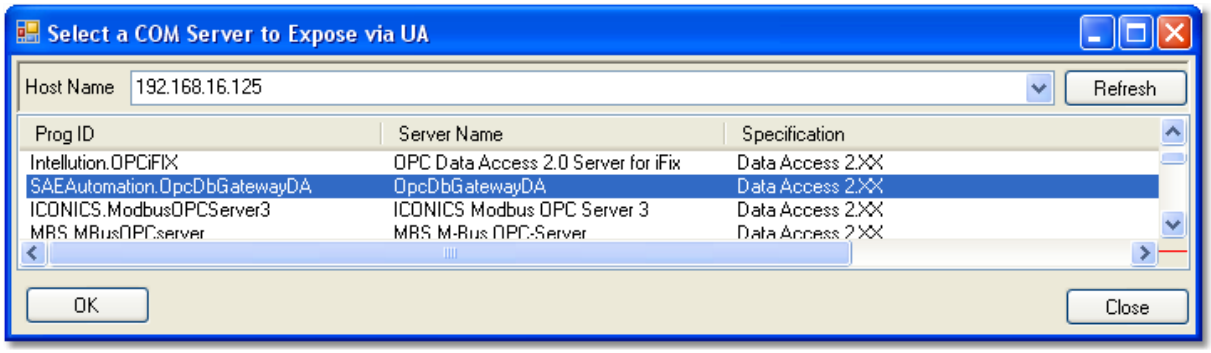


3. Select Manage COM Interop tab as in the Figure above.
4. Click the Wrap COM Servers ... button.

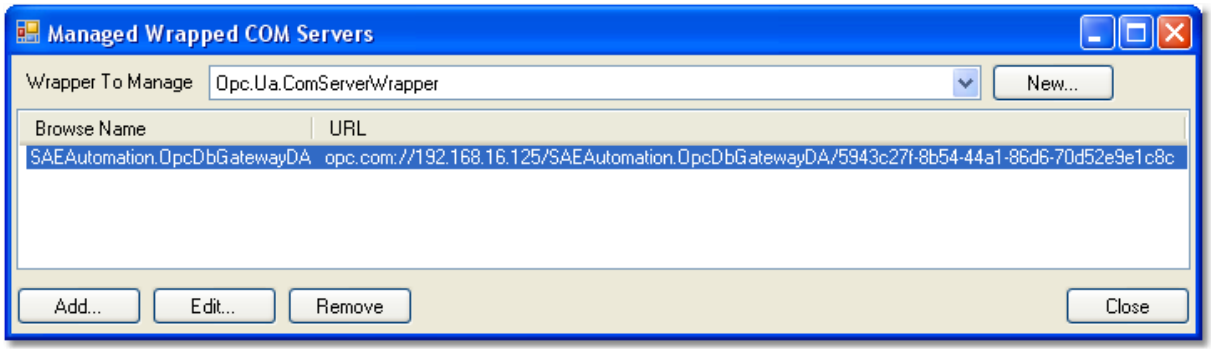


7. Click the Add... button.

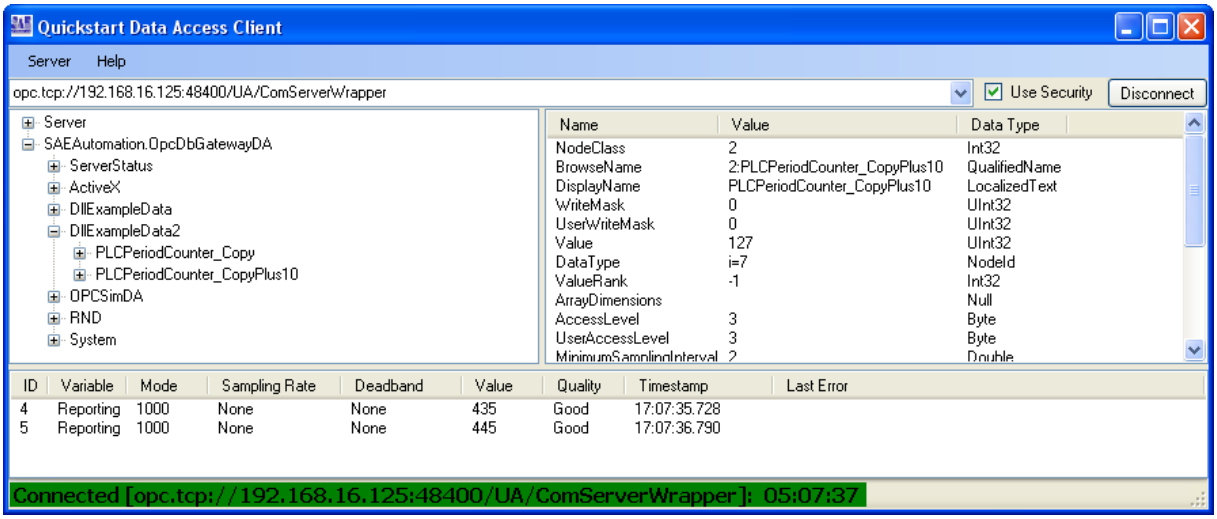
8. Enter the IP Address of remote computer (e.g. 192.168.16.125) and click the Refresh button.
9. Select SAEAutomation.OpcDbGatewayDA server.
10. Click the OK button.



11. Click the Close button.



12. The OPC UA Wrapper is already successfuly configured.
13. Test with SAEAUT UA Data Access Client. You can start this client from start⇨All Programs⇨SAEAUT UA Data Access Client ⇨SAEAUT Data Access Client.



Related articles

OLE for Process Control (OPC),
Benefits of OPC UA for End Users

4.1.3 OPC UA Wrapper - Start

START

1. Start wrapper from start⇨All Programs⇨OpcDbGateway⇨UA Wrapper⇨Start UA Wrapper (see Figure 1).

Note that: Launch the Start UA Wrapper in mode "Run as administrator" for the following systems (see Figure 2):
•Windows Vista,
•Windows 7,
•Windows 8
•Windows Server 2003,
•Windows Server 2008
•Windows Server 2012.

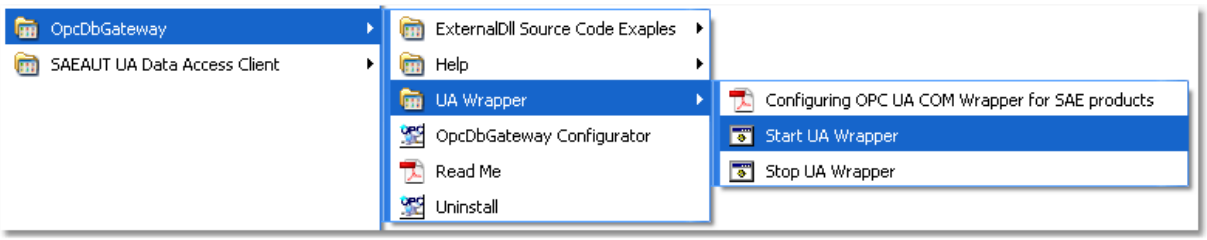


Figure: *UA Wrapper - Start.*

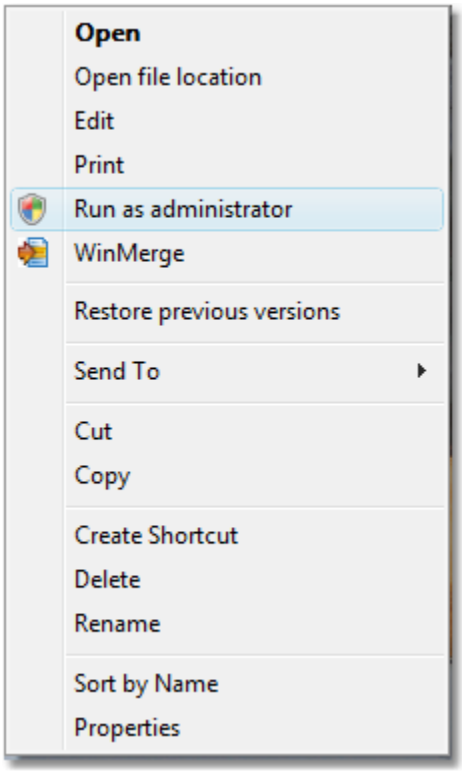


Figure: *UA Wrapper - Start: Run as administrator.*

Related articles
OLE for Process Control (OPC),
Benefits of OPC UA for End Users

4.1.4 OPC UA Wrapper - Stop

STOP

1. Start wrapper from start⇨All Programs⇨OpcDbGateway⇨UA Wrapper⇨Stop UA Wrapper (see Figure 1).

Note that: Launch the Start UA Wrapper in mode "Run as administrator" for the following systems (see Figure 2):
•Windows Vista,
•Windows 7,
•Windows 8
•Windows Server 2003,
•Windows Server 2008
•Windows Server 2012.

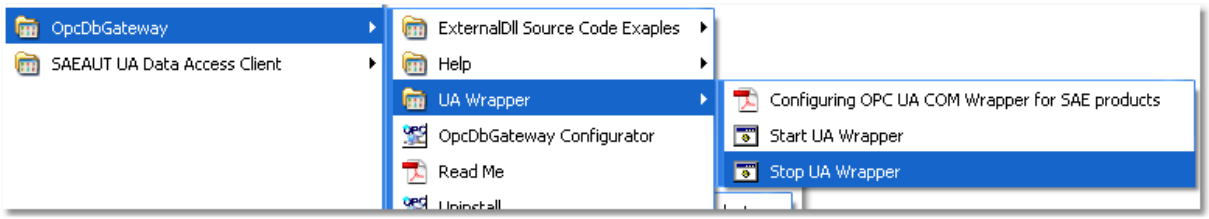


Figure 1: *UA Wrapper - Stop.*

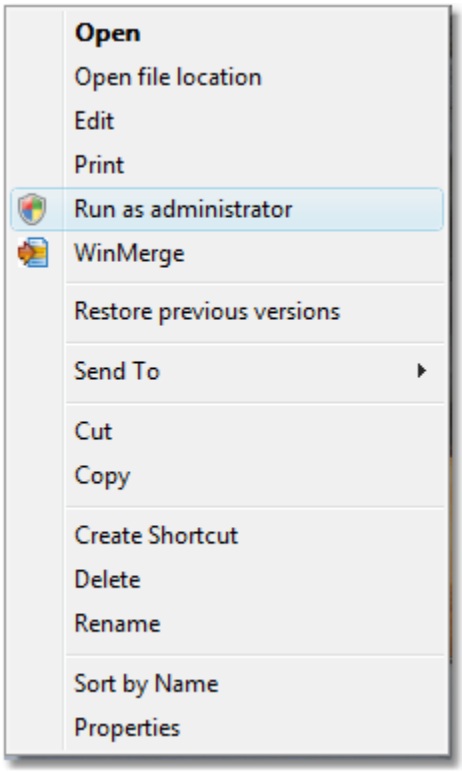


Figure 2: *UA Wrapper - Stop: Run as administrator.*

Related articles

OLE for Process Control (OPC),
Benefits of OPC UA for End Users

4.2 OPC XML-DA

Builds on the OPC Data Access specifications to communicate data in XML. Incorporates SOAP and Web services.

SAE - Automation, s.r.o., (Ltd.) delivers **XML-DA Wrapper** to make OpcDbGateway and SAEAUT Universal OPC Server available through Web Services. Installing and use of XML-DA Wrapper needs the Internet Information Services (IIS) server to be installed on the PC. IIS is a Windows component available on Windows instalation CD. Installing the XML-DA Wrapper generates an **XML-DA Web Service** and creates a virtual directory in the IIS directory. IIS server is accesible from **Control Panel → Administrative Tools**.

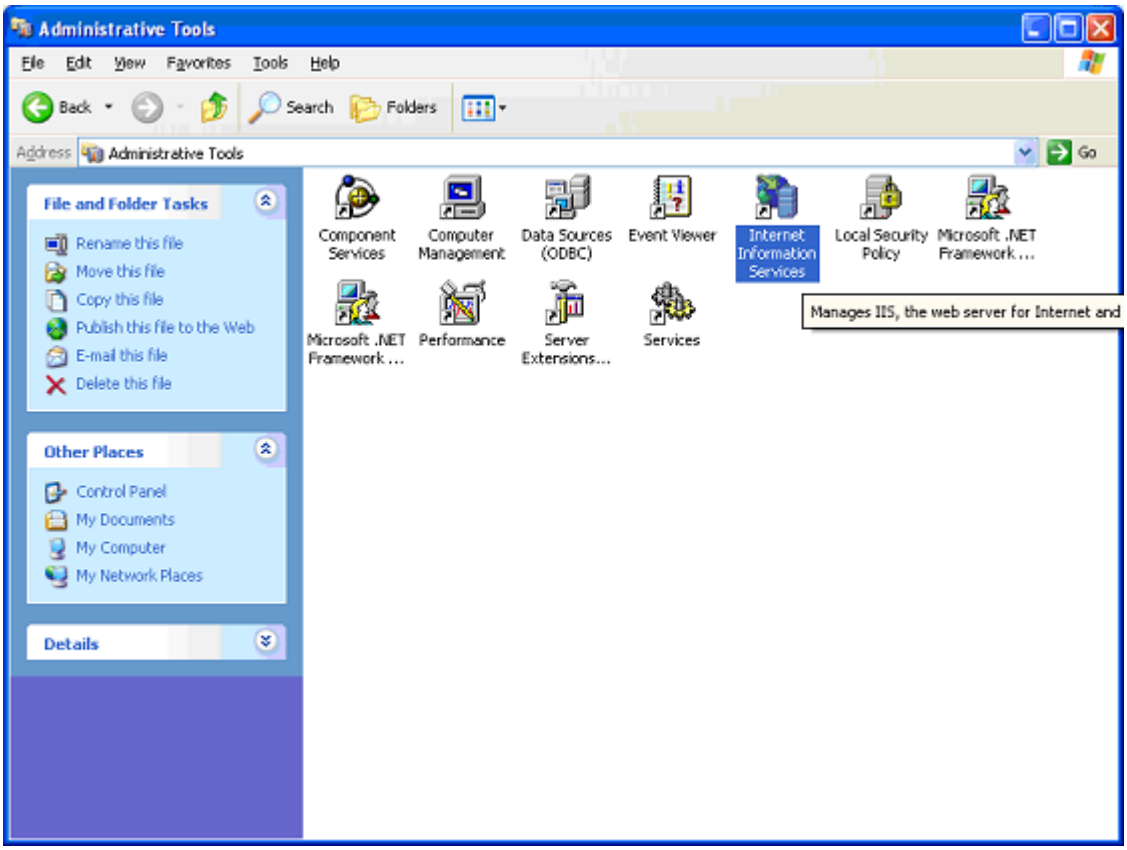


Figure: *Administrative Tools*

To enable OpcDbGateway and SAEAUT Universal OPC Server XML-DA Web Service, make a new copy of already existing Web service called **OPC_XML-DA_WrapperService.asmx** and name it as **SAEAutomation.OpcDbGatewayDA.3.asmx** (The same file is already prearranged in OpcDbGateway and SAEAUT Universal OPC Server aplication directory under Web Service). The copy of **SAEAutomation.OpcDbGatewayDA.3.asmx** file has to be located in the file directory associated with your virtual directory.).

From the ver. 5 of OpcDbGateway and SAEAUT Universal OPC Server above mentoned files are copied to the virtual directory automatically by installation.

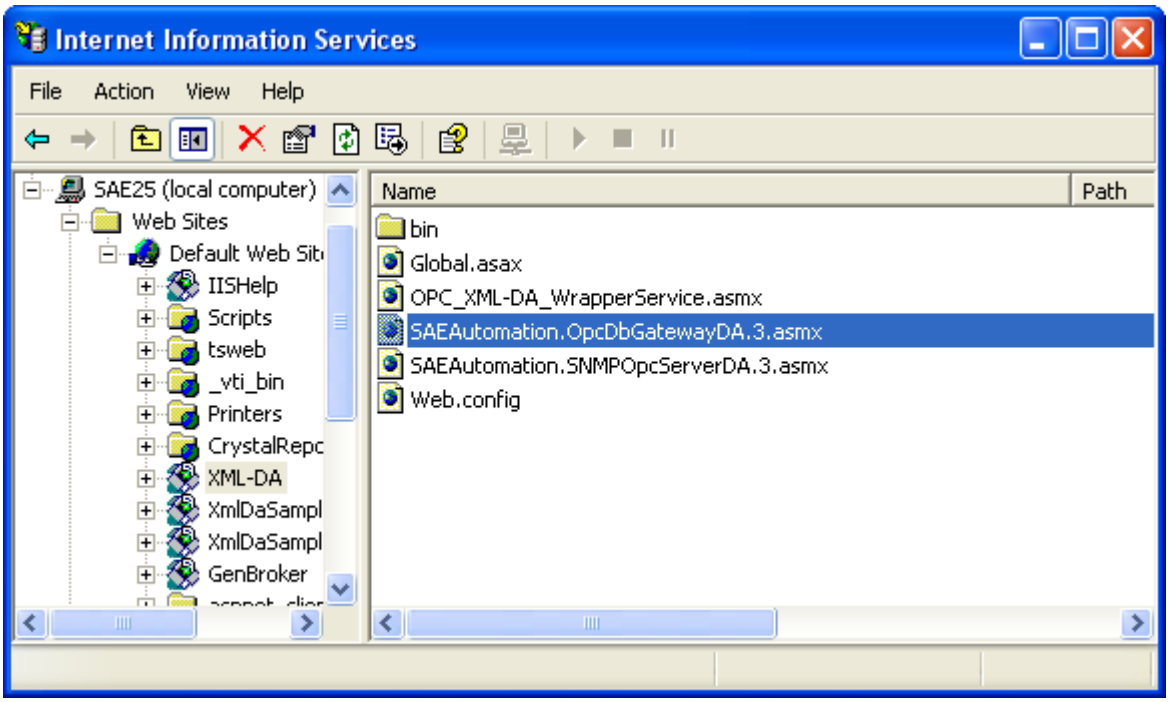


Figure 105: *Internet Information Services*

OPC_XML-DA_WrapperService.asmx is located in the directory specified by the instalation of XML-DA Wrapper. Virtual directory **XML-DA** created by installation of the Wrapper must point to the directory, where Web Service (associated with OPC COM server) is located. Check it by right clicking the virtual directory XML-DA and choosing Properties item.

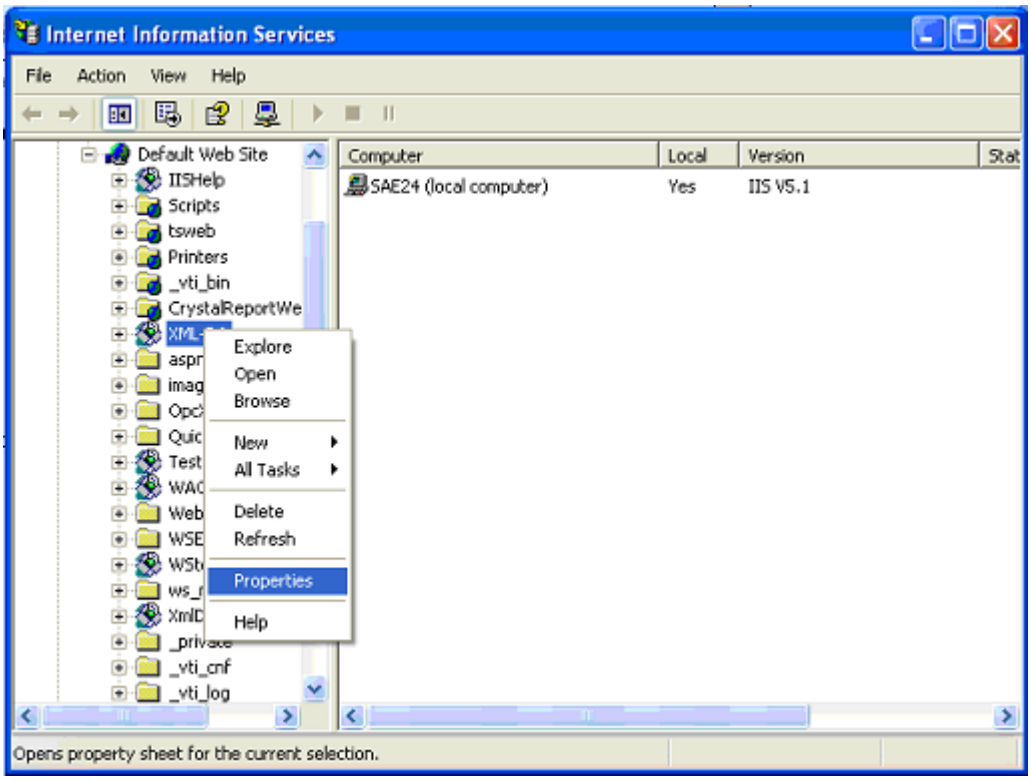


Figure: *Virtual directory XML-DA*

In Properties window check the value of Local Path setting.

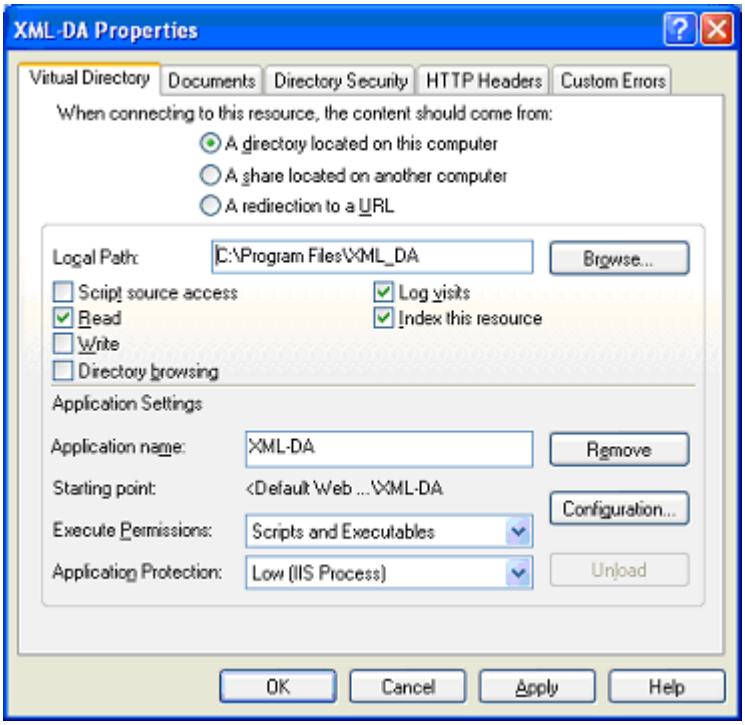


Figure 107: *Virtual directory XML-DA properties*

If the installation of the Wrapper failed and the virtual directory has been not created, it is possible to create it manually by right-click on Default Web Sites and choosing the option New -> Virtual Directory.

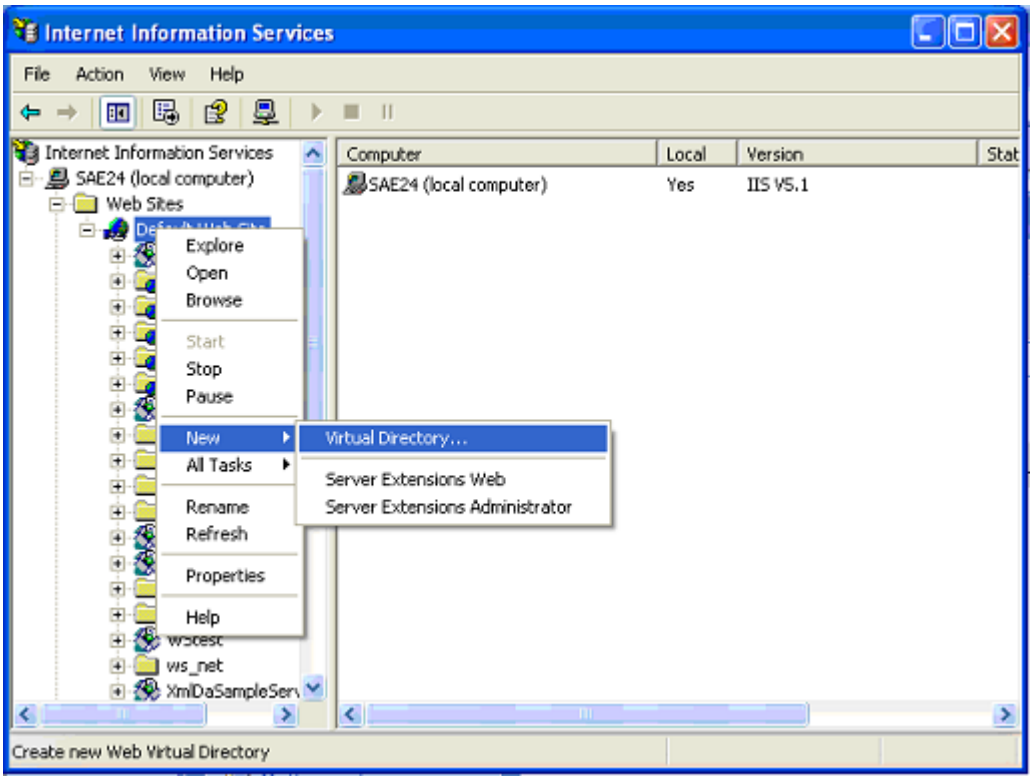


Figure 108: *creating a new virtual directory*

Virtaul Directory Creation Wizard starts and asks to type the alias of a virtual directory.

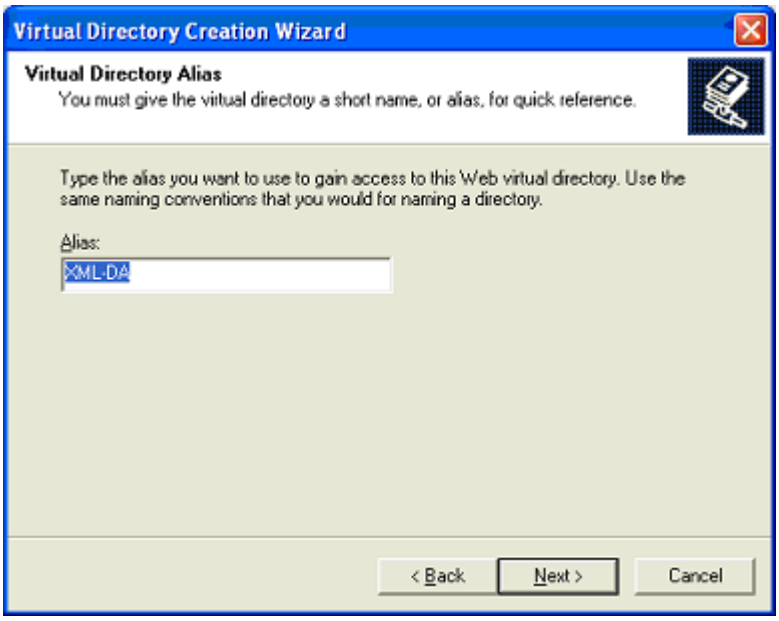


Figure 109a: *creating virtual directory XML-DA*

This alias will be used for browsing the supported functions of a web service and for connecting to the server. Then Wizard asks to set the path to the directory where desired web service is located. Access permissions can be left unchanged.

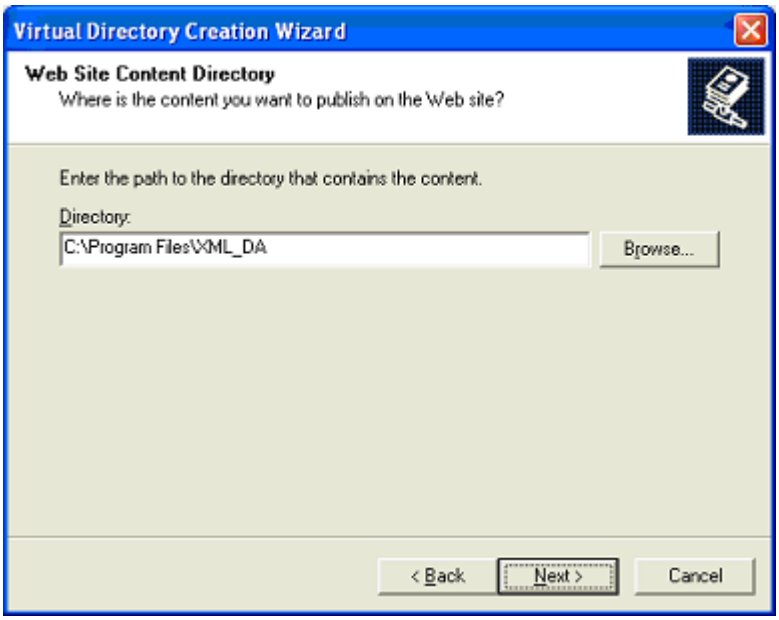


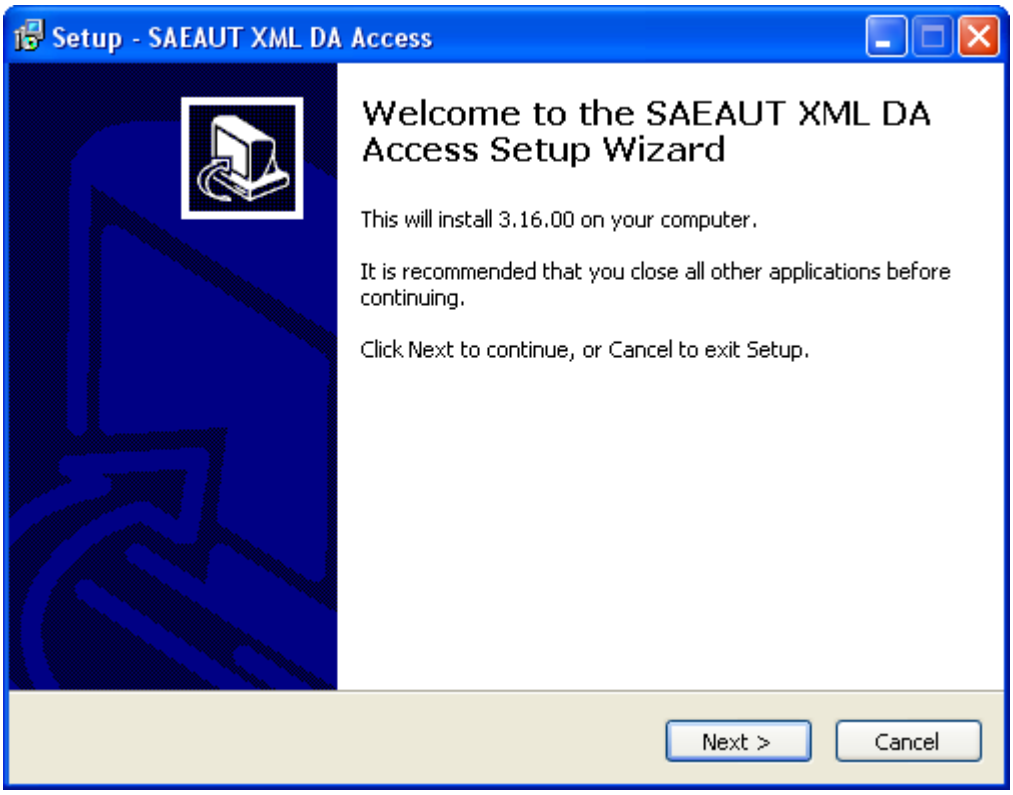
Figure: *Virtual directory XML-DA*

Related articles

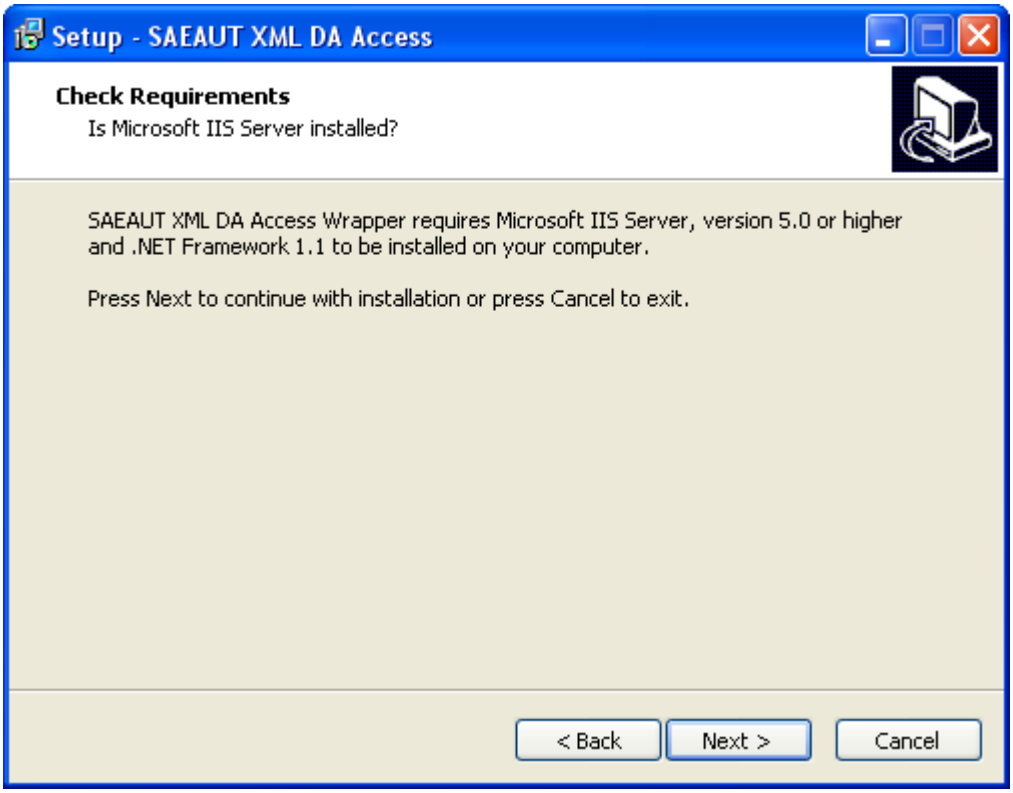
How to set the access rights for OPC XML-DA Wrapper
OpcDbGateway Web Service available from Internet Explorer
A simple OPC XML-DA Client application

4.2.1 OPC XML-DA Wrapper - Installation

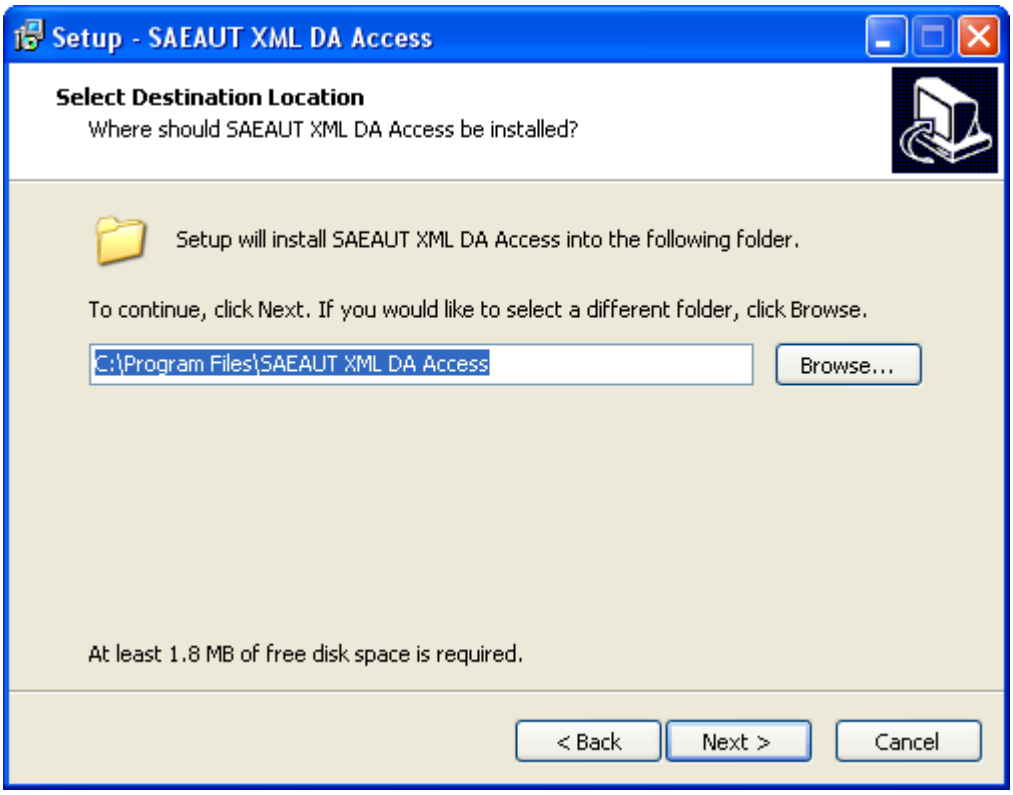
1. Go to directory XML DA Wrapper in start menu and click on XML DA Install
2. Click the Next button.



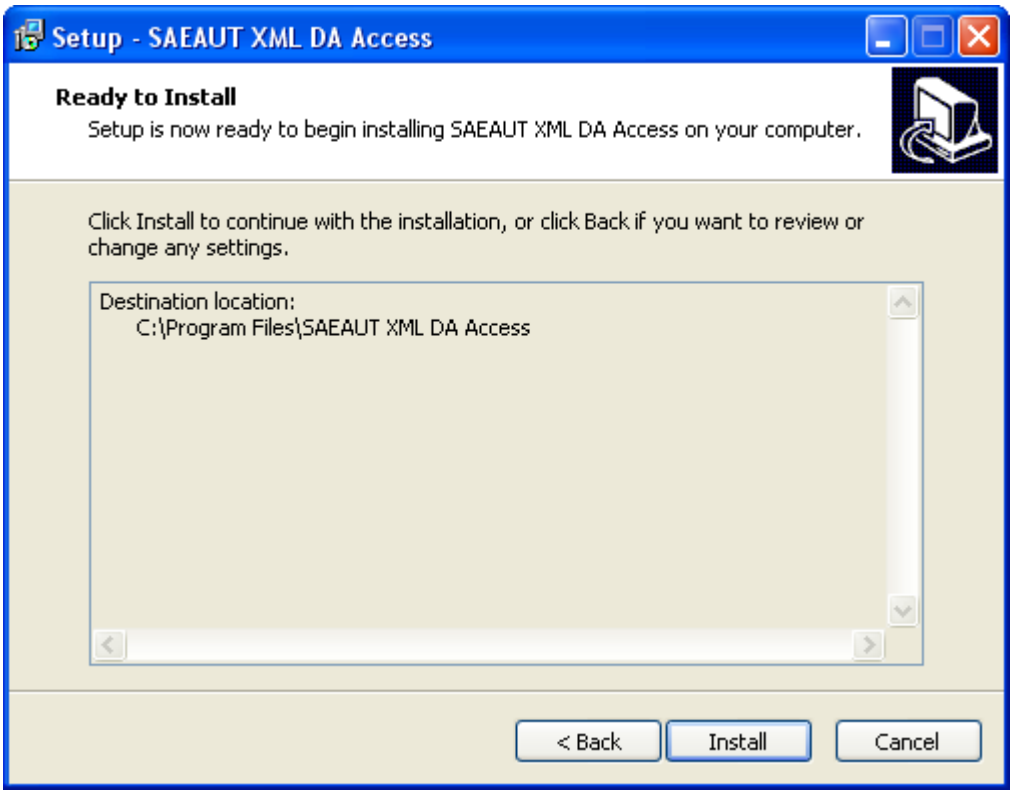
3. Click the Next button.



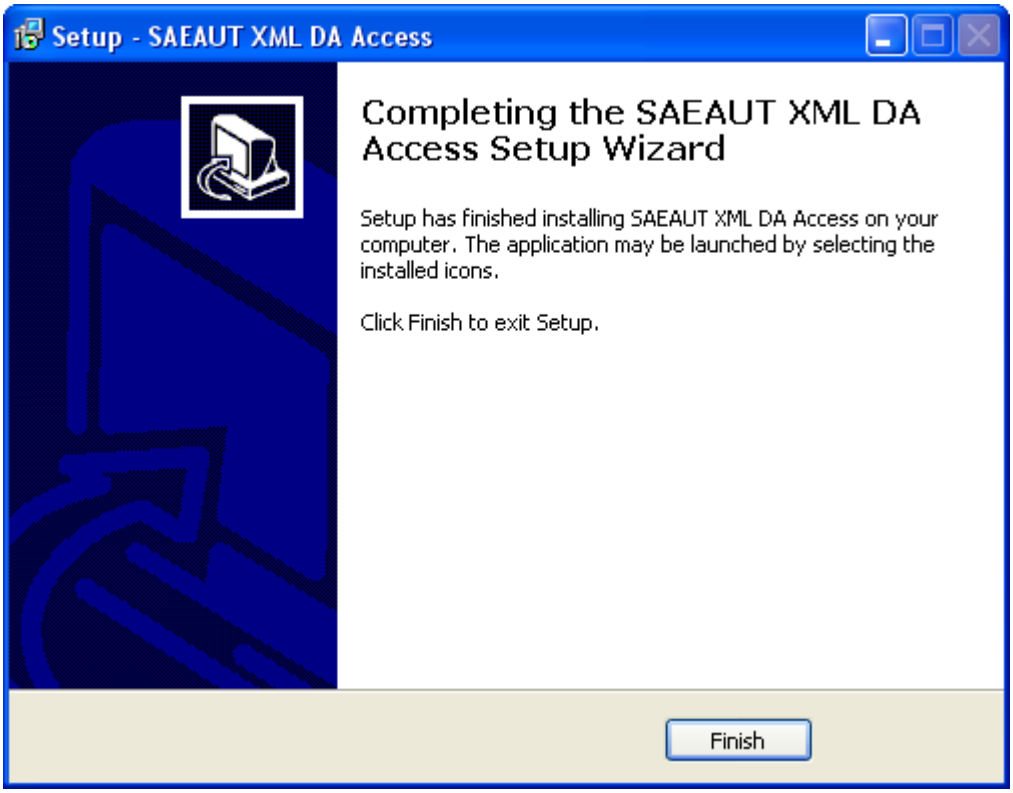
4. Select destination folder (we recommend it). Click the Next button.



5. Click the Install button.



6. Click the Finish button.



Related articles

OPC UA Wrapper - Installation on diffrent computer
OLE for Process Control (OPC),

4.2.2 How to set the access rights for OPC XML-DA Wrapper

If XML-DA client is having problems with accessing OPC COM server, the COM security of a computer might be required to be modified. To do it, go to the **Control Panel** → **Administrative Tools** → **Component Services** → **Computers** → **My Computer** and choose Properties option, click the tab COM Security and click Edit Default button under Access Permissions group.

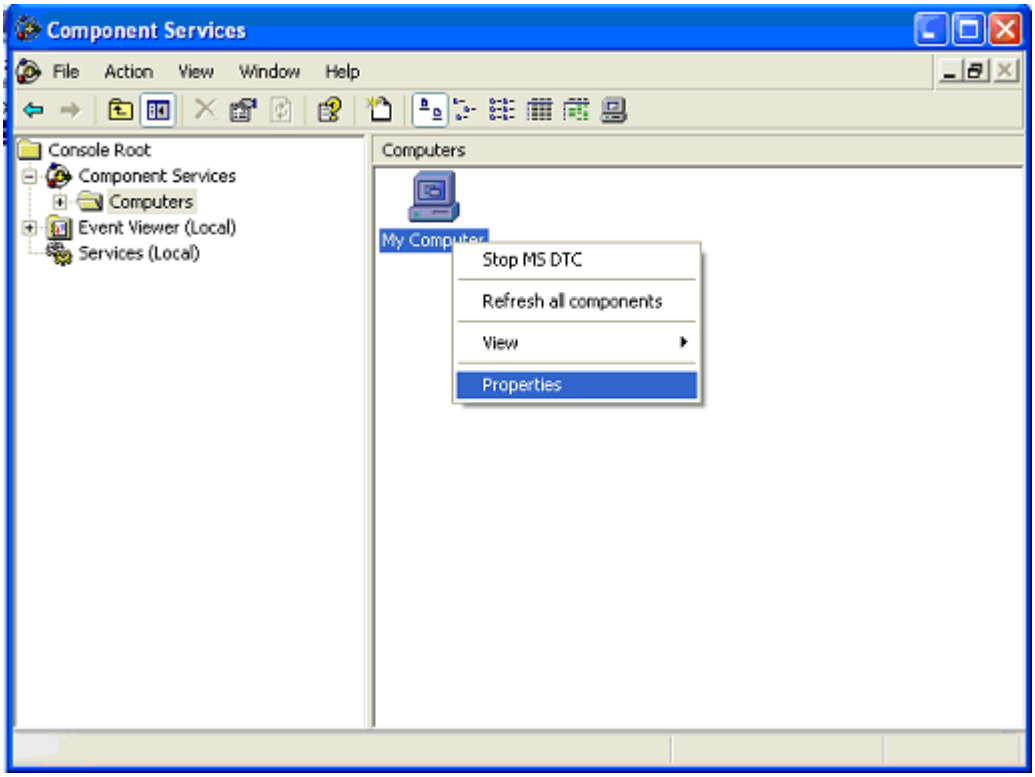


Figure: *Component Services*

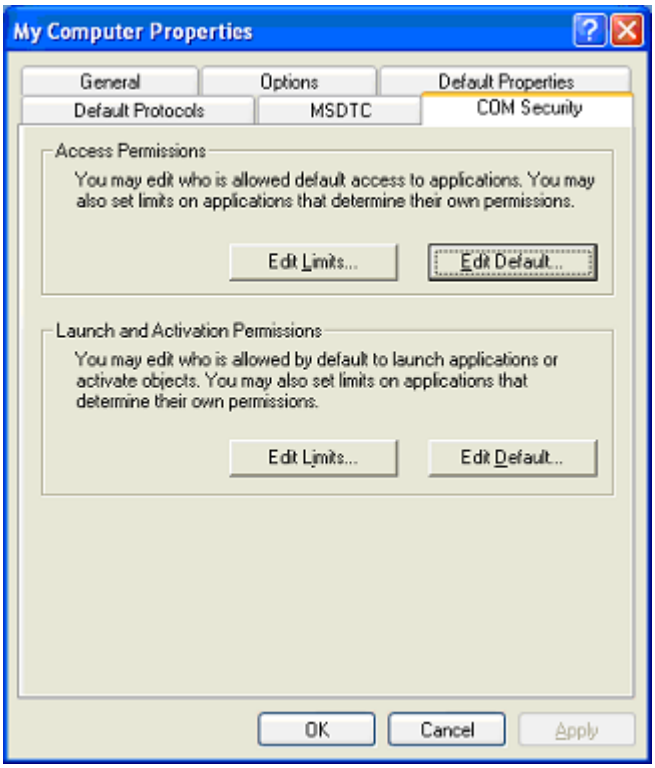


Figure 111: *My Computer Properties*

You need to set the local and remote access for the ASP.NET user.

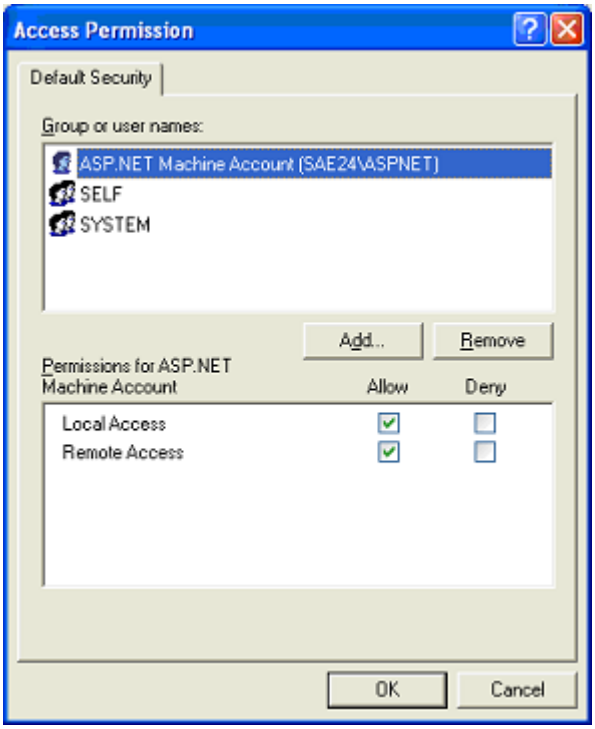


Figure 112: *Access Permission*

If ASP.NET user is not in the user list, click the Add button and make sure that the location where you

search for users is the name of your computer (**Locations** → **pc_name**).

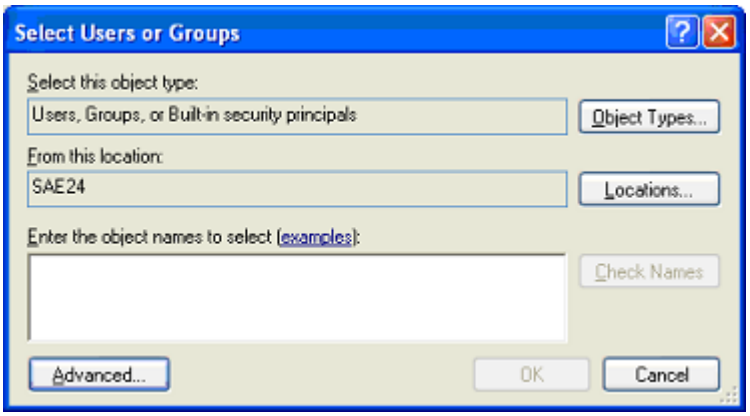


Figure 113: *My Computer Properties*

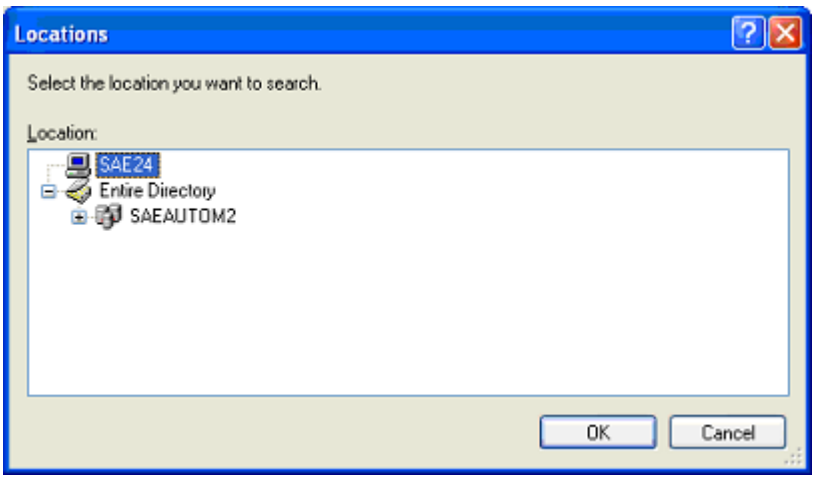


Figure: *Locations*

Then click **Advanced** → **Find Now**, choose ASPNET user and add it to the user list. Repeat the same action for Launch and Activation Permissions group. Then restart your PC.

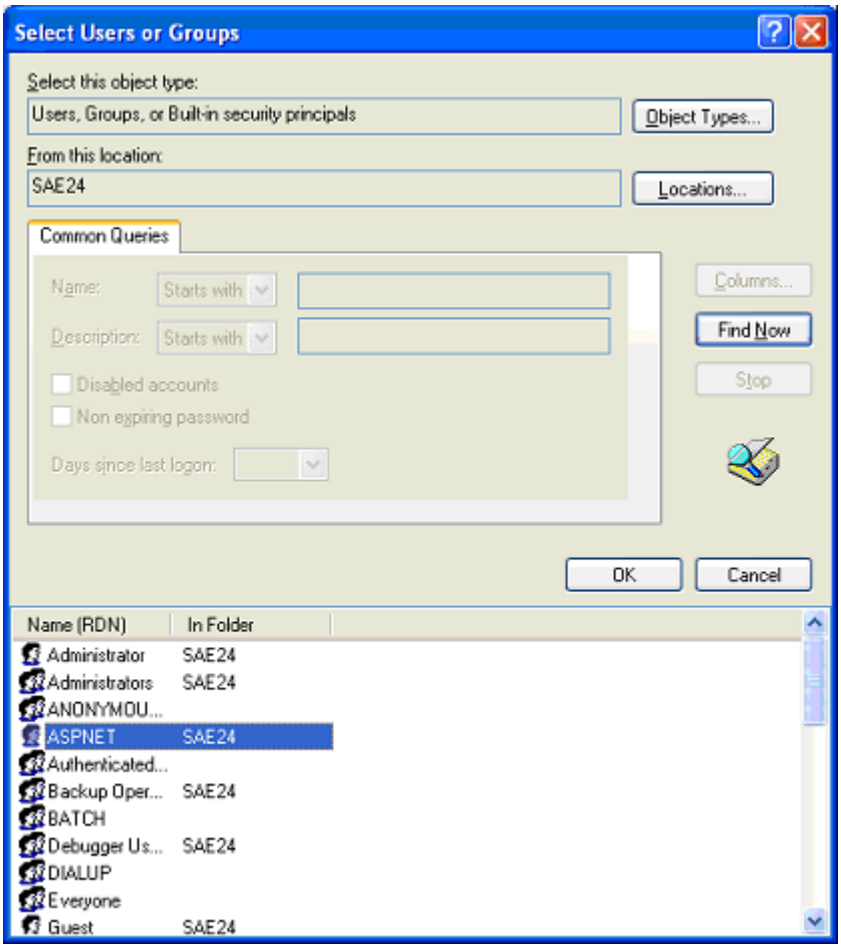


Figure: *Select Users or Groups*

Related articles

How to access OPC data from Internet/Intranet through Web Service
OpcDbGateway Web Service available from Internet Explorer
A simple OPC XML-DA Client application

4.2.3 OpcDbGateway Web Service available from Internet browser

To explore all available methods supported by OpcDbGateway Web Service type the URL in the browser as follows:

Description: http://node/<virtual directory>/<ProgID>.asmx

http://localhost/XML-DA/SAEAutomation.OpcDbGatewayDA.3.asmx

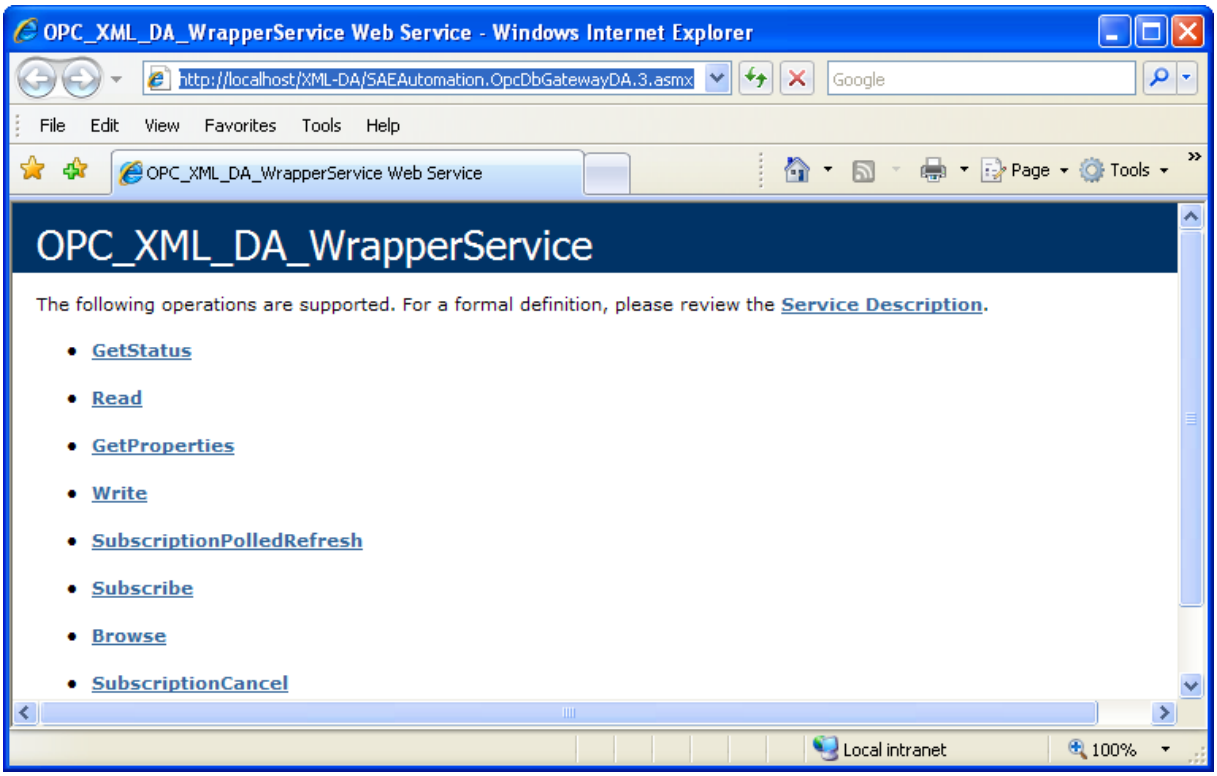


Figure: *http://localhost/XML-DA/SAEAutomation.OpcDbGatewayDA.3.ask:mx*

Remark: From the ver.5 this web service can be called also from Start menu

Related articles

How to access OPC data from Internet/Intranet through Web Service
How to set the access rights for OPC XML-DA Wrapper

OpcDbGateway and SAEAUT Universal OPC Server

Part V

5 External DLL - usage

External Dll's are program modules implemented by user (or bought from third parties) as dynamically linked libraries (DLL). Properly implemented External Dll becomes a **part of the configurable functionality** of OpcDbGateway and SAEAUT Universal OPC Server. In both products**,** it is possible to use a DLL module for implementing:

- complicated algorithms,
- database access,
- communication drivers,
- communication with I/O modules,
- other drivers,
- etc.

The OpcDbGateway and SAEAUT Universal OPC Server load external DLL by both executable modules Configurator and Runtime.

- The Configuration module provides list of all available external DLLs (DLLs in folder ExternalDLL), validates DLL interface and enables to map DLL to configuration.
- The Runtime module loads only DLLs which are defined in the active configuration. The DLL functions are executed (called) according to the configuration. External dll cooperates with other parts of runtime applications over **shared memory operands** area as shown in the Figure Nr. 1 and Figure Nr. 2.

OpcDbGateway

For the OpcDbGgateway, using of external (enhancing) dll is a way how to provide high flexibility together with high productivity of the application integration enabled by the configuring.

OpcDbGateway*xternal DLL - OpcDbGateway - interconnecting with OPC server and OPC client parts over memory operands.*

SAEAUT Universal OPC Server

Using of external Dll is inevitable mainly in SAEAUT Universal OPC Server. It is a way to create a specialised OPC server from the universal OPC server.

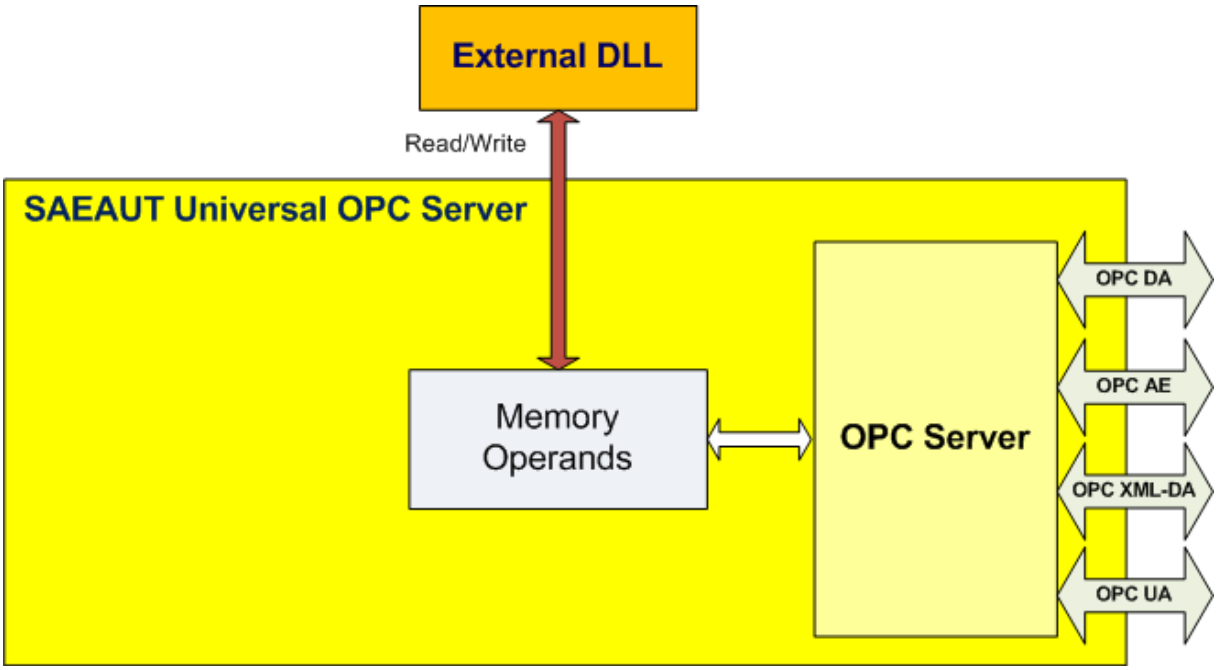


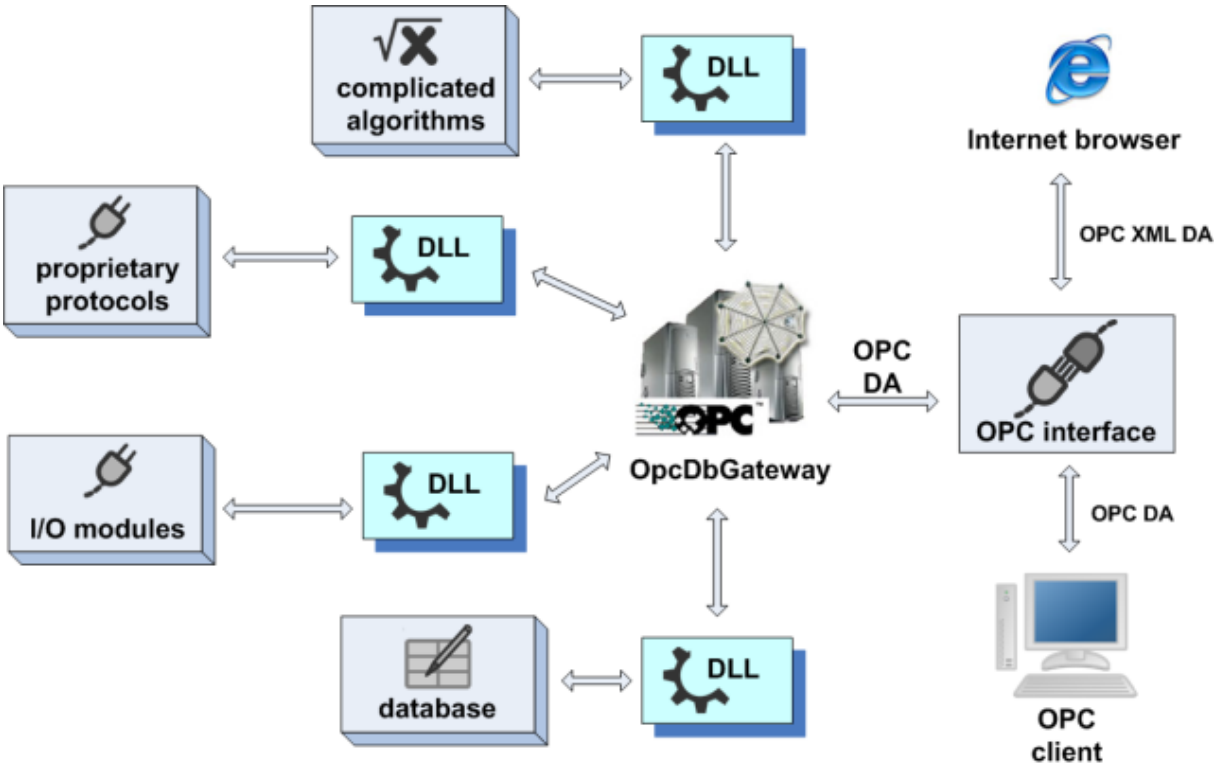
Figure: *External DLL - SAEAUT Universal OPC Server - interconnecting with OPC server and OPC client parts over memory operands.*

Essential advantages resulting from integration of "External DLL" to the OpcDbGateway and SAEAUT Universal OPC Server application:

- enhancement of OpcDbGateway and SAEAUT Universal OPC Server functionality,
- possibility for customers to build their own product.

The picture below demonstrates what benefits brings the conjunction of external DLL with OpcDbGateway and SAEAUT Universal OPC Server. If we split the picture to more small parts then we get several new scenarios how the OpcDbGateway and SAEAUT Universal OPC Server can be used:

- Usage of External DLL enhances functionality of OpcDbGateway and SAEAUT Universal OPC Server,
- OpcDbGateway and SAEAUT Universal OPC Server may be used as a bridge between DLL and DLL,
- OpcDbGateway and SAEAUT Universal OPC Server may be used as a bridge between DLL and OPC server,
- OpcDbGateway and SAEAUT Universal OPC Server may be used as a bridge between DLL and OPC client,
- OpcDbGateway and SAEAUT Universal OPC Server may be used as a bridge between DLL and web client.



More details about external DLLs can be found in the following topics:
Interface of external DLL
In this topic you can find all information about functions.

How to build your own OpcDbGateway compatible DLL?
In this topic you can find all information how to build your own DLL which may be called from OpcDbGateway and SAEAUT Universal OPC Server.

How to call en external DLL from OpcDbGateway?
In this topic you can find all information how to call an external DLL directly from OpcDbGateway and SAEAUT Universal OPC Server.

See Also
Configuring External DLLs, Operation CALL DLL

Related articles on the web
Customization of the runtime and configuration

5.1 Scenarios: How to create application using DLL?

We can use three methods:

1. As a called function with input and output arguments – in that case the function implemented within DLL is called by configurable command *CALL DLL*
2. As independent thread(s) – communication with threads of the runtime application used for configured and implicit functionality is executed over **shared memory operands**.

3. Combination of two previous methods.

All mentioned methods go out from assumption that external DLL is allready mapped in configuration.

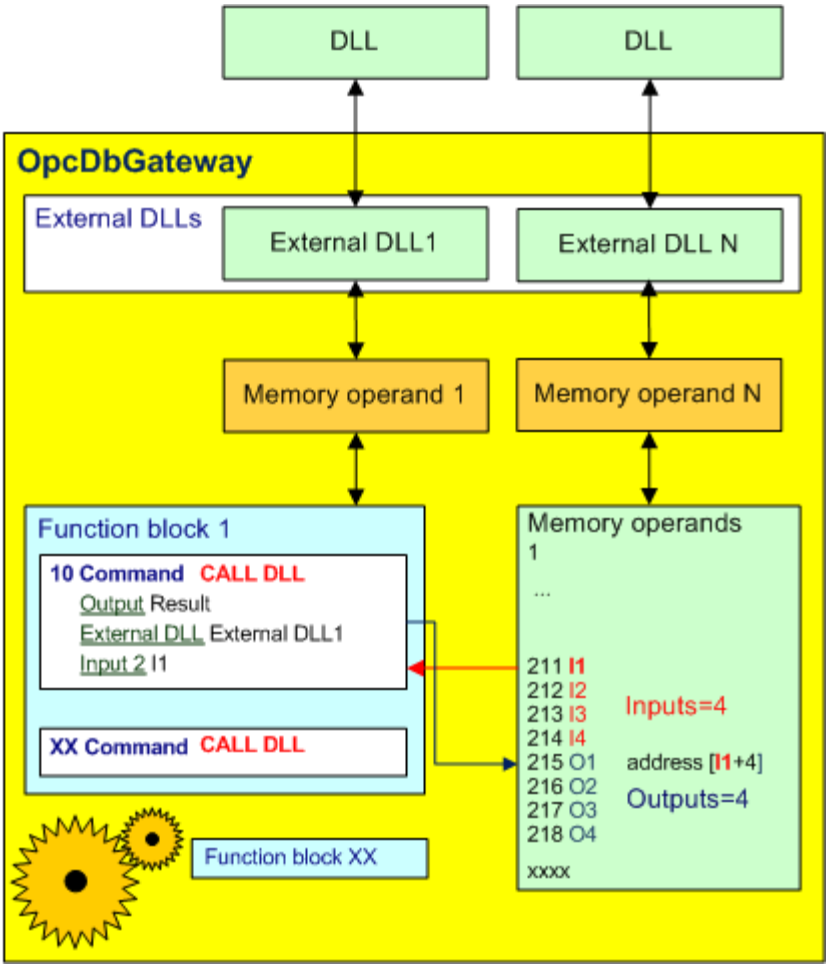


Figure: *Usage external DLL with OpcDbGateway (or SAEAUT Universal OPC Server).*

See Also

Operation CALL DLL,
External DLL module,
Configuring external DLLs,
Interface of external DLL,
How to build your own OpcDbGateway compatible DLL?

5.1.1 Calling function in DLL using CALL DLL configurable command

This method enables using of different functionalities (not implemented as configurable commands) within function blocks. The functionality is called explicitly by configurable command CALL DLL. The functionality within dll is executed only in the time when it is explicitly called contrary to a functionality continually executed within dll. This method is useful e.g. for implementing of complex mathematical functions. (Continual functionality on the other hand is useful e.g. for implementation of communication drivers).

The CALL DLL operation is used for execution the DoProcessIO function from mapped DLL. Please

do these steps as follows:

1. The number of input/output memory operands have to be the same as it is defined in DLL in the GetCountOfIO function (please see *Input parameters*, *Output parameters*). The set of used memory operands must follow one after another in the Process memory (e.g. 201, 202, 203, etc.). The memory operand with lowest address is most important (e.g. I1), because this operand will be used in CALL DLL command (e.g. **20 CALL DLL DllExample**).

```
DllExport void WINAPI GetCountOfIO(LPWORD lpInputs, LPWORD lpOutputs)
{

    // TODO: Set your count of inputs and outputs

    // EXAMPLE: Inputs=4, Outputs=4
    // BEGIN {
    *lpInputs  = 4;          // TODO: Set your count of inputs
    *lpOutputs = 4;          // TODO: Set your count of outputs
    // } END

    return;
}
```

Figure: *External DLL: Number of used Memory operands*

For example: You can add memory operands as on the picture below.

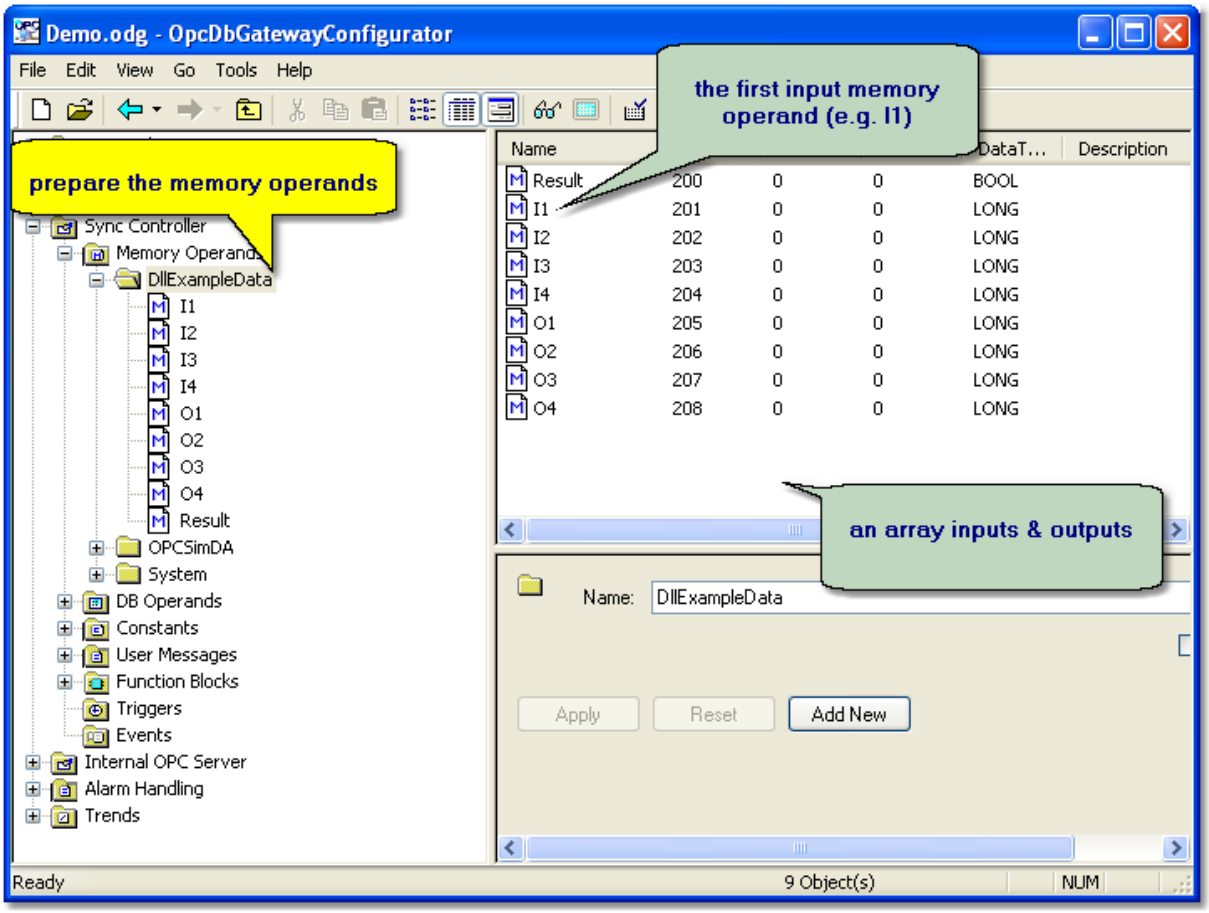


Figure: *External DLL: Memory operands*

2. Add a new command that uses CALL DLL operation to call Example1.dll. Click on a Function block (e.g. **Main**) and add a new command (e.g. **20 CALL DLL DllExample**) as on the pictures below.

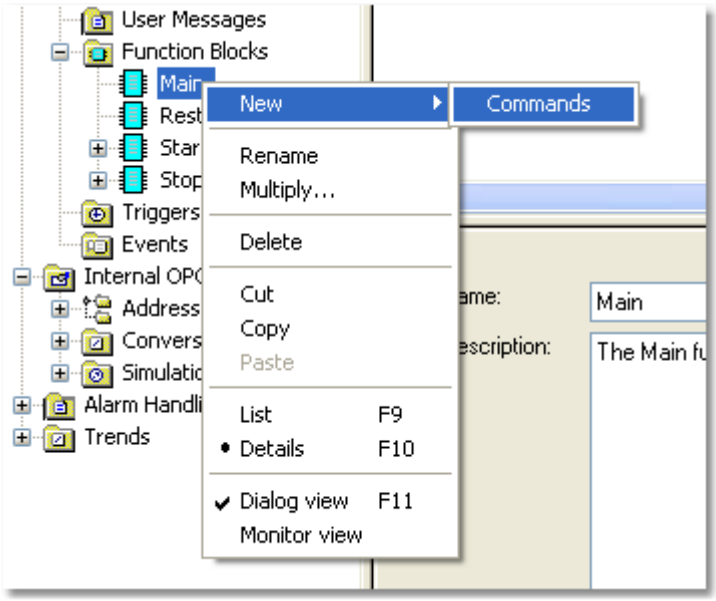


Figure 3: *A new Command.*

The memory operand with lowest address is most important (e.g. I1), because this operand will be used as input parameter in CALL DLL command (e.g. **20 CALL DLL DllExample**).

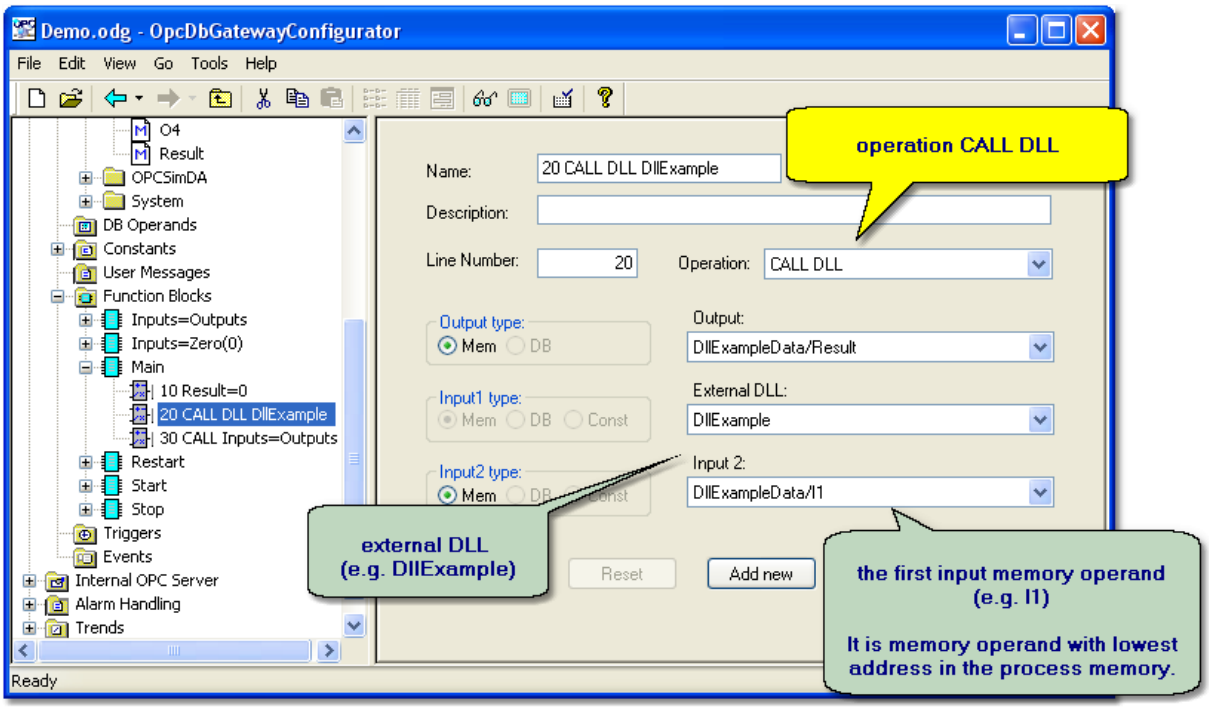


Figure: *A new Command definition (e.g. 20 CALL DLL DllExample).*

3. Commands with defined CALL DLL operation executes the DoProcessIO function from your external DLL.

```
DllExport bool WINAPI DoProcessIO(const CComVariant* lpInputs, WORD wInCnt,
                                  CComVariant* lpOutputs, WORD wOutCnt)
{
    // TODO: Add your functionality

    // EXAMPLE: Output[i] = Inputs[i] + 1
    // BEGIN {
    for(DWORD i=0; i < wInCnt; i++)
    {
        CComVariant value = lpInputs[i];
        value.ChangeType(VT_I4);

        ATLTRACE("\nDLL: i:%d VARTYPE=%d Value=%d", i, value.vt, value.lVal);
        lpOutputs[i] = value.lVal + 1;
    }
    // } END

    return 1; // OK
}
```

write your "requested functionality"

IN: values 'lpInputs' will be passed from OpcDbGateway

OUT: values from 'lpOutputs' will be passed to OpcDbGateway

Figure 5: *The DoProcessIO function defined in source codes.*

See Also

Calling DLL function from OpcDbGateway, Operation CALL DLL, Configuring external DLLs

5.1.2 Cooperation of runtime core with DLL over memory operands

This method enables executing an activity within thread(s) implemented in DLL continually (from start till stop of the runtime application) and independently on runtime core functionality (configured and implicit – no requiring configuring or programming). Coordination between functionality implemented in DLL and in the runtime core is provided through shared memory operands. Memory operands can be used e.g. in triggers of the type value to start a functionality configured in function blocks (as event) or in opposite direction - using a memory operand set by configured functionality for calling a function implemented in DLL. Memory operands can be used also to transfer data from communication driver implemented in DLL or for transfer of an error message from DLL to the logging provided by the runtime core.

Each enhancing DLL provides a specific pointer that enables to read/write data from/to the memory operands. The pointer to memory operands is ailable in the OnInitMemory DLL function (see Figure 1).

```
//*************************************************************************
// @mfunc   bool | OnInitMemory | The function is called after the main proce
//
// @parm    CProcessImageMemory* | lpPIM  | [in] Pointer to the buffer that i
//
// @rdesc   Returns void
//
// @comm    The function is called after the main process application memry i
//
// @history 02.06.2011 13:11, created by SAE-Automation,s.r.o.
//*************************************************************************
DllExport void WINAPI OnInitMemory(CProcessImageMemory* lpPIM)
{
    // DO NOT CHANGE THIS FUNCTION
    g_pProcessImageMemory = lpPIM;

    return;
}
```

Figure 1: *The OnInitMemory function provides a pointer to memory operands area.*

The Figure 2 shows example reading a value from memory operand (SYSADDR_PLC_PERIOD_COUNTER = 7) and writing this value to memory operand defined by address 5001. In addition, this value is incremented about 10 and written also to memory operand on address 5002.

```
// Example of worked thread.
UINT IOThread(LPVOID pParam)
{
    while(!g_bCloseThread)
    {
        Sleep(1000);

        if (g_pProcessImageMemory != NULL)
        {
            // Example 1: How to Read a item value.
            CProcessValue PV;
            g_pProcessImageMemory->Read(CProcessImageMemory::SYSADDR_PLC_PER
            ATLTRACE(_T("\nDLL Example 1: Read value(%d)=%u\n"), CProcessImag


            // Example 2: How to Write a new item value.
            g_pProcessImageMemory->Write(5001, PV);
            ATLTRACE(_T("\nDLL Example 2: Write value(%d)=%u\n"), 5001, PV.v

            // Example 3: How to Write a new item value.
            CComVariant CV(PV.vValue.ulVal + 10);
            g_pProcessImageMemory->Write(5002, CV);
            ATLTRACE(_T("\nDLL Example 3: Write value(%d)=%u\n"), 5002, CV.u
        }
    }

    return 1;
}
```

Figure 2: *Example of reading/writing data through memory operands.*

Memory Operands used in DLL have to be defined in ODG configuration

The Figure 3 below shows definition of the PLCPeriodCounter_Copy memory operand with memory address 5001. If you will read/write memory operand from DLL then you have to consider also suitable memory operand data type. The Detailed description how to add a new memory operand to configuration you can find in the Memory operands topic.

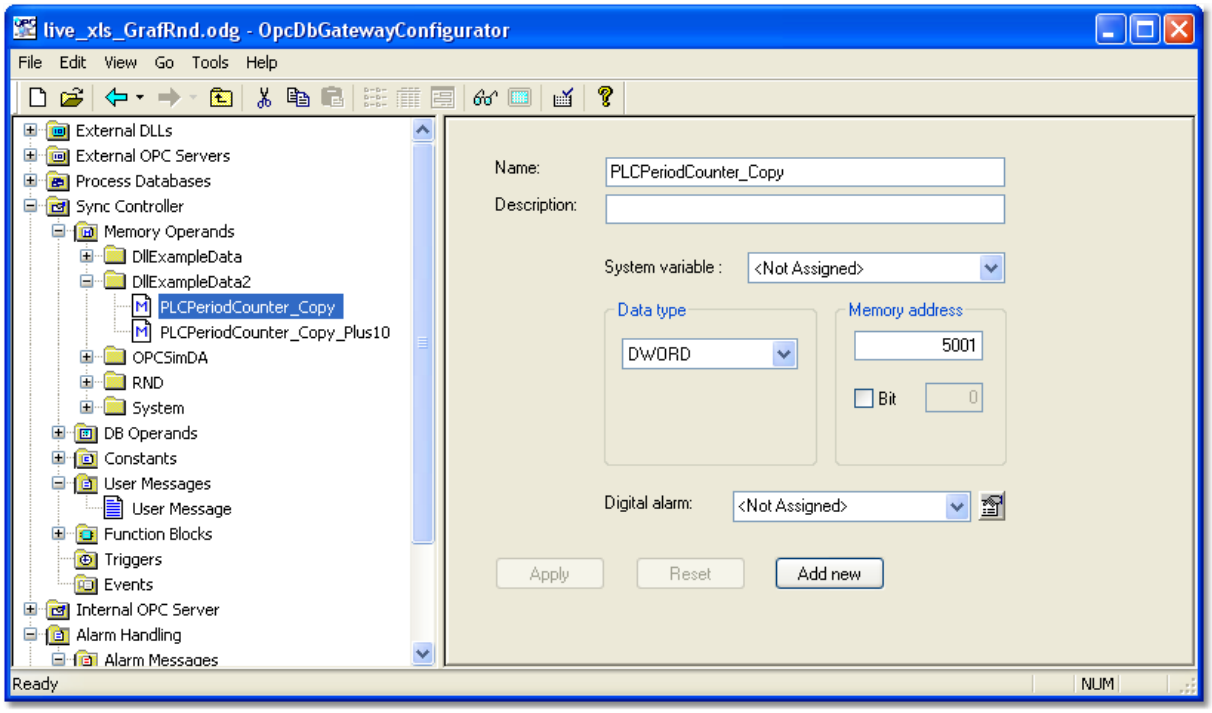


Figure 3: *Definition of the PLCPeriodCounter_Copy memory operand in the configuation file.*

See Also

Calling DLL function from OpcDbGateway, Operation CALL DLL, Configuring external DLLs

5.2 Example: How to build your external DLL?

PREREQUISITES

We recomend you to build a new DLL in environment Microsoft Visual C++ (Visual Studio 2005 or newer). The simplest way how to build own DLL is to modify existing example source codes.

BUILDING DLL STEP-BY-STEP

The goal of the section is to show you how to build your external DLL, which may be called from OpcDbGateway and SAEAUT Universal OPC Server. The sample goes out from description of individual parts of already existing **Example1.dll**. The Example1.dll is a very simple **external DLL**, which may be used as a **template for developing your own external DLLs**. The Example1.dll binary file and Visual Studio project **source codes** (C++) are part of OpcDbGateway and SAEAUT Universal OPC Server installation package and will be available in application's directory (see ..*OpcDbGateway\Examples\ExternalDlls\Example1\Example1.sln*).

Note: Example.dll is installed with OpcDbGateway and SAEAUT Universal OPC Server and an example of usage can be seen directly in DEMO.odg project.

Please build your external DLL step by step accordingly as follows:

1. Please make your own copy of Example1 project.
2. Open your project in Microsoft Visual Studio .NET 2003.
3. Update the GetProductName function.
4. Update the GetProductVersion function.

5. Update the GetCompanyName function.
6. Update the GetLegalCopyright function.
7. Update the GetDescription function.
8. Update the GetCountOfIO function. This function defines the count of inputs/outputs for the main process function DoProcessIO.
9. Update the DoProcessIO function. This is the most important function when OpcDbGateway and SAEAUT Universal OPC Server is running in real-time operation mode.
10. Update the OnInitMemory function. If you want share PIM directly.
11. Update the OnStart function.
12. Update the OnStop function.
13. Build your project Microsoft Visual Studio .NET 2003.
14. Copy your new DLL to the application directory (*OpcDbGateway\ExternalDll*).
15. Open the OpcDbGateway and SAEAUT Universal OPC Server configurator.
16. Add a new External DLL item.
17. Select your DLL in combo-box **File name**.

See Also

External DLL module, Interface of external DLL, How to call en external DLL from OpcDbGateway?

5.2.1 example GetProductName

The function retrieves a DLL product name as "External DLL sample" string.

```
DllExport DWORD WINAPI GetProductName(LPTSTR lpReturnedString, DWORD dwSize)
{
    // TODO: Add your product name

    // EXAMPLE: "1.0.0.0"
    // BEGIN {
    const TCHAR* ReturnedString = _T("External DLL sample");

    if( _tcslen(ReturnedString) < dwSize ) {
        _tcscpy(lpReturnedString, ReturnedString);
    }
    else {

        if( !dwSize )
            return 0;

        _tcsncpy(lpReturnedString, ReturnedString, dwSize - 1);
    }
    // } END

    return (DWORD)_tcslen(lpReturnedString);
}
```

write your "Product name"

Figure 1: *Example of DLL function definitions: GetProductName function*

See Also

Function GetProductName description, Interface of external DLL, External DLL module

5.2.2 example GetProductVersion

The function retrieves a DLL product version as "1.0.0.0" string.

```
DllExport DWORD WINAPI GetProductVersion(LPTSTR lpReturnedString, DWORD dwSize)
{
    // TODO: Add your product version

    // EXAMPLE: "1.0.0.0"
    // BEGIN {
    const TCHAR* ReturnedString = _T("1.0.0.0");

    if( _tcslen(ReturnedString) < dwSize ) {
        _tcscpy(lpReturnedString, ReturnedString);
    }
    else {

        if( !dwSize )
            return 0;

        _tcsncpy(lpReturnedString, ReturnedString, dwSize - 1);
    }
    // } END

    return (DWORD)_tcslen(lpReturnedString);
}
```

write your "Product version"

Figure 1: *Example of DLL function definitions: GetProductVersion function*

See Also

Function GetProductVersion description, Interface of external DLL, External DLL module

5.2.3 example GetCompanyName

The function retrieves a DLL company name as "SAE - Automation, s.r.o." string.

```
DllExport DWORD WINAPI GetCompanyName(LPTSTR lpReturnedString, DWORD dwSize)
{
    // TODO: Add your company name

    // EXAMPLE: "SAE - Automation, s.r.o."
    // BEGIN {
    const TCHAR* ReturnedString = _T("SAE - Automation, s.r.o.");

    if( _tcslen(ReturnedString) < dwSize ) {
        _tcscpy(lpReturnedString, ReturnedString);
    }
    else {

        if( !dwSize )
            return 0;

        _tcsncpy(lpReturnedString, ReturnedString, dwSize - 1);
    }
    // } END

    return (DWORD)_tcslen(lpReturnedString);
}
```

write your "Company name"

Figure 1: *Example of DLL function definitions: GetCompanyName function*

See Also

Function GetCompanyName description, Interface of external DLL, External DLL module

5.2.4 example GetLegalCopyright

The function retrieves a DLL copyright as "Copyright © 2002-2007 SAE - Automation, s.r.o. All rights reserved." string.

```
DllExport DWORD WINAPI GetLegalCopyright(LPTSTR lpReturnedString, DWORD dwSize)
{
    // TODO: Add your Copyright

    // EXAMPLE: "Copyright © 2002-2007 SAE - Automation, s.r.o. All rights reserved."
    // BEGIN {
    const TCHAR* ReturnedString = _T("Copyright © 2002-2007 SAE - Automation, s.r.o. All

    if( _tcslen(ReturnedString) < dwSize ) {
        _tcscpy(lpReturnedString, ReturnedString);
    }
    else {

        if( !dwSize )
            return 0;

        _tcsncpy(lpReturnedString, ReturnedString, dwSize - 1);
    }
    // } END

    return (DWORD)_tcslen(lpReturnedString);
}
```

write your "Legal copyright"

Figure 1: *Example of DLL function definitions: GetLegalCopyright function*

See Also

Function GetLegalCopyright description, Interface of external DLL, External DLL module

5.2.5 example GetDescription

The function retrieves a DLL description as "Output[i] = Inputs[i] + 1" string.

```
DllExport DWORD WINAPI GetDescription(LPTSTR lpReturnedString, DWORD dwSize)
{
    // TODO: Add your external DLL description

    // EXAMPLE: "Output[i] = Inputs[i] + 1"
    // BEGIN {
    const TCHAR* ReturnedString = _T("Output[i] = Inputs[i] + 1");

    if( _tcslen(ReturnedString) < dwSize ) {
        _tcscpy(lpReturnedString, ReturnedString);
    }
    else {

        if( !dwSize )
            return 0;

        _tcsncpy(lpReturnedString, ReturnedString, dwSize - 1);
    }
    // } END

    return (DWORD)_tcslen(lpReturnedString);
}
```

write your "Description"

Figure 1: *Example of DLL function definitions: GetDescription function*

See Also
Function GetDescription description, Interface of external DLL, External DLL module

5.2.6 example GetCountOfIO

The function retrieves count used inputs and outputs (inputs = 4/outputs = 4).

```
DllExport void WINAPI GetCountOfIO(LPWORD lpInputs, LPWORD lpOutputs)
{

    // TODO: Set your count of inputs and outputs

    // EXAMPLE: Inputs=4, Outputs=4
    // BEGIN {
    *lpInputs  = 4;          // TODO: Set your count of inputs
    *lpOutputs = 4;          // TODO: Set your count of outputs
    // } END

    return;
}
```

write your count of "Inputs & Outputs"

Figure 1: *Example of DLL function definitions: GetCountOfIO function*

See Also
Function GetCountOfIO description, Interface of external DLL, External DLL module

5.2.7 example DoProcessIO

The function processes all values from input buffer (lpInputs) and retrieves the processed values into output buffer (lpOutputs).

```
DllExport bool WINAPI DoProcessIO(const CComVariant* lpInputs, WORD wInCnt,
                                  CComVariant* lpOutputs, WORD wOutCnt)
{
    // TODO: Add your functionality

    // EXAMPLE: Output[i] = Inputs[i] + 1
    // BEGIN {
    for(DWORD i=0; i < wInCnt; i++)
    {
        CComVariant value = lpInputs[i];
        value.ChangeType(VT_I4);

        ATLTRACE("\nDLL: i:%d VARTYPE=%d Value=%d", i, value.vt, value.lVal);
        lpOutputs[i] = value.lVal + 1;
    }
    // } END

    return 1; // OK
}
```

write your "requested functionality"

IN: values 'lpInputs' will be passed from OpcDbGateway

OUT: values from 'lpOutputs' will be passed to OpcDbGateway

Figure 1: *Example of DLL function definitions: DoProcessIO function*

See Also

Function DoProcessIO description, Interface of external DLL, External DLL module

5.2.8 example OnInitMemory

The function enables to share the Process Image Memory of OpcDbGateway (or SAEAUT Universal OPC Server) via parameter input lpPIM and sets global variable g_pProcesImageMemory (Figure Nr.1). Then, the Figure Nr.2 shows example reading/writing data via g_pProcessImageMemory.

```
//*************************************************************************
// @mfunc   bool | OnInitMemory | The function is called after the main proce
//
// @parm    CProcessImageMemory* | lpPIM  | [in] Pointer to the buffer that i
//
// @rdesc   Returns void
//
// @comm    The function is called after the main process application memry i
//
// @history 02.06.2011 13:11, created by SAE-Automation,s.r.o.
//*************************************************************************
DllExport void WINAPI OnInitMemory(CProcessImageMemory* lpPIM)
{
    // DO NOT CHANGE THIS FUNCTION
    g_pProcessImageMemory = lpPIM;

    return;
}
```

Figure 1: *Example of DLL function definitions: OnInitMemory function*

```
// Example of worked thread.
UINT IOThread(LPVOID pParam)
{
    while(!g_bCloseThread)
    {
        Sleep(1000);

        if (g_pProcessImageMemory != NULL)
        {
            // Example 1: How to Read a item value.
            CProcessValue PV;
            g_pProcessImageMemory->Read(CProcessImageMemory::SYSADDR_PLC_PER
            ATLTRACE(_T("\nDLL Example 1: Read value(%d)=%u\n"), CProcessImag


            // Example 2: How to Write a new item value.
            g_pProcessImageMemory->Write(5001, PV);
            ATLTRACE(_T("\nDLL Example 2: Write value(%d)=%u\n"), 5001, PV.vV

            // Example 3: How to Write a new item value.
            CComVariant CV(PV.vValue.ulVal + 10);
            g_pProcessImageMemory->Write(5002, CV);
            ATLTRACE(_T("\nDLL Example 3: Write value(%d)=%u\n"), 5002, CV.ul
        }
    }

    return 1;
}
```

Figure 2: *Example of reading/wiring data.*

See Also

Function DoProcessIO description, Interface of external DLL, External DLL module

5.2.9 example OnStart

In this example, the function starts the working thread.

```
//*************************************************************************
// @mfunc   bool | OnStart | This function is called after main process appli
//
// @rdesc   Returns void
//
// @comm    You can use this function to perform tasks such as allocating res
//
// @history 02.06.2011 13:11, created by SAE-Automation,s.r.o.
//*************************************************************************
DllExport void WINAPI OnStart()
{

    g_bCloseThread = false;

    // @flow0 | create suspended IOThread
    g_pIOThread = AfxBeginThread(IOThread,
                                 (LPVOID) NULL,
                                 THREAD_PRIORITY_NORMAL,
                                 0,
                                 0);

    return;
}
```

Figure 1: *Example of DLL: The function strats working thread.*

See Also

Function DoProcessIO description, Interface of external DLL, External DLL module

5.2.10 example OnStop

In this example, the function terminates the working thread.

```
//*************************************************************************
// @mfunc   bool | OnStop | This function is called after main process appli
//
// @rdesc   Returns void
//
// @comm    You can use this function to perform tasks such as releasing res
//
// @history 02.06.2011 13:11, created by SAE-Automation,s.r.o.
//*************************************************************************
DllExport void WINAPI OnStop()
{
    g_bCloseThread = true;

    return;
}
```

Figure 1: *Example of DLL: The function terminates working thread.*

See Also

Function DoProcessIO description, Interface of external DLL, External DLL module

5.3 Mapping DLL to configuration

First, please be aware that you may map only DLLs from this directory:

OpcDbGateway
..\OpcDbGateway\ExternalDll

SAEAUT Universal OPC Server
..\SAEAUT Universal OPC Server\ExternalDll

Please copy your DLL to this directory and make mapping of your external DLL step by step as follows:

1. Click on **start** → **Programs** → **OpcDbGateway and SAEAUT Universal OPC Server** → **OpcDbGateway and SAEAUT Universal OPC Server Configurator**.

Figure 21: *Launching OpcDbGateway Configurator*

2. Click on **External DLL** → **New** → **External DLL**.

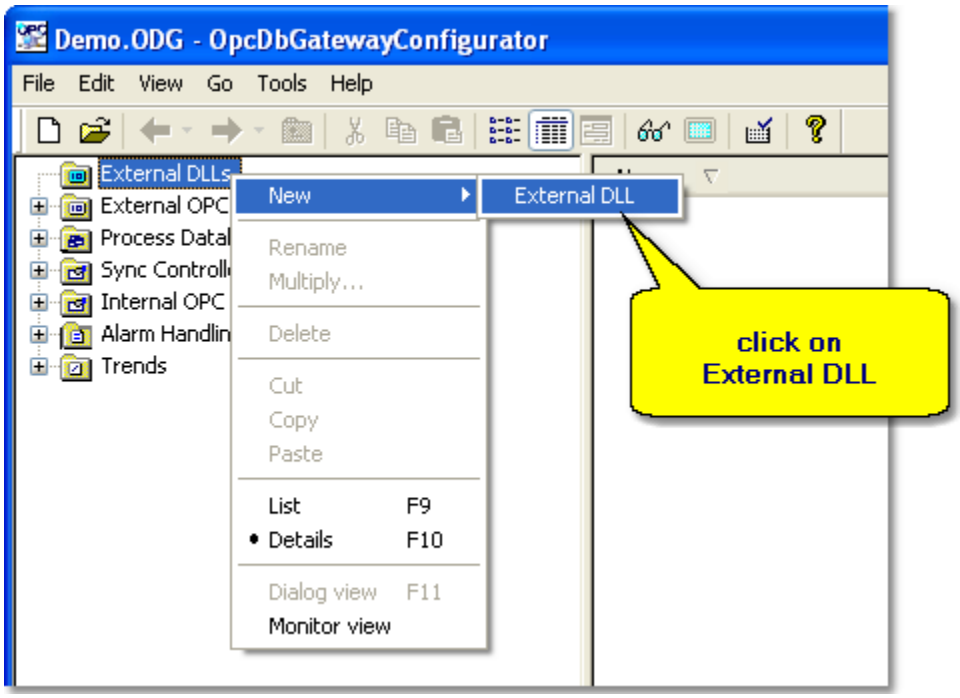


Figure 22: *A new external DLL.*

3. Write a symbolic **Name** for DLL (e.g. **DllExample**).
4. Select proper **File Name**. (e.g. **Example1.dll**. The combo-box shows only DLL files located in the ***..\OpcDbGateway\ExternalDll*** directory.)
5. Click on the **Apply** button.

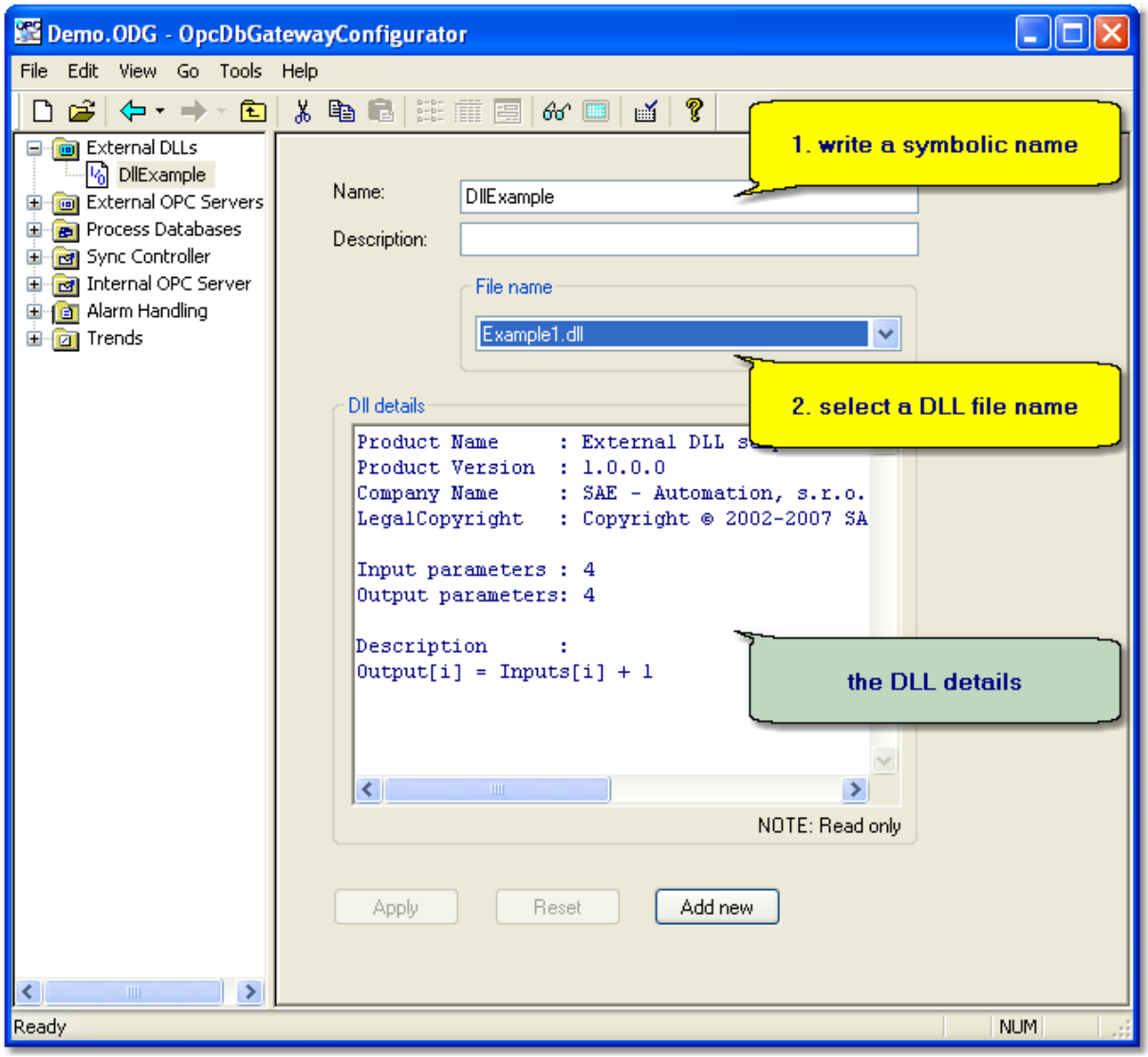


Figure 23: *A new external DLL.*

Name	External DLL symbolic name.
Description	Brief external DLL description.
File name	The DLL file name.
Dll details	Brief description obtained directly from DLL library.

Table 1 - External DLLs parameters

7. If you want to continue in the second step please click on Calling DLL function from OpcDbGateway topic.

See Also
Calling DLL function from OpcDbGateway, Operation CALL DLL

5.4 Interface of external DLL

The programmer API interface defines a set of function definitions which are available for external DLL. Each external DLL that is called from OpcDbGateway and SAEAUT Universal OPC Server has to include following exported functions:

- GetProductName
- GetProductVersion
- GetCompanyName
- GetLegalCopyright
- GetDescription
- GetCountOfIO The function defines the count of inputs/outputs for the main process function DoProcessIO.
- DoProcessIO This function is called only in runtime module.
- OnInitMemory This function is called only in runtime module.
- OnStart This function is called only in runtime module.
- OnStop This function is called only in runtime module.

```
#define DllExport extern "C" __declspec (dllexport)
#define DllImport extern "C" __declspec (dllimport)

class CProcessImageMemory;

////////////////////////////////////////////////////////////////////////////////
// EXTERNAL DLL INTERFACE
////////////////////////////////////////////////////////////////////////////////
// The function retrieves a DLL product name as string.
DllExport DWORD WINAPI GetProductName       (LPTSTR lpReturnedString, DWORD dwSize);
// The function retrieves a DLL product version as string.
DllExport DWORD WINAPI GetProductVersion    (LPTSTR lpReturnedString, DWORD dwSize);
// The function retrieves a DLL company name as string.
DllExport DWORD WINAPI GetCompanyName       (LPTSTR lpReturnedString, DWORD dwSize);
// The function retrieves a DLL copyright as string.
DllExport DWORD WINAPI GetLegalCopyright    (LPTSTR lpReturnedString, DWORD dwSize);
// The function retrieves a DLL description as string.
DllExport DWORD WINAPI GetDescription       (LPTSTR lpReturnedString, DWORD dwSize);
// The function retrieves a specified count of inputs/outputs.
DllExport void  WINAPI GetCountOfIO         (LPWORD lpInputs, LPWORD lpOutputs);
// The function processes all values from input buffer and retrieves the processed values
DllExport bool  WINAPI DoProcessIO          (const CComVariant* lpInputs, WORD wInCnt,
                                             CComVariant* lpOutputs, WORD wOutCnt);
// The function is called after the main process application memory is sucessfully initial
DllExport void  WINAPI OnInitMemory         (CProcessImageMemory* lpPIM);
// This function is called after main process application thread is sucessfully started.
DllExport void  WINAPI OnStart              ();
// This function is called after main process application thread is sucessfully stopped.
DllExport void  WINAPI OnStop               ();
```

Figure 1: *Example of DLL interface.*

See Also

External DLL module, How to build your own OpcDbGateway compatible DLL?, How to call en external DLL from OpcDbGateway?

5.4.1 GetProductName

The function retrieves a DLL product name as string.

```
DllExport DWORD WINAPI GetProductName(
  LPTSTR lpReturnedString,
  DWORD dwSize
);
```

Parameters

lpReturnedString

[out] Pointer to the buffer that receives the retrieved string.

dwSize

[in] Size of the buffer pointed to by the lpReturnedString parameter, in TCHARs.

Return Values

The return value is the number of characters copied to the buffer, not including the terminating null character.

If the supplied destination buffer is too small to hold the requested string, the string is truncated and followed by a null character, and the return value is equal to dwSize minus one.

Remarks

The function will be called from OpcDbGateway configurator module.

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Example

example GetProductName

See Also

Function GetProductName sample, Interface of external DLL, External DLL module

5.4.2 GetProductVersion

The function retrieves a DLL product version as string.

```
DllExport DWORD WINAPI GetProductVersion(
  LPTSTR lpReturnedString,
  DWORD dwSize
);
```

Parameters

lpReturnedString

[out] Pointer to the buffer that receives the retrieved string.

dwSize

[in] Size of the buffer pointed to by the lpReturnedString parameter, in TCHARs.

Return Values

The return value is the number of characters copied to the buffer, not including the terminating null character.

If the supplied destination buffer is too small to hold the requested string, the string is truncated and followed by a null character, and the return value is equal to dwSize minus one.

Remarks

The function will be called from OpcDbGateway configurator module.

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Example

example GetProductVersion

See Also

Function GetProductVersion sample, Interface of external DLL, External DLL module

5.4.3 GetCompanyName

The function retrieves a DLL company name as string.

```
DllExport DWORD WINAPI GetCompanyName(
  LPTSTR lpReturnedString,
  DWORD dwSize
);
```

Parameters

lpReturnedString

[out] Pointer to the buffer that receives the retrieved string.

dwSize

[in] Size of the buffer pointed to by the lpReturnedString parameter, in TCHARs.

Return Values

The return value is the number of characters copied to the buffer, not including the terminating null character.

If the supplied destination buffer is too small to hold the requested string, the string is truncated and followed by a null character, and the return value is equal to dwSize minus one.

Remarks

The function will be called from OpcDbGateway configurator module.

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Example

example GetCompanyName

See Also

Function GetCompanyName sample, Interface of external DLL, External DLL module

5.4.4 GetLegalCopyright

The function retrieves a DLL copyright as string.

```
DllExport DWORD WINAPI GetLegalCopyright(
  LPTSTR lpReturnedString,
  DWORD dwSize
);
```

Parameters

lpReturnedString

[out] Pointer to the buffer that receives the retrieved string.

dwSize

[in] Size of the buffer pointed to by the lpReturnedString parameter, in TCHARs.

Return Values

The return value is the number of characters copied to the buffer, not including the terminating null character.

If the supplied destination buffer is too small to hold the requested string, the string is truncated and followed by a null character, and the return value is equal to dwSize minus one.

Remarks

The function will be called from OpcDbGateway configurator module.

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Example

example GetLegalCopyright

See Also

Function GetLegalCopyrigt sample, Interface of external DLL, External DLL module

5.4.5 GetDescription

The function retrieves a DLL description as string.

```
DllExport DWORD WINAPI GetDescription(
  LPTSTR lpReturnedString,
  DWORD dwSize
);
```

Parameters

lpReturnedString

[out] Pointer to the buffer that receives the retrieved string.

dwSize

[in] Size of the buffer pointed to by the lpReturnedString parameter, in TCHARs.

Return Values

The return value is the number of characters copied to the buffer, not including the terminating null character.

If the supplied destination buffer is too small to hold the requested string, the string is truncated and followed by a null character, and the return value is equal to dwSize minus one.

Remarks

The function will be called from OpcDbGateway configurator module.

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Examples

example GetDescription

See Also

Function GetDescription sample, Interface of external DLL, External DLL module

5.4.6 GetCountOfIO

The function retrieves a specified count of inputs/outputs.

```
DllExport void WINAPI GetCountOfIO(
  LPWORD lpInputs,
  LPWORD lpOutputs
);
```

Parameters

lpInputs

[out] Pointer to count of input values.

lpOutputs

[out] Pointer to count of output values

Return Values

The return value is void.

Remarks

The function will be called from OpcDbGateway configurator module and runtime module. The function defines count of inputs/outputs for the main process function DoProcessIO.

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Examples

example GetCountOfIO

See Also

Function GetCountOfIO sample, Interface of external DLL, External DLL module

5.4.7 DoProcessIO

The function processes all values from input buffer and retrieves the processed values into output buffer.

```
DllExport bool WINAPI DoProcessIO(
     const CComVariant* lpInputs, WORD wInCnt,
           CComVariant* lpOutputs, WORD wOutCnt
);
```

Parameters

lpInputs

[in] Pointer to the buffer that inludes input values.

wInCnt

[in] Size of the buffer pointed to by the lpInputs parameter, in CComVariant

lpOutputs
[out]Pointer to the buffer that recieves output values.

wOutCnt
[in] Size of the buffer pointed to by the lpOutputs parameter, in CComVariant

Return Values
If the function succeeds, the return value is nonzero.
If the function fails, the return value is zero.

Remarks
The function will be called from OpcDbGateway runtime module as Function Blocks/Command. The most important function when OpcDbGateway and SAEAUT Universal OPC Server is running in real-time operation mode. The count of incoming inputs & outputs is the same as it is defined in function DoCountOfIO.

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Example
example DoProcessIO

See Also
Function DoProcessIO sample, Interface of external DLL, External DLL module

5.4.8 OnInitMemory

The function enables to share the Process Image Memory of OpcDbGateway (or SAEAUT Universal OPC Server).

```
DllExport void WINAPI DoProcessIO(CProcessImageMemory* lpPIM);
```

Parameters
lpPIM
[in] A pointer to the Process Image Memory of OpcDbGateway (or SAEAUT Universal OPC Server).

Return Values
The return value is void.

Remarks
In the Process Image Memory are stored all current data used by OpcDbGateway (or SAEAUT Universal OPC Server). Using this *lpPIM* pointer, the external DLL can read/write data from/to working memory of OpcDbGateway (resp. SAEAUT Universal OPC Server).

In the function description is used *DllExport*. *DllExport* is defined as follows:

```
#define DllExport extern "C" __declspec (dllexport)
```

Example
example OnInitMemory

See Also
Function DoProcessIO sample, Interface of external DLL, External DLL module

5.4.9 OnStart

This function notifies external DLL that the OpcDbGateway (or SAEAUT Universal OPC Server) runtime module just started the main SoftPLC loop.

```
DllExport void WINAPI OnStart();
```

Parameters
None

Return Values
The return value is void.

Remarks
You can use this function in order to allocate specific resources used by DLL.

In the function description is used *DllExport*. *DllExport* is defined as follows:
```
#define DllExport extern "C" __declspec (dllexport)
```

Example
example OnStart

See Also
Function DoProcessIO sample, Interface of external DLL, External DLL module

5.4.10 OnStop

This function notifies external DLL that the OpcDbGateway (or SAEAUT Universal OPC Server) runtime module just stopped the main SoftPLC loop.

```
DllExport void WINAPI OnStop();
```

Parameters
None

Return Values
The return value is void.

Remarks
You can use this function in order to release specific resources used by DLL.

In the function description is used *DllExport*. *DllExport* is defined as follows:
```
#define DllExport extern "C" __declspec (dllexport)
```

Example
example OnStop

See Also
Function DoProcessIO sample, Interface of external DLL, External DLL module

OpcDbGateway and SAEAUT Universal OPC Server

Part VI

6 Configuring and programming

Structure of the OpcDbGateway configuration database is organized into following folders:

- External DLLs
- + External OPC servers
- + Process databases
  - Process tables
  - Queries
- + Sync Controller
  - Memory operands
  - DB Operands
  - Constants
  - User messages
  - Function blocks
  - Triggers
  - Events
- + Internal OPC Server
  - Address space
  - Conversions
  - Simulation signals
- + Alarm handling
  - Alarm messages
  - Alarm definitions MOP
- DDE Servers

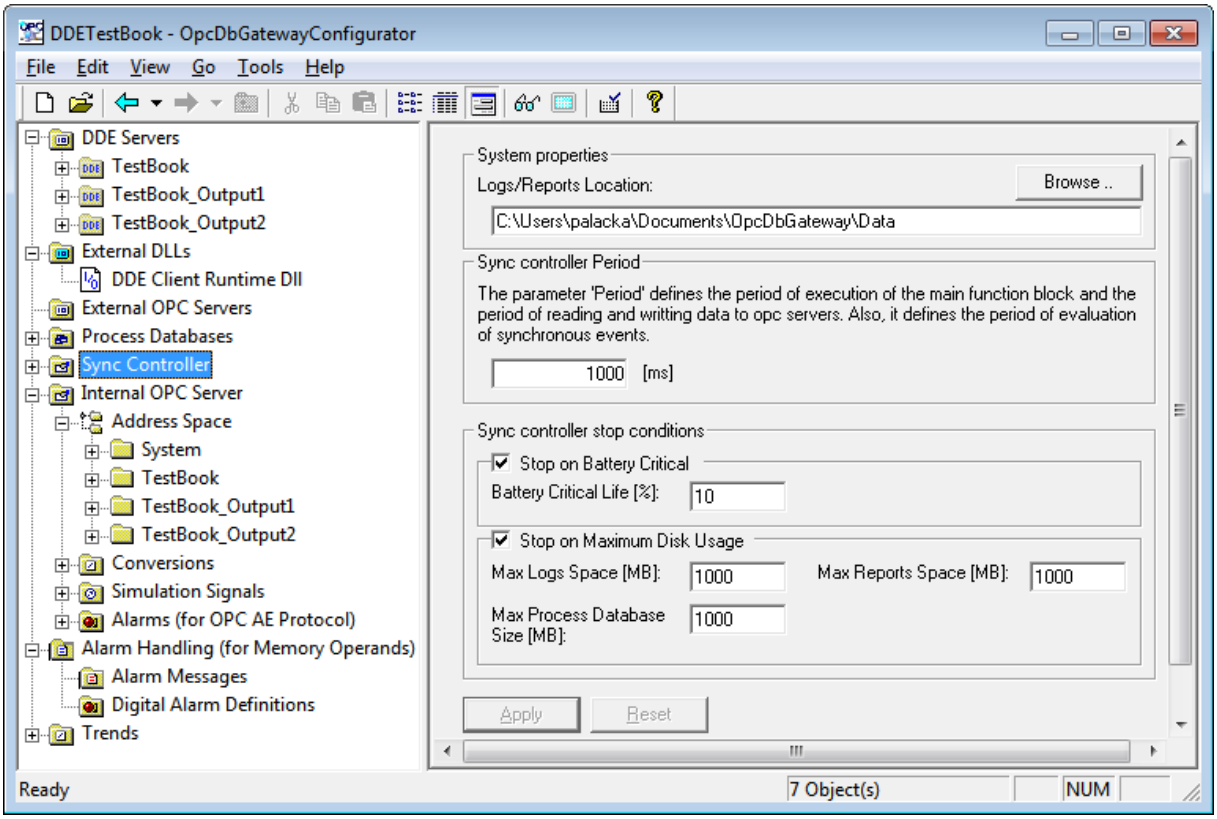


Figure: *The OpcDbGateway configuration database*

6.1 Configurator - User Interface

OpcDbGatway Configurator

The OpcDbGateway runtime doesn't have any user interface. The configuration data of the server is created by the configurator module whose user interface is described in this chapter.
The user interface of the configurator is similar to Microsoft Explorer. It provides numerous functions to make the configuration easier, such as Copy and Paste, Drag and Drop, or context based popup menus.

- File menuViews layout
- Edit menu
- View menu
- Go menu
- Tools menu

6.1.1 Views layout

Views layout is divided into three base view sectors:

- Tree view
- List view
- Dialog view

And three included view sectors that you can recall through the main menu:

- Monitor view
- Checker / Find
- Graphic project viewer

database*re:* *Views layout*

Remark: *from the ver 5, the Output view with Log file viewer, alarm viewer and dtabase tables viewer are shown in standalone window.*

Tree view
The tree view is used for exploring the structure of the configuration database

List view
The list view displays the contents of the item selected in the tree view.

Dialog view
The dialog view is used for editing parameters of the item selected in the tree view.

Monitor view
The monitor view is used for start OpcDbGateway runtime and monitoring the address space using built in OPC DA client.

Checker / Find view enables

- checking of created or uploaded configuration. After choosing of line with an error message using double click related dialog box will be shown. The main purpose of this feature is check data in configuration database. User can start checking through the menu item **Tools⇨Check configuration** or with toolbar button with the same name. If the checking finished without errors the database should be all right. But the checking finished with errors, user can through the double click on error line in Checker view show the view that consists the error or errors.

- view different objects in configuration according to the name using Find dialog

Graphic project viewer
There is a graphical presentation of the whole project. By double click on a graphic object related dialog box will be opened for editing.

6.1.1.1 Checker view

The main purpose of this feature is check data in configuration database.
User can start checking through the menu item **Tools⇨Check configuration** or with toolbar button with the same name.

If the checking finished without errors the database should be all right. But the checking finished with errors, user can through the double click on error line in Checker view show the view that consists the error or errors.

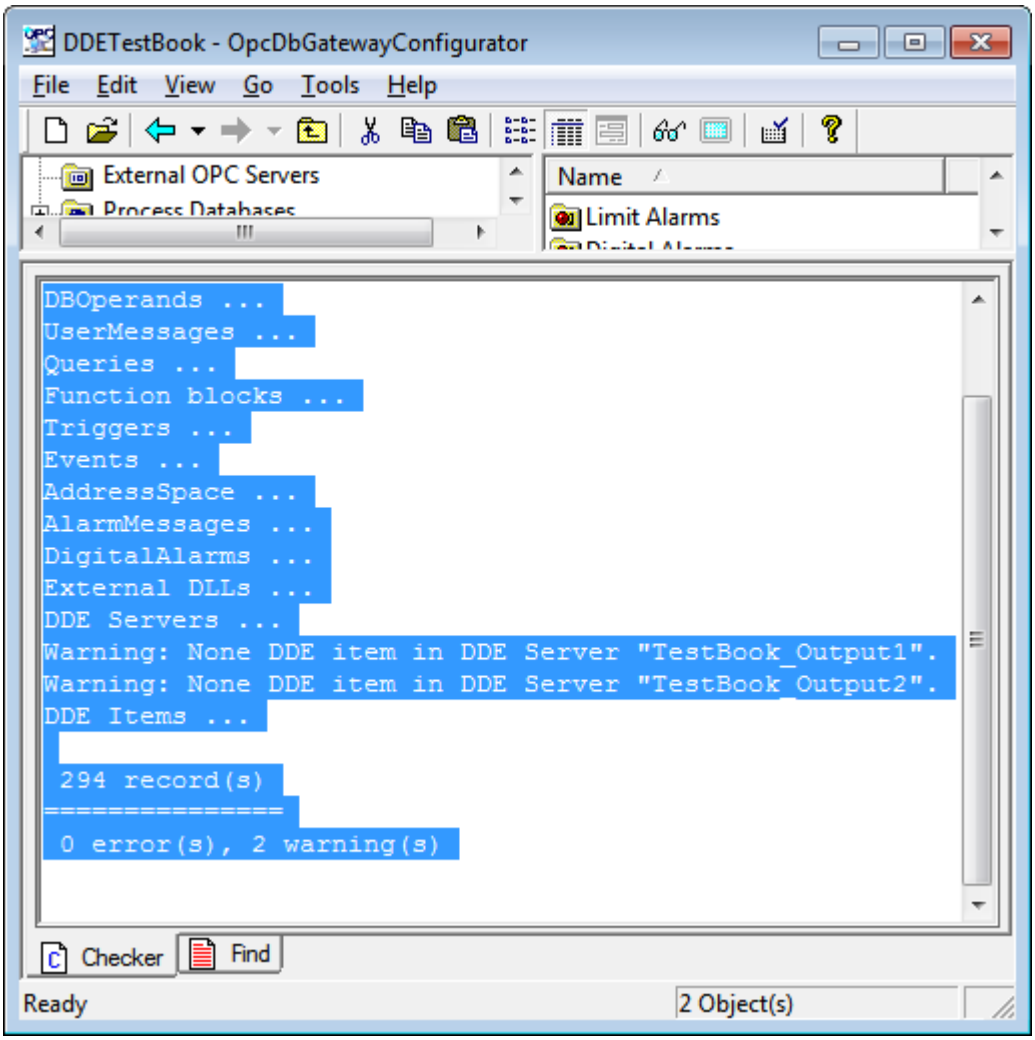


Figure: *Checker view*

This error is saying about that the column 0 is missing and the ID for this table doesn't exists.

6.1.1.2 Find

This view is recalled from main menu **Edit⇨Find**. More description about finding you can see here.

After double click on founded item you can show item view.

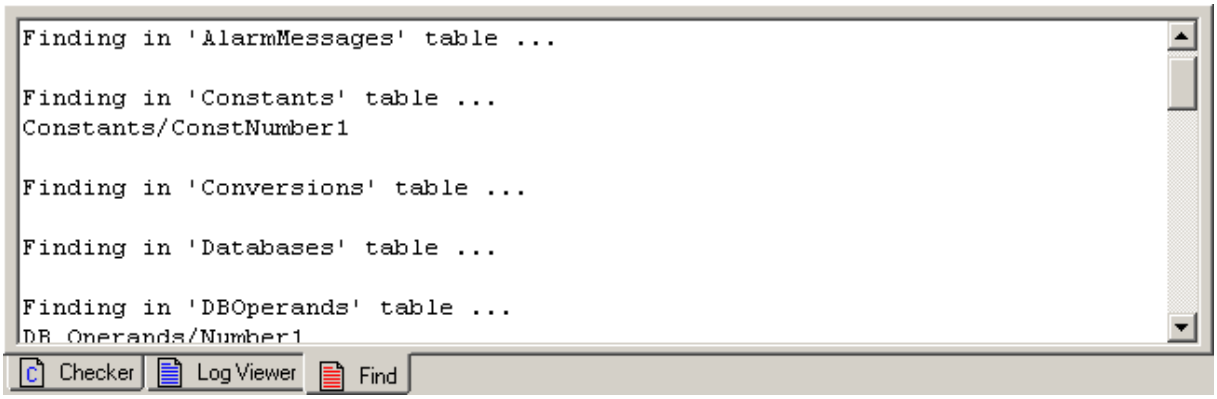


Figure 61: *Find view*

6.1.2 File menu

There are several commands in the menu File that work with the configuration database file.

New, Open, Save As
These commands create new, open existing or save changed configuration database

Connection properties...
General information about the configuration database (Data source name, provider, version ....).

Export CSV
Commands in this submenu allow to export different parts of the configuration into a standard CSV

(comma separated variables) file.

Import CSV
Commands in this submenu allow to import different parts of the configuration from a standard CSV (comma separated variable) file.

XML Export
Commands in this submenu allow to export different parts of the configuration into a standard XML file.

XML Export Schema
Commands in this submenu allow to export schema into a standard XML file.

XML Import
Commands in this submenu allow to import different parts of the configuration from a standard XML file.

XML Validate
Commands in this submenu allow to validate configuration stored in a standard XML file.

Make Active...
This command makes active the current configuration.

Exit
This command will close the configurator.

6.1.2.1 Make active...

The active database
The active database is the database that the runtime part of the server will use when started. The active database may (but does not have to) be the same as the database currently edited in the configurator. You can work on any database inside the configurator, while the runtime part has its own active database.

Make Active command
The Make Active command sets the currently edited database as the active database. Next time the server runtime part starts, it will use this active database for all its operations. This menu item is disabled when the currently edited database is already active.
Before the active database is actually set, the program asks you to confirm the setting.

The dialog box that is invoked looks like this:

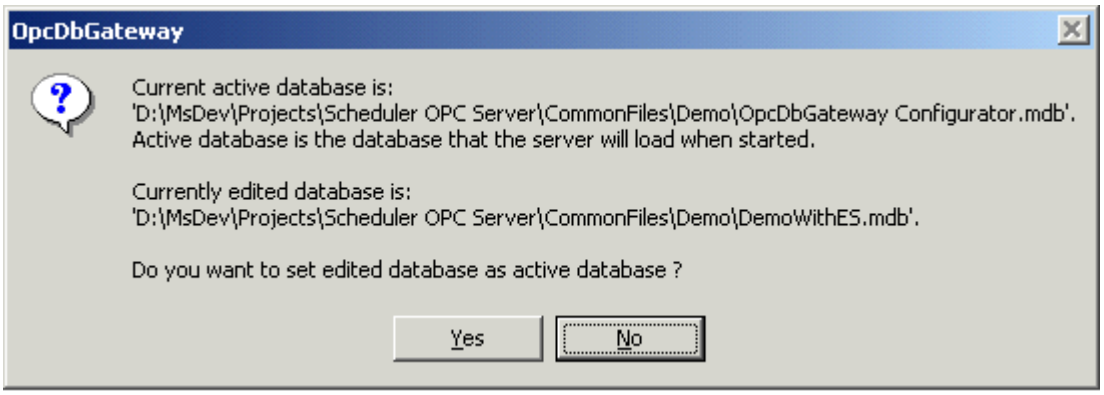


Figure 55: *Current active database /dialog/*

6.1.3 Edit menu

Edit menu contains many standard commands, such as Rename, Delete, Cut, Copy, Paste, Select All and Invert Selection.
The configurator supports standard drag-and-drop behavior in the list and tree views.

Other edit menu items are following:
- New
- Find

New
This submenu contains commands to add new objects into an active branch in the tree. The actual contents of this submenu is based on the type of the branch you have selected.

Multiply ...
This command multiplies the item selected in the tree view or list view.

Find
Through this menu item is the possibility to make finding in configuration database. After click on this menu item cofigurator shown this dialog:

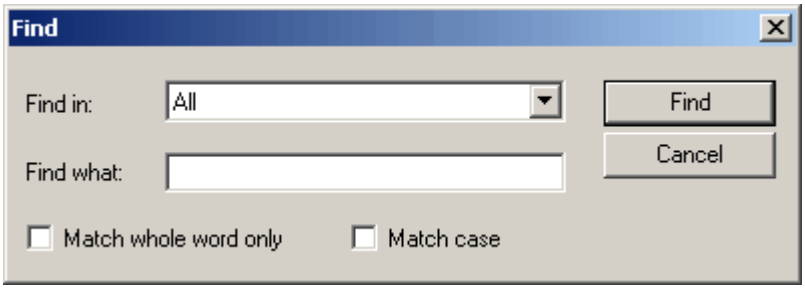


Figure: *Find /dialog/*

The combo item **Find** consists what do you want to find:
- All - finding in whole configurator database.
- Process database - find a process database in all tables where is used.
- Process table - find a process table in all tables where is used.
- Memory operand - find a memory operand in all tables where is used.
- DB operand - find a DB operand in all tables where is used.
- Constant - find a constant in all tables where is used.
- Querie - find a querie in all tables where is used.
- User message - find a user message in all tables where is used.
- Function block - find a function block in all tables where is used.
- Trigger - find a trigger in all tables where is used.
- Conversion - find a conversion in all tables where is used.
- Simulation signal - find a simulation signal in all tables where is used.
- Alarm message - find a alarm message in all tables where is used.
- Alarm definitions MOP - find a alarm definitions MOP in all tables where is used.

In the edit box **Find what** write the text what do you want to find. Below are the standard settings.
After click on **Find** button configurator open Output view and in the Find tab shown all founded items.

6.1.3.1 Multiply ...

Multiply... command within Edit menu belongs to the set of non-standard commands. Its purpose is straightforward. Selecting this command will launch Multiply Item Dialog.

The actual look and contents of the dialog is server-dependent. The simplest example of this dialog is on the picture below:

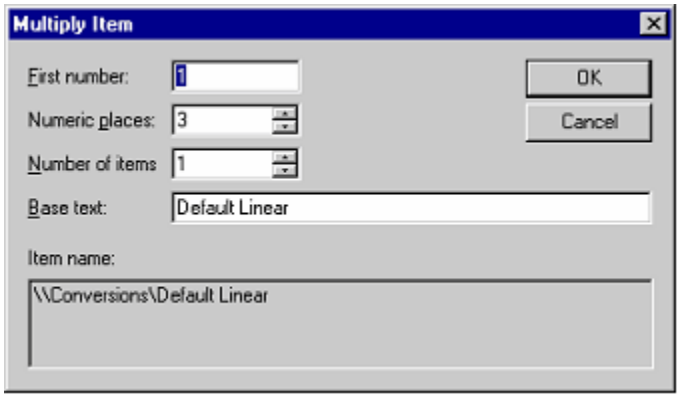


Figure 57: *Multiply Item /dialog/*

First Number
The number appended to the first generated item name. Following items will be numbered consecutively.

Numeric places
The count of digits of the number appended to the item name

Number of items
The count of items to be generated

Base Text
String that the generated item names will start with

Item name
Path to data item in the left tree view pane

6.1.4 View menu

View menu allows users to show/hide tool bars such as Standard Buttons and Data Manipulation Buttons, or a status bar.
The items in the List view pane could be displayed in one of four following modes:

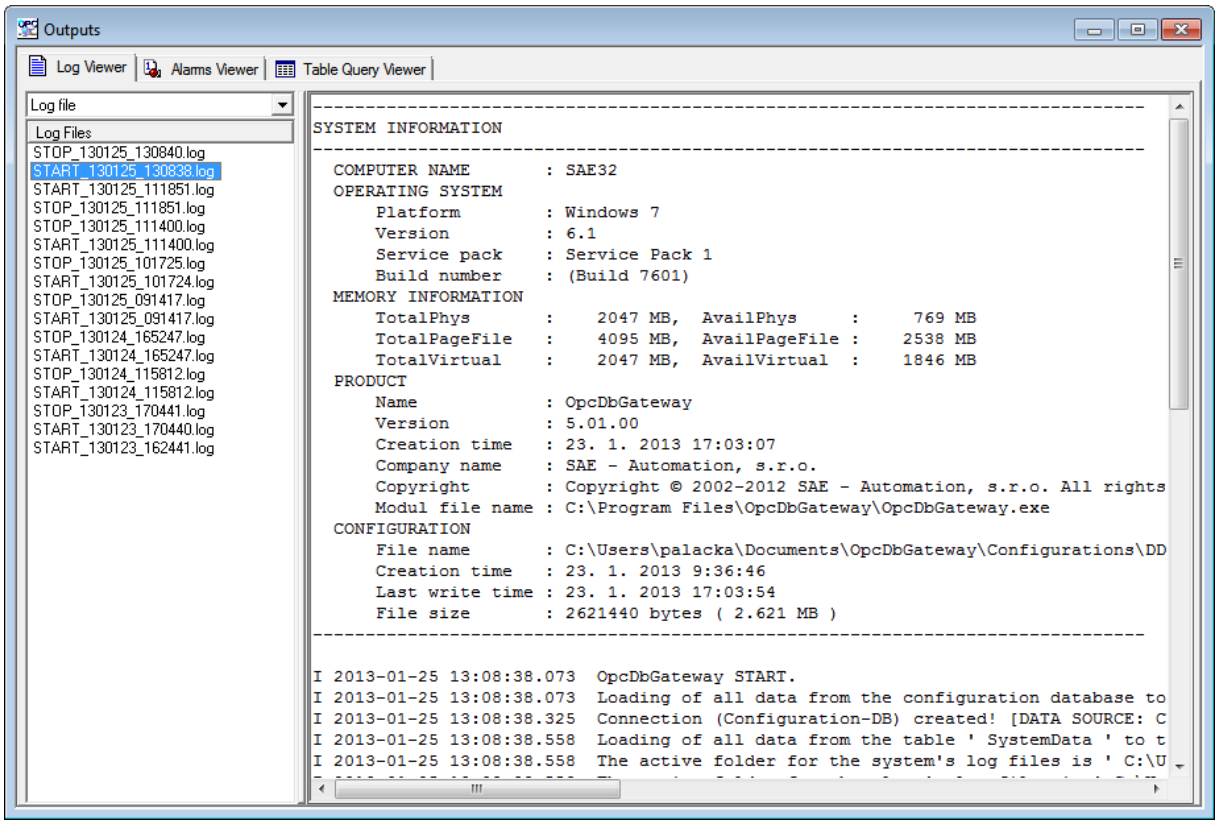
- large Icons
- small Icons
- list
- details.

View panes could be shown or hidden selecting the appropriate command from the menu or pressing F11 for Dialog view.

Choose **Show/hide** columns to adjust report view in the right pane of the configurator. Checking appropriate columns within Sort can influence the order of data items in the right pane by submenu commands.

Through the menu item Output view or on pressing Ctrl+F12 is able to show or hide the Output viewOutput view.

6.1.4.1 Output view



Output view is tabbed view and consists from three tabs:

- Log view
- Alarm Viewer
- Table Query View

6.1.4.1.1 Log view

Viewing the log files and the alarm log files
Log view serves for viewing the log files and the alarm log files. User can choose it in the combo box. Path for these files are configurable in view **General Settings/Settings** in **Log/Reports Location**.

List control display all log files in specified directory. If the server is running and creating new log files

list control automatically display new created log files.

Rich edit shown the contents of actually selected log file. The content is still refreshed. Error line is in red color.
With the right mouse button on rich edit control can user validate the checksum.

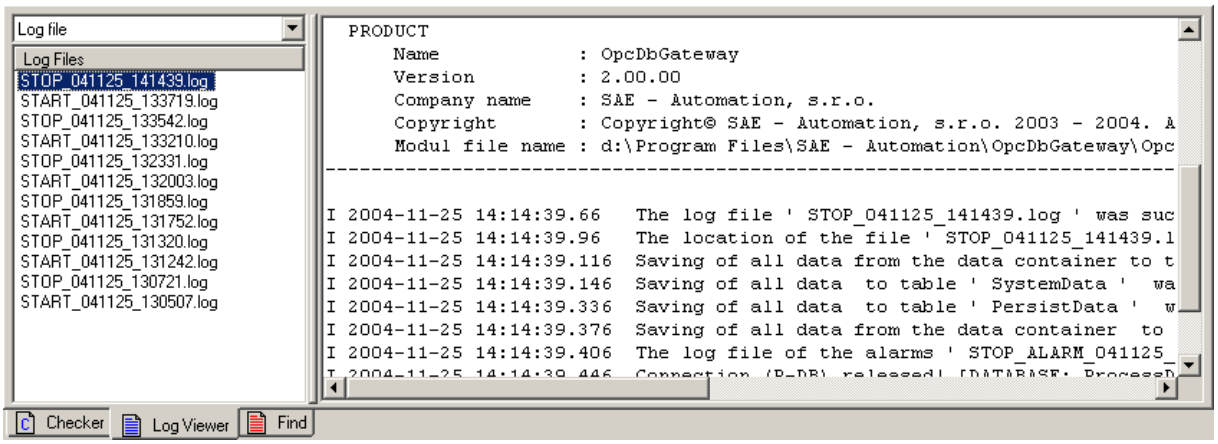


Figure 60: *Log view*

6.1.4.1.2 Alarm Viewer

Alarm viewer enables to watch, quit (acknowledge) and comment alarm status of different alarm sources of the proprietary alarm system of the OpcDbGateway.

The table in the viewer has maximally as many rows as configured alarm sources.

Viewer can be **connected** or **disconnected** from alarm process database using buttons "Connect" / "Disconnect".

Viewing of the alarm sources can be filtered according to:

- Status of alarm sources (**Inactive** – alarm conditions are not fulfilled, **Come** – alarm was activated, **Gone** – alarm conditions are not more valid, **Acknowledged** - operating personnel confirmed seeing of alarm appearance)
- **Group** – Nr. of the group configured for a alarm source
- **Priority** – a severity configured for a alarm source
- **Date** – date of alarm occurrence

Alarm sources can be deleted from displaying in alarm viewer by choosing the alarm source (with mouse or cursor and enter keys) using "Delete" button. Using button "Show undeleted", they can be displayed again.

An alarm status can be commented using after choosing the alarm source (with mouse or cursor and enter keys) in the edit box "Commentary". The comment is written together with alarm source name to the alarm history table.
Alarm in status" Come" can be acknowledged after choosing the alarm source (with mouse or cursor and enter keys) using button "Acknowledge".
Every change in status of the alarm source is written to the alarm log file and to AlarmStatusHistory of the process database. This table can be viewed in configurator using Table query viewer. It can be seen also in alarm log file using Log Viewer

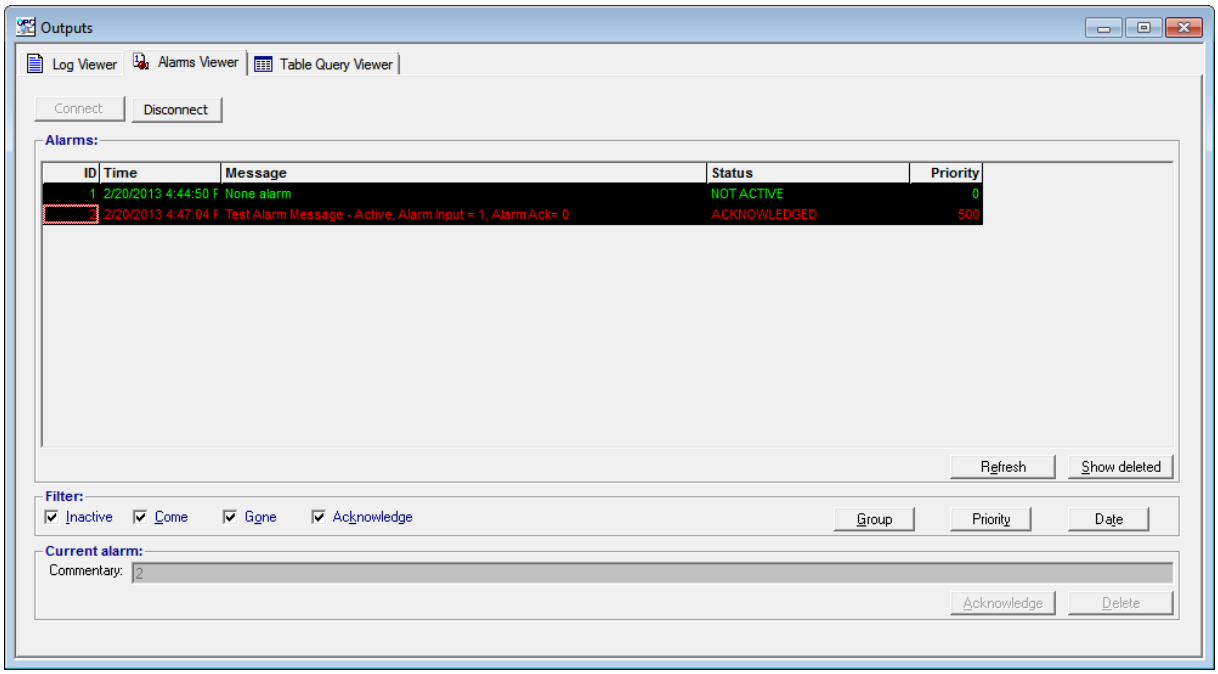


Figure: Alarm Viewer in in OpcDbGateway configurator

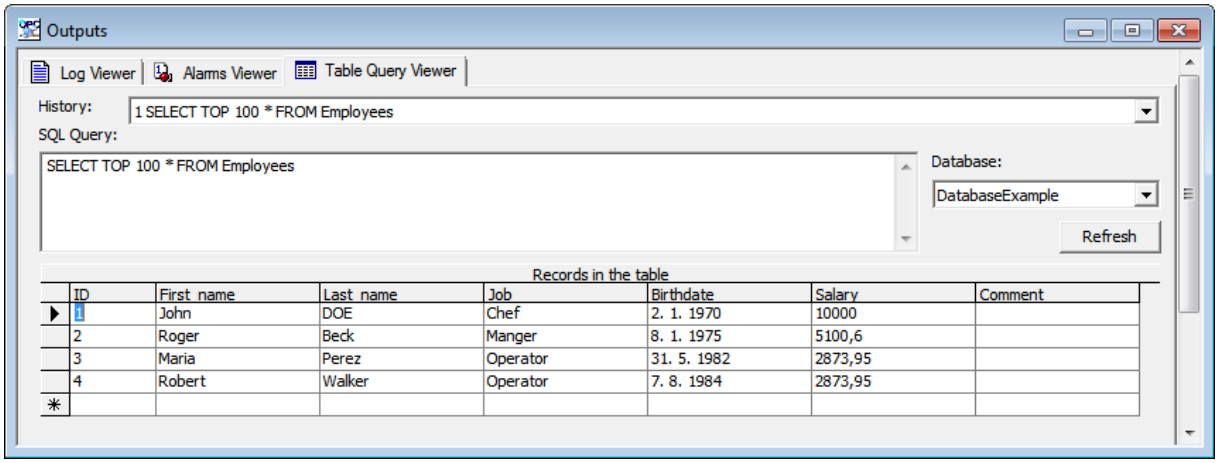
Related articles

Alarm system functionalitty
Alarms - configuring

6.1.4.1.3 Table Query Viewer

It enables to create view from whatever table from database configured in a configuration using a SELECT SQL query. It is sugested to reduce number of displaed records eg. such a way as shown in the figure.

To see a content of the database view choose a configured database using the *Database* select box. Create a new query in the *SQL query* edit box or chose a query from the *History* select box.



.

To see

6.1.4.1.4 Graphic project Viewer and Editor

This viewer presents project structure by graphical schema and gives a new look to the project structure. This feature provides the user with a possibility to have a graphical overview of the project for the purpose of configuration check. In this viewer, the user can find the representation of all parallel processes configured in the project. This is made with help of ladder diagrams, each of the ladder parts represents one parallel process. The user can see the difference between the synchronous and asynchronous processes.

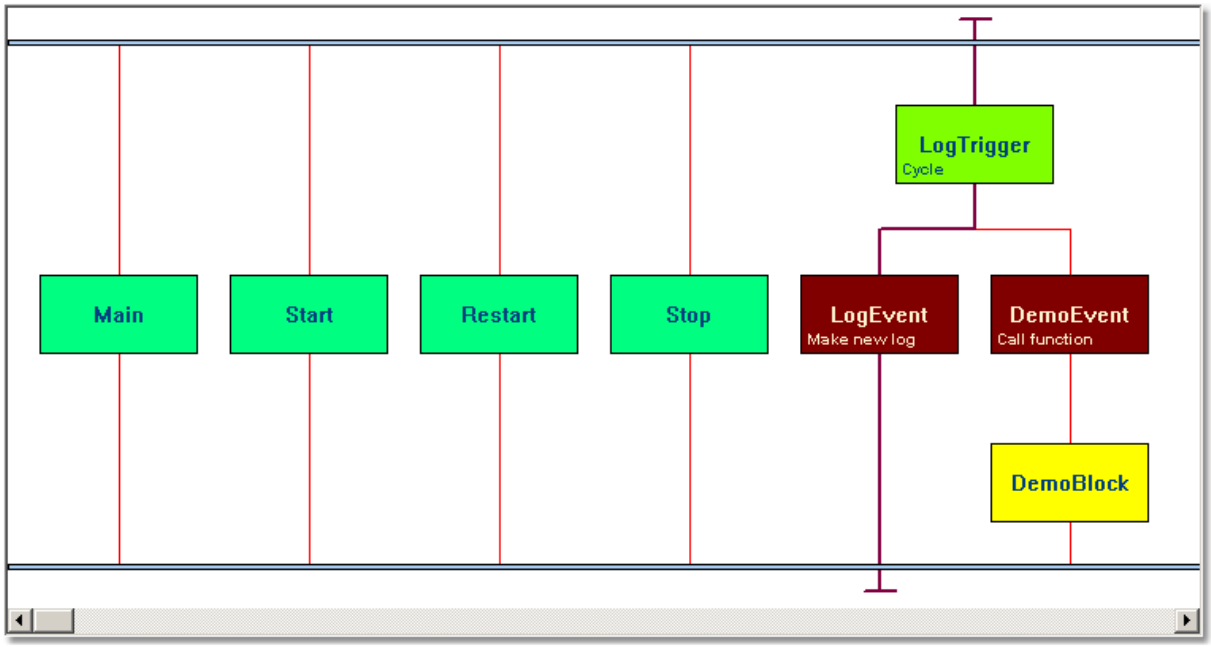


Figure 1: *The Graphic project viewer*

Related articles

OLE for Process Control (OPC)
OPC in OpcDbGateway

6.1.5 Go menu

To view available records in any selected window, use a command from a list that appears in the Go menu, or use the Arrow, Page Up/Down, Home, End keys for moving in the tree view in combination with Alt key.
Up One Level command moves the cursor one level closer to the root of the tree.
Next Pane or Previous Pane allows users to loop through panes, if more than one is displayed.

6.1.6 Tools menu

Options command

Options dialog within contains the following parameters. User can invoke this dialog after click on menu **Tools⇨Options**.

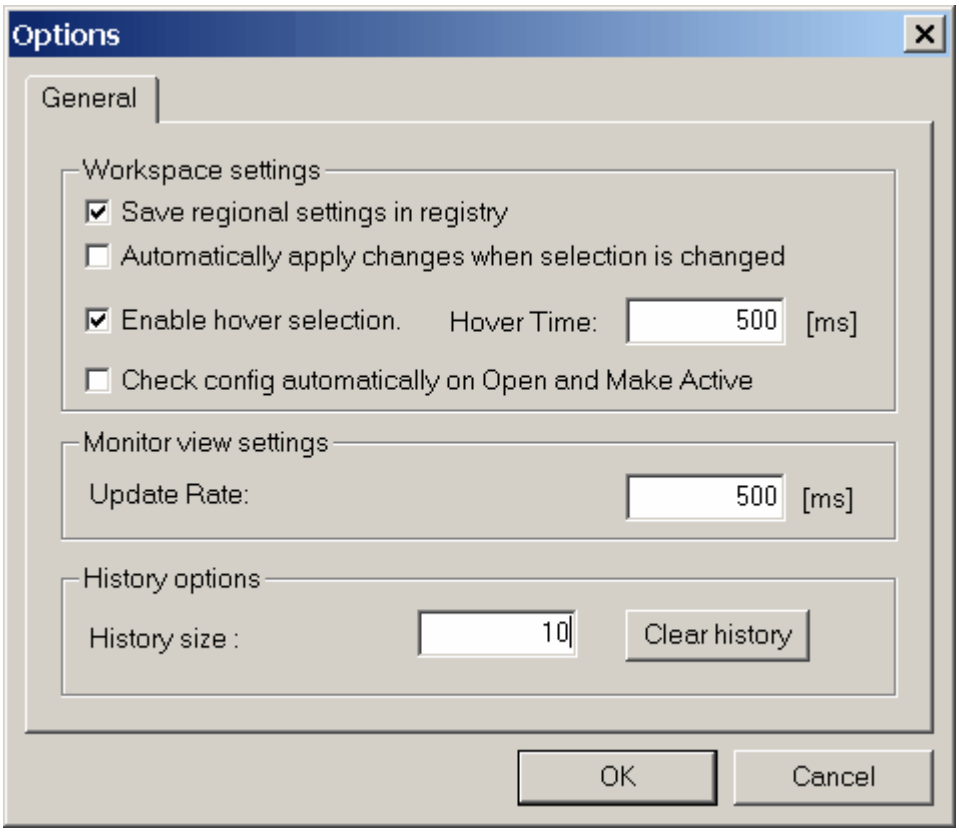


Figure 62: *Options /dialog/*

Worspace settings

Save regional settings in registry
Enables/disables the configurator to save active language chosen from the Select Language dialog into registry and set it up during initialization.

Automatically apply changes when selection is changed
If checked, the configurator itself tries to update the edited data without requiring clicking on **Apply** button, and displaying the confirmation dialog.

Enable hover selection, Hover time
Allow configurator support automatic selection of the item above which the mouse is hovering. Enter Hover time parameter in milliseconds

Update rate
Specifies the update frequency of **Monitor View** items.

Check config automatically on Open and Make Active
Allow configurator to check database configuration on **Open** and **Make Active** events.

Monitor view settings

Update Rate
Specify how often monitor view refresh its contents.

History options

History size
The size of history for toolbar buttons back and forwards. It means how many views can remember for

back and forward actions.

Clear history
This button clear all remembered views from memory.

6.2 External DLL

More details about External Dlls you can find in the External DLL - usage topic.

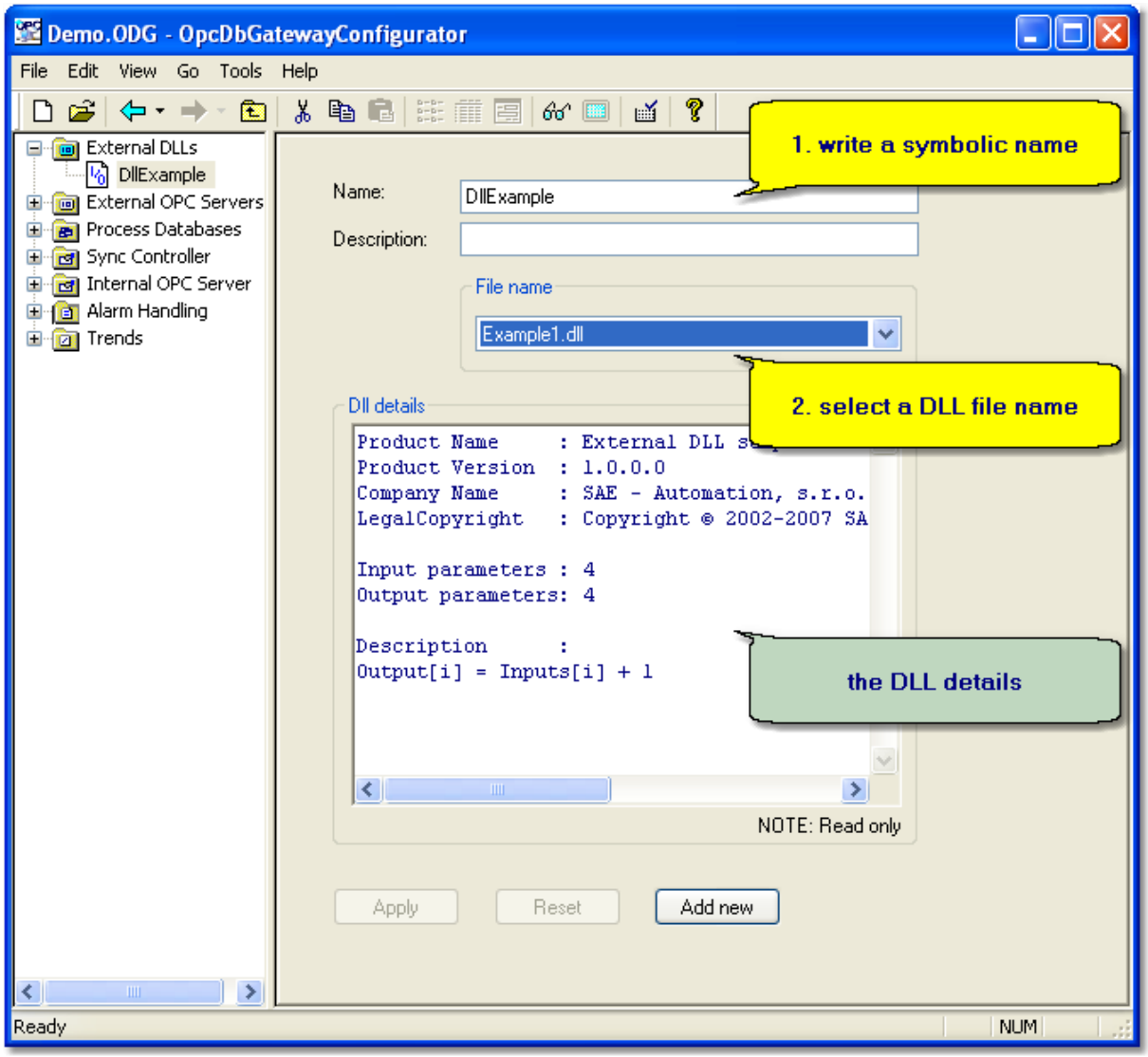


Figure: *External DLLs*

Name	External DLL symbolic name.
Description	Brief external DLL description.
File name	The DLL file name.
Dll details	Brief description obtained directly from DLL library.

Table: *External DLLs parameters*

How to add a new external DLL?

Click on the folder **External DLLs** with the right mouse button and from the context menu choose the command **New⇨External DLLs**.

For more details see topic External DLL module or Operation CALL DLL.

Related articles

External DLL module, Operation CALL DLL, How to call en external DLL from OpcDbGateway?

6.3 External OPC servers

OPC client of the OpcDbGateway runtime can be connect with one or more external OPC servers. For every external OPC server one ore more OPC groups can be configured.

The OpcDbGateway runtime can be connected either to local or remote OPC servers. The remote OPC servers are installed on another computer.

The data from an OPC server can be read either from its CACHE or directly from the DEVICE. CACHE is a kind of memory of the external server where the current state of all opc items is stored. On the contrary, the DEVICE represents the source from which the external opc server retrieves the values - e.g. physical input or output. It is obvious that reading from the CACHE must be faster than from DEVICE.

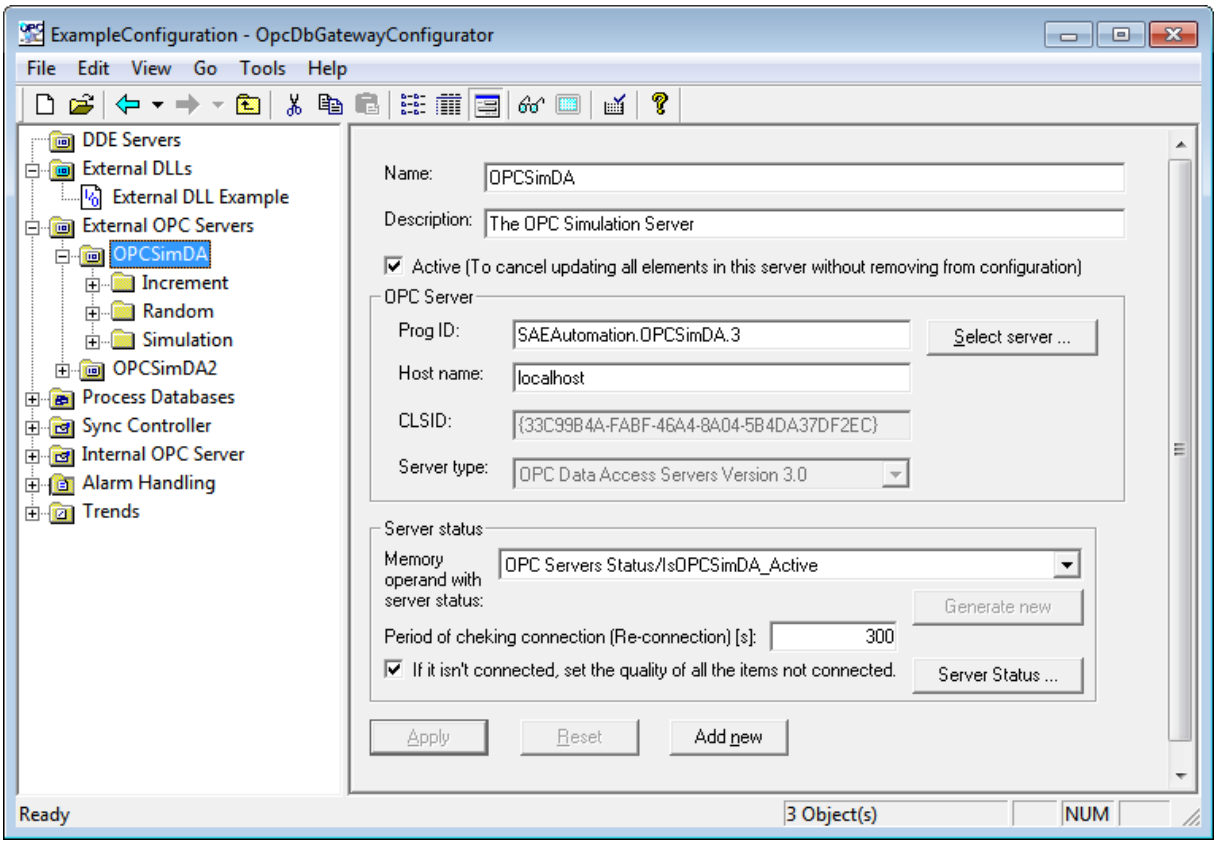


Figure: *Tree view and dialog box to configure **e**xternal OPC servers*

Name	Name of the external OPC server which will be shown in configuration and can be edited
Description	Shown only in configuration
Active	If not checked all elements from server will not be uopuptadet at runtime
ProgID	ProgID of an external OPC Server choosen by browsing or edited
Select server	Browsing of available external OPC serversDescription of OPC Server
CLSID	CLSID related to chosen ProgID
Host	Host where external OPC server resides
Server type	OPC interface of the server DA2.x/DA3.x
Check connection & Re-connect of OPC Server	The parameter Period defines the period of checking of the connection to the external OPC Server and the period when the OpcDbGateway atempts to establish the lost connection again (re-connection).
Asynchronous update	With this option you can specify whether data will be readed automatically or when used. If this option is disabled you MUST use any OPC item in any function block. It will read opc items from the DEVICE of the opc server.
Memory operand with server stauts	Memory operand for information if OPC server is connected
Period of checking connection	In case that connection has been disrupted client will try reconnect using this period
If it isn't connected set quality of all items not connected	If not checked the last quality will be used
Server status	Tries to connect to the server and shows status as in the picture bellow

Table: *External OPC server parameters*

How to add a new opc server?

Click on the folder **External OPC Servers** with the right mouse button and from the context menu choose the command **New⇨ExternalServers**.
In the dialog-like view click on the button **Select server** and from the list of all local opc servers choose one.

Related articles

Select an external OPC server
OPC items mapping
Server Status
OLE for Process Control (OPC)
Memory operands
Address space

6.3.1 Browse remote OPC servers

The OpcDbGateway runtime can connect not only to a local opc server. It can also communicate with remote opc servers. You just browse the OPC server wich you want from the list, it doesn't matter if you want local or remote OPC server.

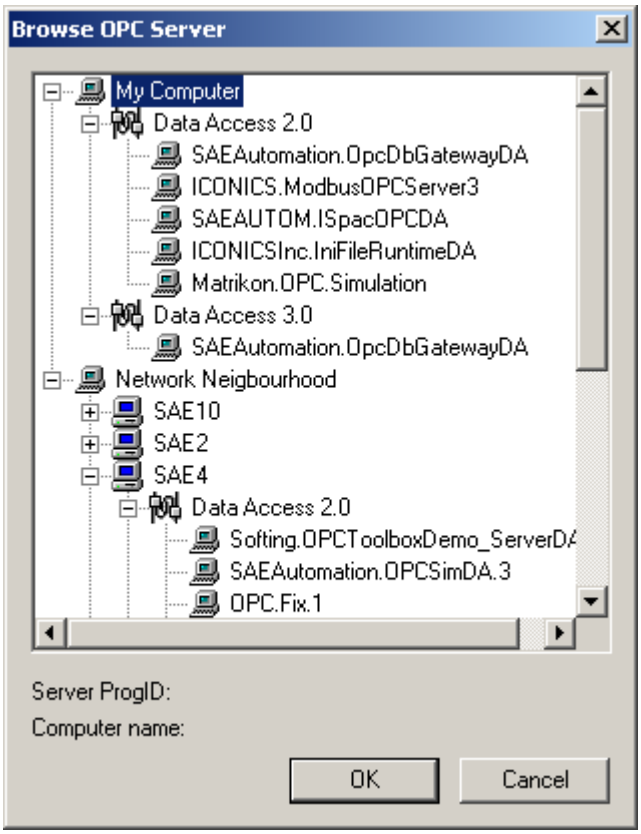


Figure: *Select server*

Related articles

External OPC servers
OPC items mapping
Server Status
OLE for Process Control (OPC)
Memory operands
Address space

6.3.2 Create OPC group

One or more OPC groups can be created to access data on external OPC server. It enables to define specific conditions for accessing of different groups of data items.

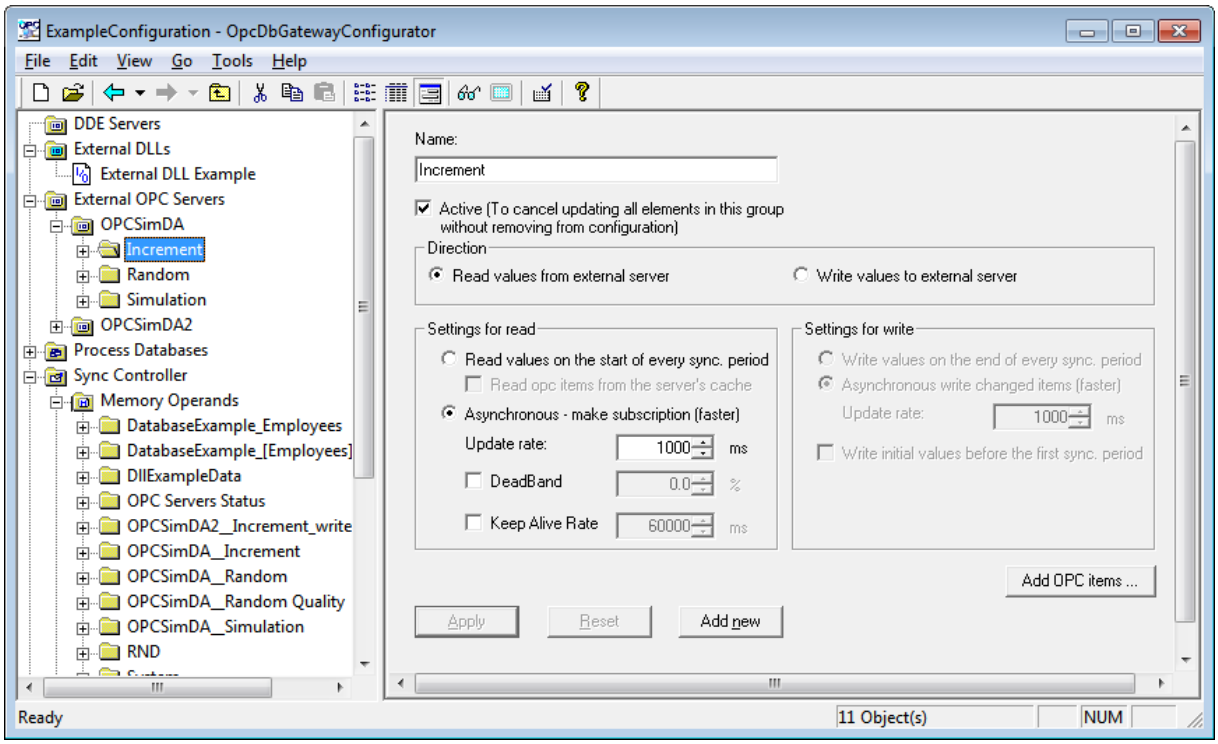


Figure: *Configuring of OPC group parameters*

Name	OPC Group name
Active	To enable or disable updating of OPC group itemsat runtime
Direction	
Read values from external server	OPC Group will be configured for reading
Write values to external server	OPC Group will be configured for reading
Settings for reading	
Read values on the start of every sync. period	Values will be read also in case of no change
Read items from the OPC server's cach	Values will be read from cache instead of device if checked
Asynchronous - make subscription - faster	Cleint is notified by server in case of change
Update rate	The smallest period for notifications required by client (server can offer nearest higher own period in case that it is not able satisfy the client
Dead band	minimal change which should cause notification from server to client
Keep alive Rate	period for notifications from server in case of no change
Settings for writing	
Write values on end of every sync. period	Values will be written also in case of no change
Asynchronous write changed items (faster)	Written as fast as possible (limited by a client update rate)
Winitial valuesrite values before the first sync. period	self describing
Add OPC items	Start wizzard to browse, add and map OPC items from external server to memory operands abd OPC items of the internal OPC server

Table: *OPC Group configuring*

6.3.2.1 Server Status

Each OPC Server provides information abvout its status. This status inludes various important properties:

- OPC Version,
- Vendor Info,
- product Version,
- server State,
- server Start time,
- server Last Update Time,
- server Current Time,
- Bandwidth

- Groups Count.

How to show an OPC Server Status?

Click on the button **Server Staus ...** in the dialog-like view of an external OPC server.

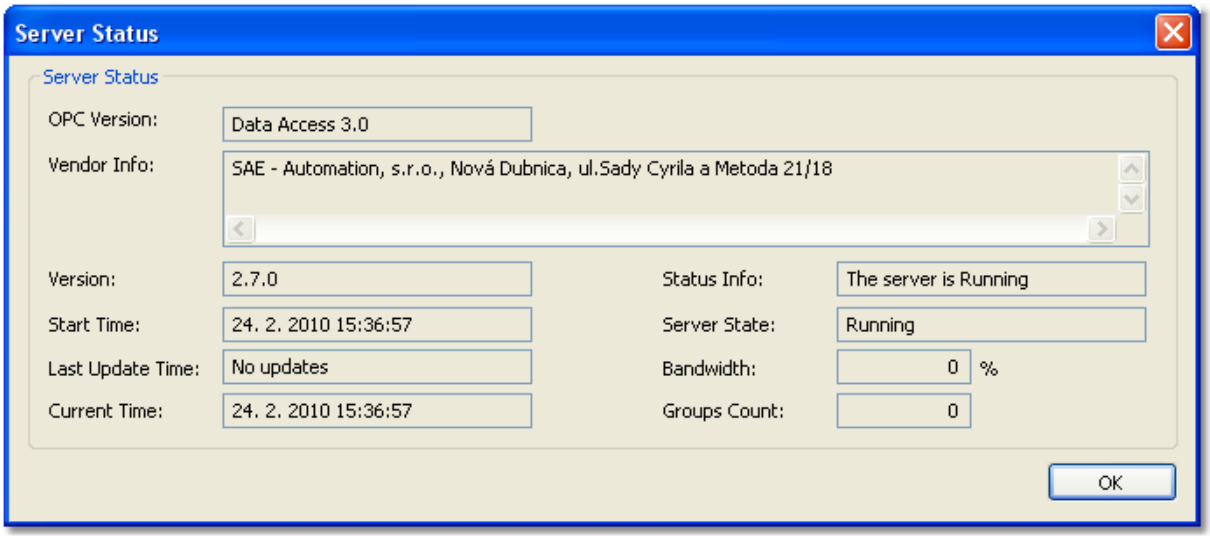


Figure: *Server Status dialog*

Related articles

External OPC servers
Select an external OPC server
OPC items mapping
OLE for Process Control (OPC)
Memory operands
Address space

6.3.3 Add OPC Items

OPC items from external OPC servers have to be chosen by browsing on external OPC server and mapped to memory operands that are used within configured or programmed functionality of the OpcDbGateway. They can be also exposed for external applications trough OPC items in address space of the internal OPC server.

There are 2 steps in this process:

- chosing of OPC items for mapping
- chosing of element settings for mapping

Related articles

External OPC servers
Select an external OPC server
Server Status
OLE for Process Control (OPC)
Memory operands
Address space

6.3.3.1 Map OPC items

OPC items for mapping can be chosen either one by one or the by whole directories including subdirectories in tree view (see **Figure**: *Choosing of OPC items for mapping*).

To chose one item, click on the item with left mouse button and click button **Add One**. Item will be added to the list. This process can be repeated till all required OPC items will be in the list.

To choose all items from a directory with all subdirectories choose a directory in the tree view and click **Add Tree** button.

When all variables have been chosen click on **Next** button

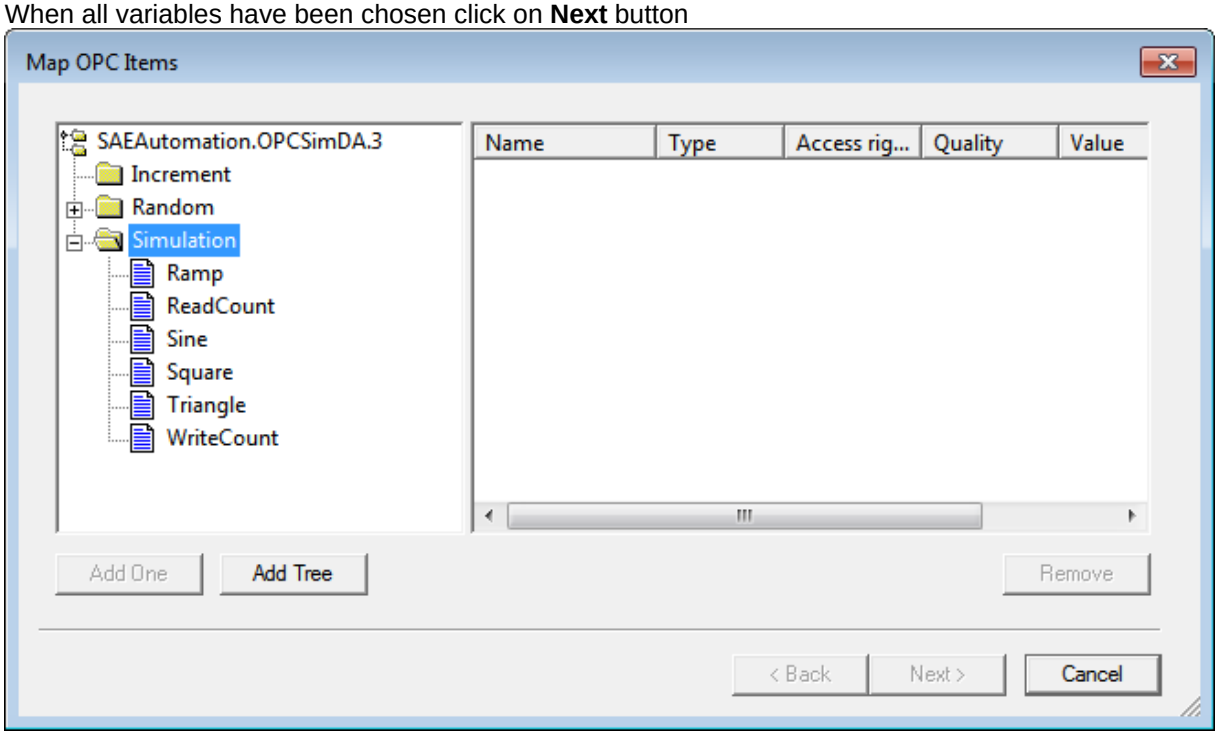


Figure: *Choosing of OPC items for mapping*

Element settings for mapping will be defined in next step.

6.3.3.2 Element settings

Element settings for mapping of OPC items from external OPC servers to memory operands and OPC items of the internal OPC server can be chosen from dialog box in the figure bellow.

Either default setting or own settings can be chosen. In the default settings, the first free memory operands area with enough space for placement of all chosen OPC items is suggested.

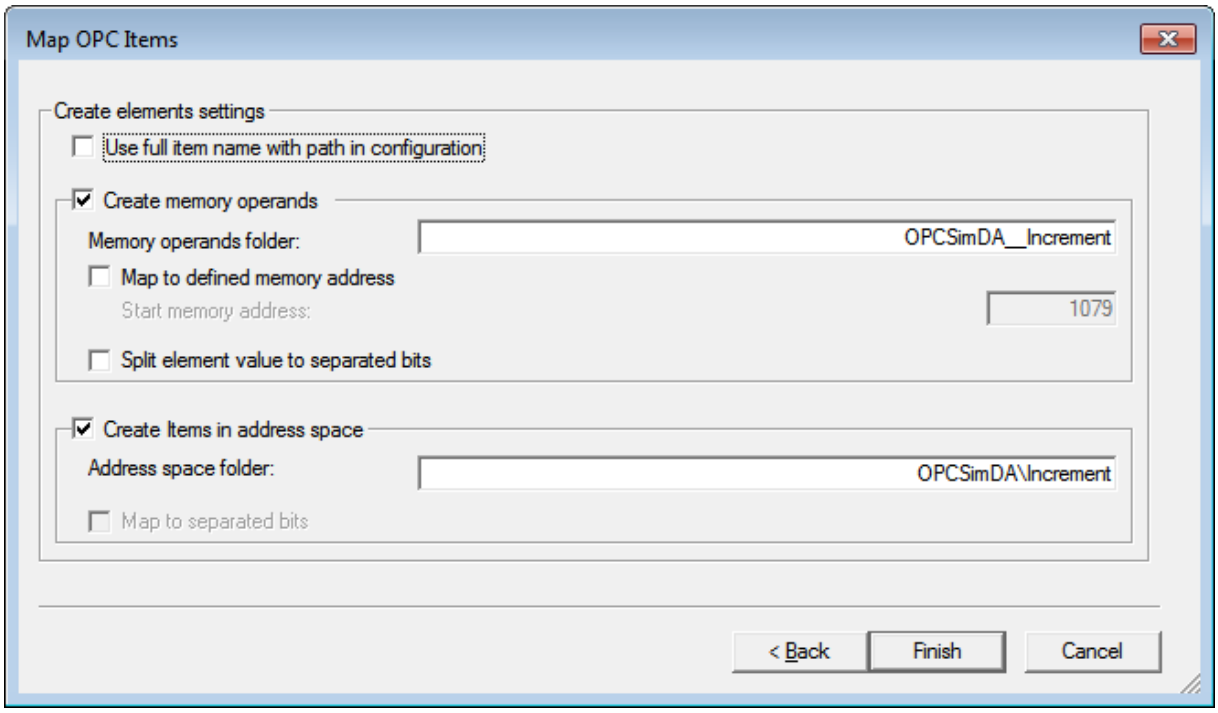


Figure: *Element settings for mapping of OPC items from external OPC server*

Using check boxes "**Create memory operands**" and "**Create items in address space**", it can be chosen which mappings have to be done.

Value from external OPC server can be interpreted as a bit set in case that check boxes "**Split element values to separated bits**" and/or "**Map to separated bits**" will be checked. In such case every bit of the OPC item from external OPC server will be mapped to distinct memory operand and/or OPC item of the internal OPC server.

Within edit boxes "**Memory operands folder**" and "**Address space folder**" folder names related to the folder name in address space of the external OPC server are suggested. They can be changed if required.

6.4 Process databases

The OpcDbGateway runtime can access one or more **databases** at the same time. The server supports many types of providers.
Access to databases is defined using **connection string (**see in figure bellow**)**.

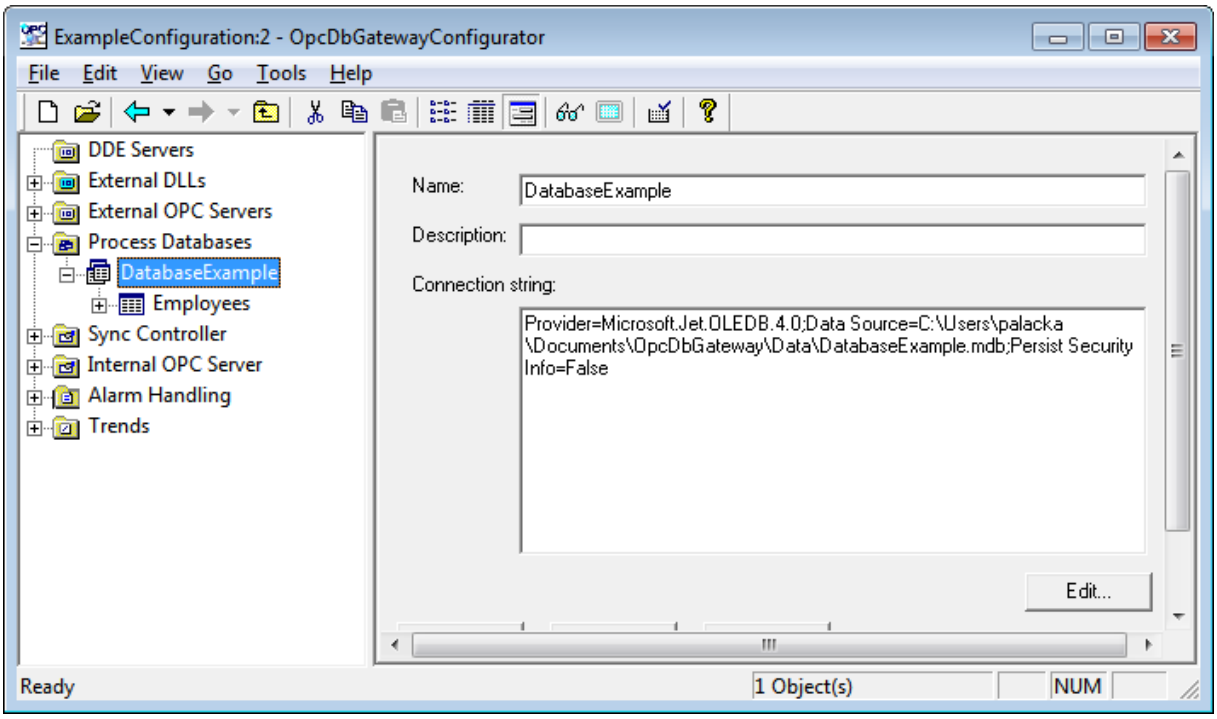


Figure: *Process databases*

How to add a new database?

Click with the right mouse button on the folder **Process databases**. Select the command **New⇨Databases** from the context menu. A new database item will be created. The parameters necessary for definition of the new database are following:

Name	It is only for usage within configuration
Description	It is only for usage within configuration
Connection string	Can be edited directly in edit box or created steb by step using the connection wizard started by **Edit** button

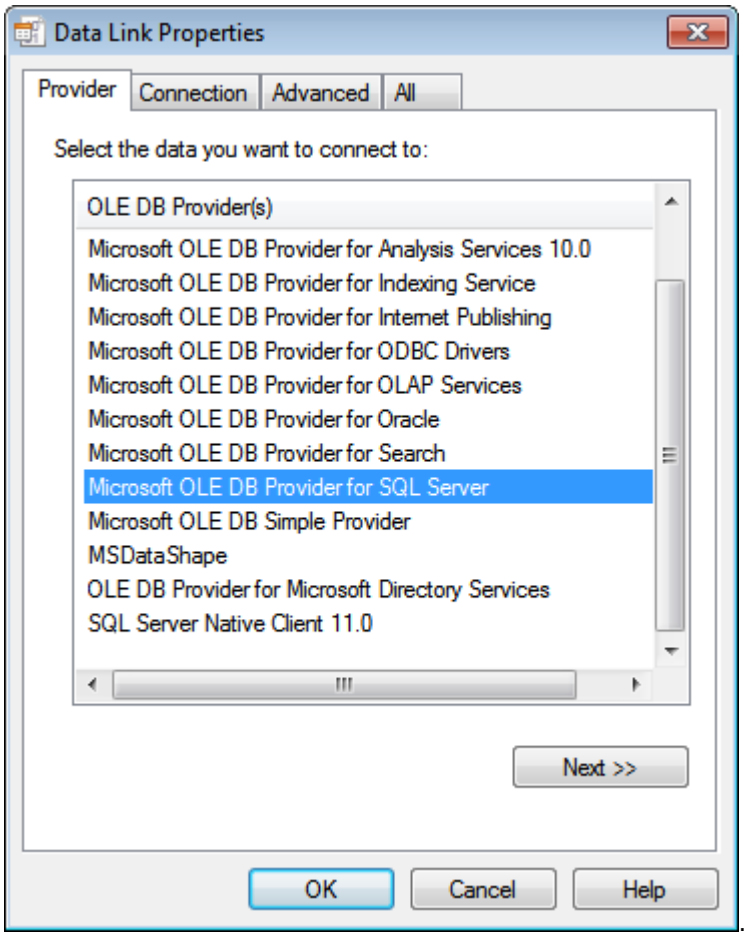
Table - ***Process database parameters***

A wizard will guide you through the process of configuring of the driver. You will have to define the data source name (DSN) and other parameters depending on the choosed driver.

If a new configuration is created from **Menu->File->New** a new process database - ProcessDB with predefined tables AlarmStatus, AlarmStatusHistory, GeneratedReports and NonPrintedReports is created as well.
You need not use it. If you like to use another database you can delete it, change the connection string and map the created tables (if you need them) to the new database.

6.4.1 Connection wizard

It enables creating of connectiong string to a database. It is initiated using "Edit" button in Process database dialog box or from Database table mapping wizard



6.4.2 Process tables

To be able to work with data from database tables using configured or programmed functionality of the OpcDbGateway, it is necessary to provide mapping of used database tables from a database to the configuration or in opposite direction from configuration to database. It is very important to have identical table definitions on both sides. It can be provided by following tools.

In case that you have tables ready in database you can call the wizard from Tools->Wizards->Create Mapping to database table - figure bellow.

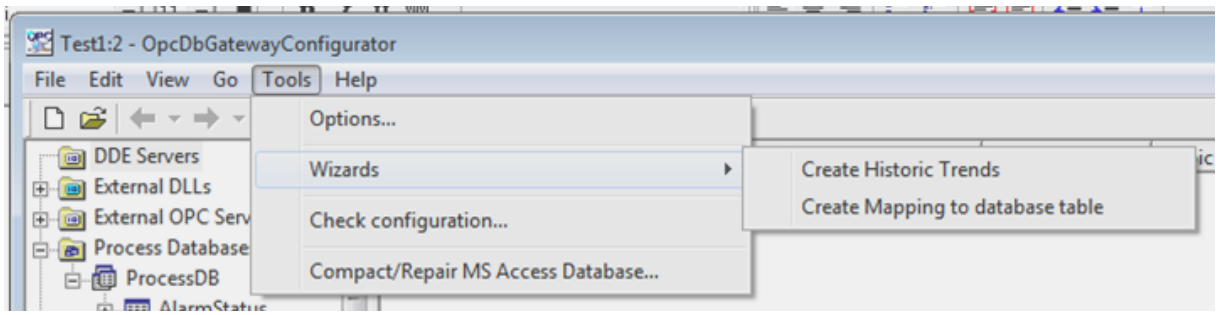


Figure: *Calling the "Create Mapping to database table"*

In case that you need create table on database according to the table definition in the configuration chose a table in tree view and, in the related dialog box, click button "Create" (figure bellow).

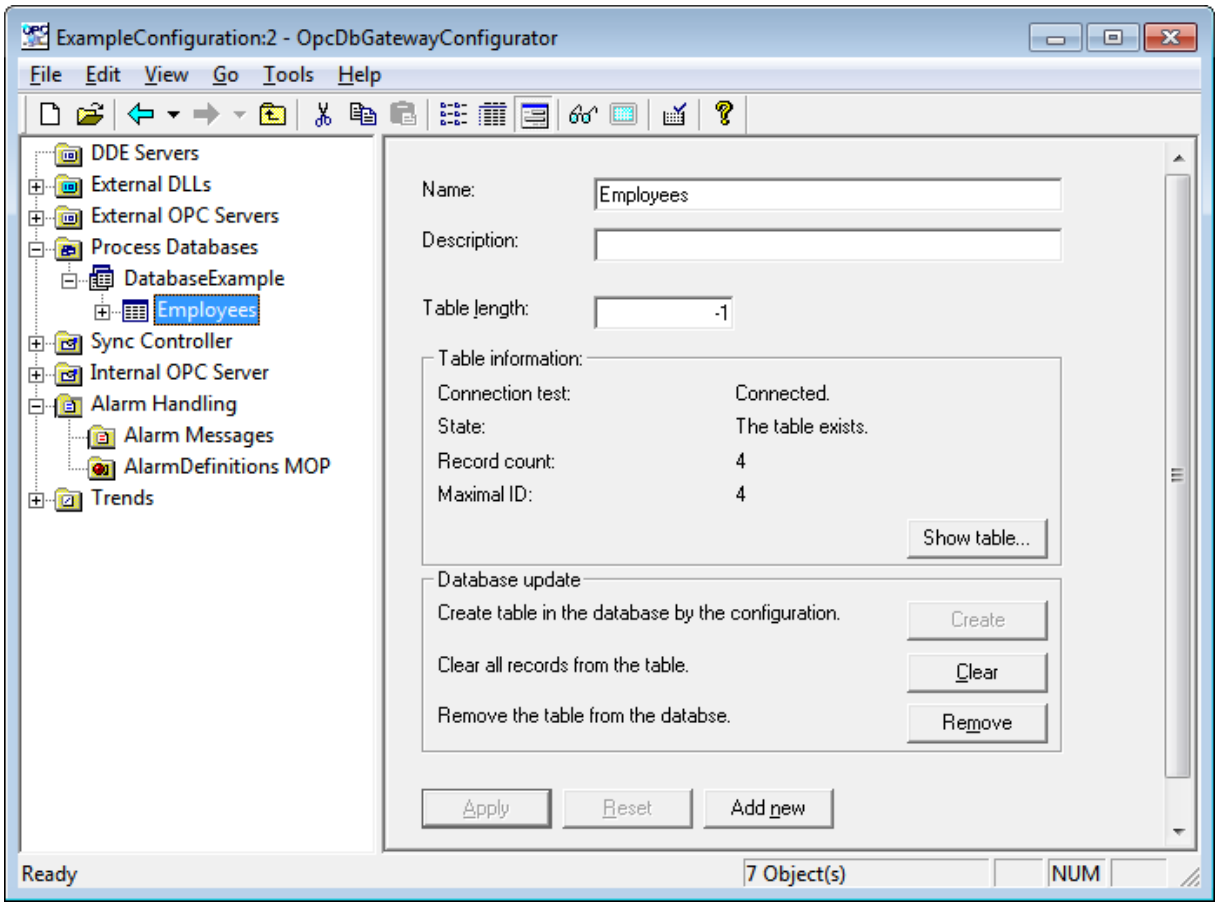


Figure: *Working with database tables*

Using dialog box according to the figure above, it is possible:

- to see contents of the table (button **Show table**),
- to remove it from configuration and also from database (button **Remove**)
- to clear content of the table (but ton **Clear**)

6.4.2.1 Database table mapping wizard

Wizard should be used for tables that have not been created in configuration yet. It means, the table must be created in database first to be able use wizard.

Wizard contains following main steps:

1. Choosing a database either (**Figure 1**)
   a. already configured
   b. existing but not configured – new connection string must be defined or created using connection string wizard (see defining connection string)
2. Choosing table from database that have to be mapped to configuration (**Figure 2**)
3. Optional step (when Checkbox "Create elements to see specific table cells " is chosen) Choosing table columns that have to be mapped to **database operands** and/or to **OPC items on internal OPC server**. (**Figure 3**)
4. Optional step (when Checkbox "Map the cells of DB operands" is chosen) – mapping concrete table cells (instead of whole columns) to be mapped to **database operands** and/or to **OPC items on internal OPC server**. (**Figure 4**)

Remarks:

1. In the 2nd step only those database tables that have not been already created in configuration will be possible to choose in the list box
2. In the 3rd step only those table columns will be possible to choose in list box that have not been already mapped
3. If a table has already been mapped we cannot additionally map unused columns. To be able add the columns ,the table must be deleted from configuration and mapping of table must be done from beginning.

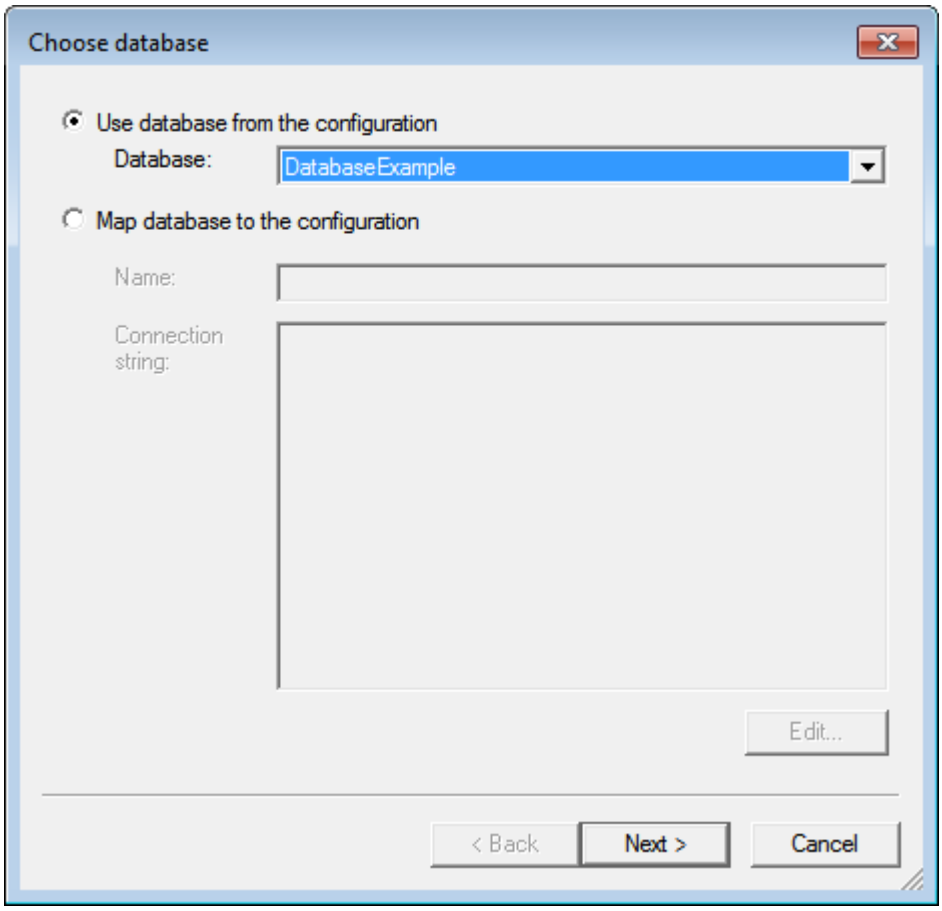


Figure 1: *Starting the wizard – choose using database from configuration or creating connection string to another database*

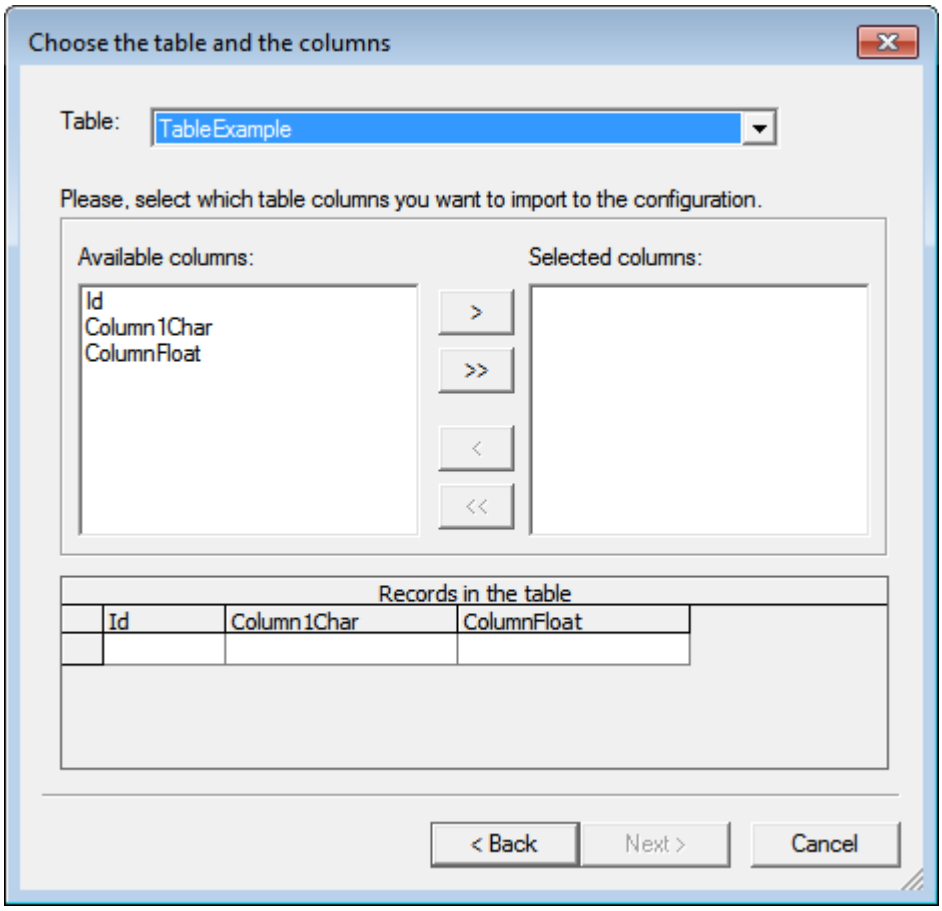


Figure 2: Choosing columns (only not configured columns are shown between available columns) to put into the configured table

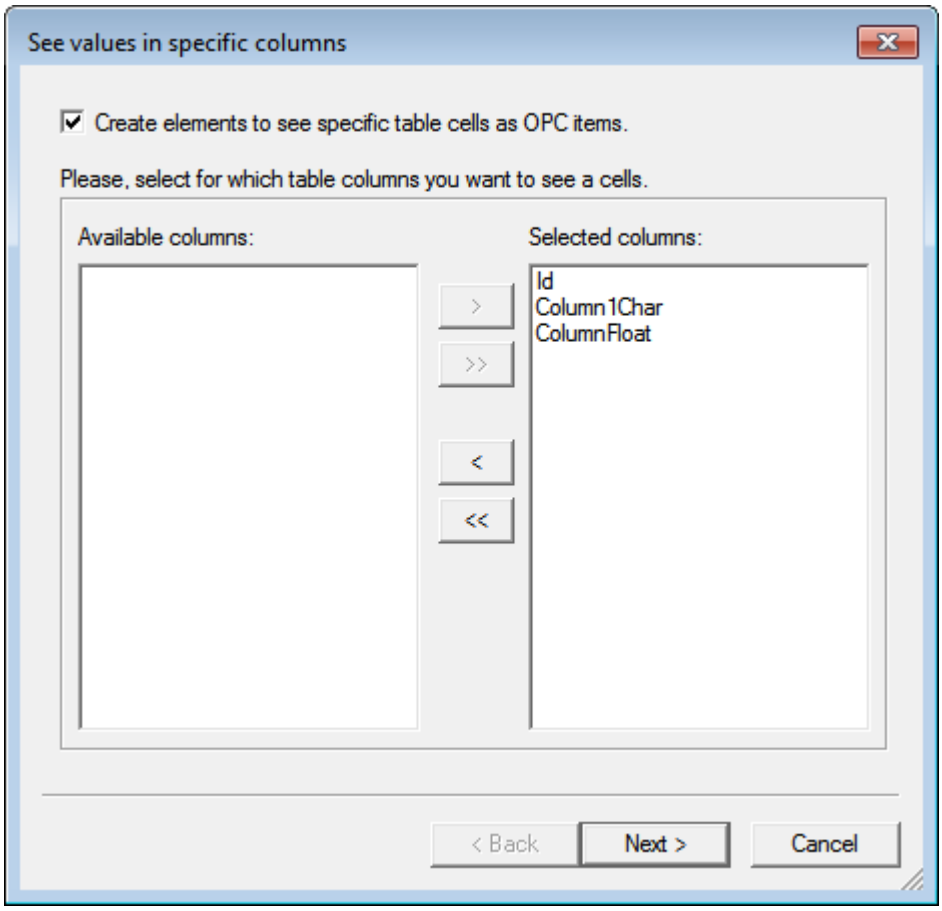


Figure 3: Choosing columns for mapping

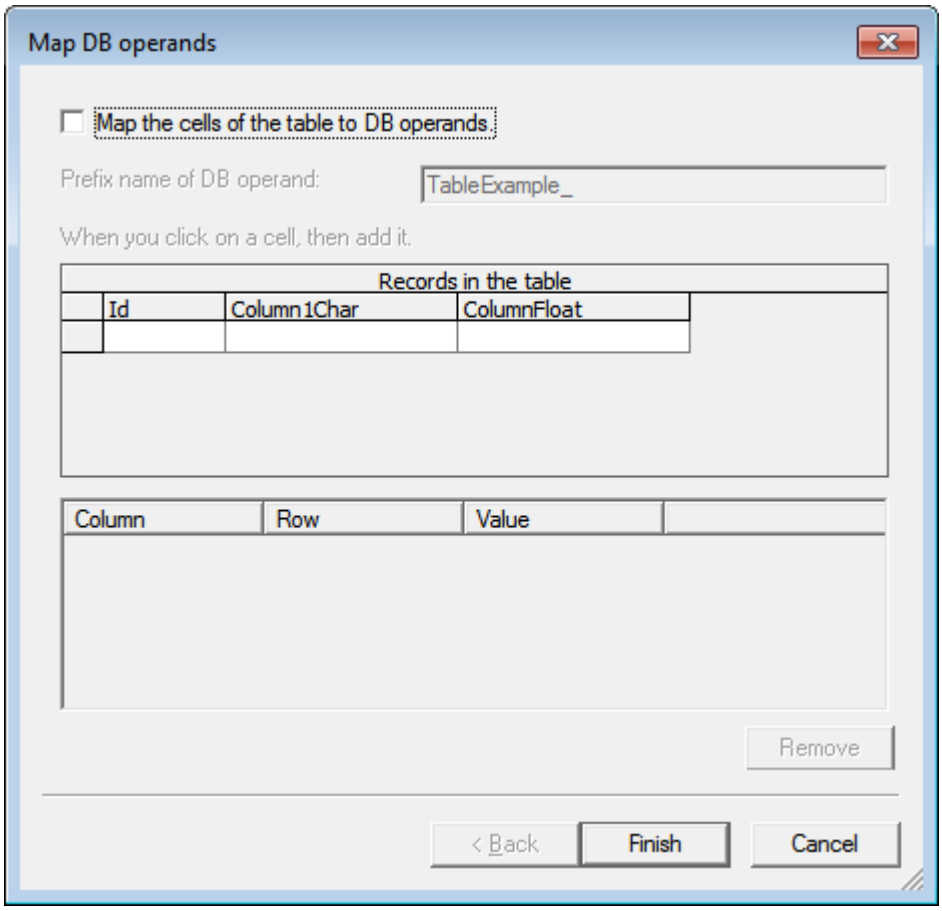


Figure 4: Choosing cells for mapping into database operands

6.4.2.2 Create table in configuration

To create new table on a database, chose the database in the tree view and from the context menu chose **New->Process Tables**.
After defining name and table description using the table dialog box crate definitions of columns as in the figure bellow. Chose the table in the tree view and using context menu chose **New->Columns**

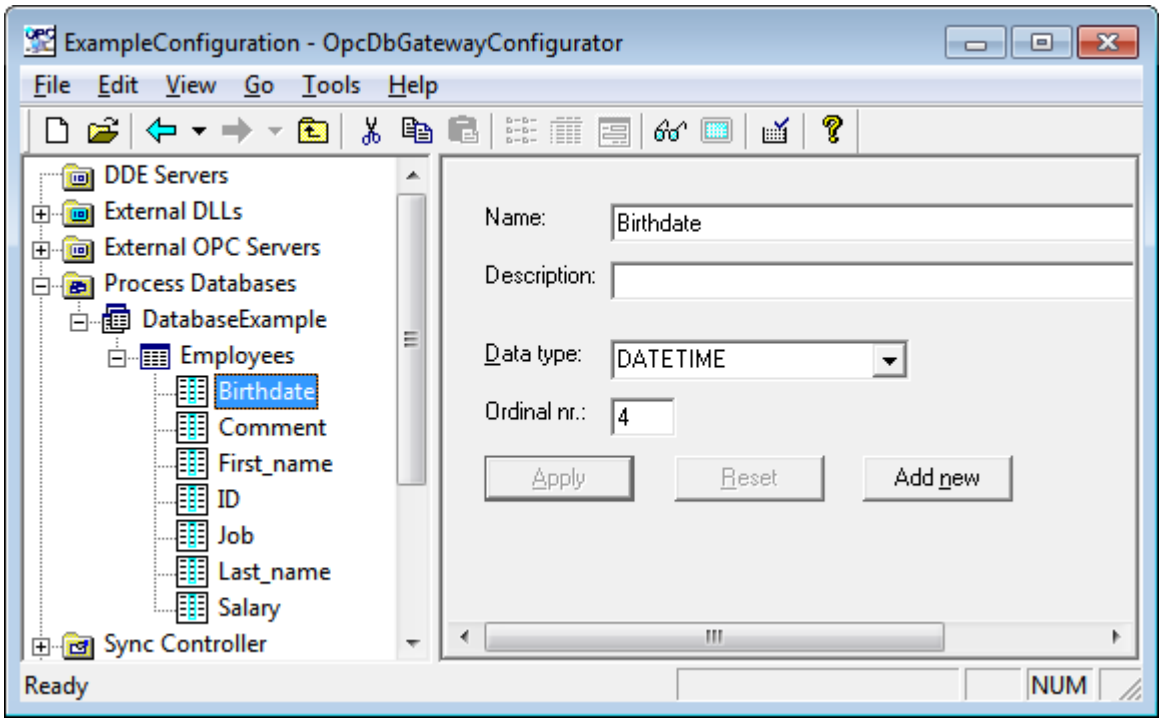


Figure: *Defining columns on database table*

The position of the column in the table is given by Ordinal Nr. It does not depend on order of columns defining.

6.4.3 Queries

SQL queries provide an easy way how to manipulate with the data in the process databases. They can be used for updating values, inserting new records, deleting records from tables or even selecting data from one table and inserting them into another. The main purpose of the queries is to select data from a database table and generate a report from the result set. Also, the queries can be used as operands in function block commands - e.g. to delete records or to update values of one or more existing records.

Parametrization of Query
One very interesting feature of queries is their parametrization. Parameters provide very powerfull tool of changing of the query definition according to current value of one or more memory operands.

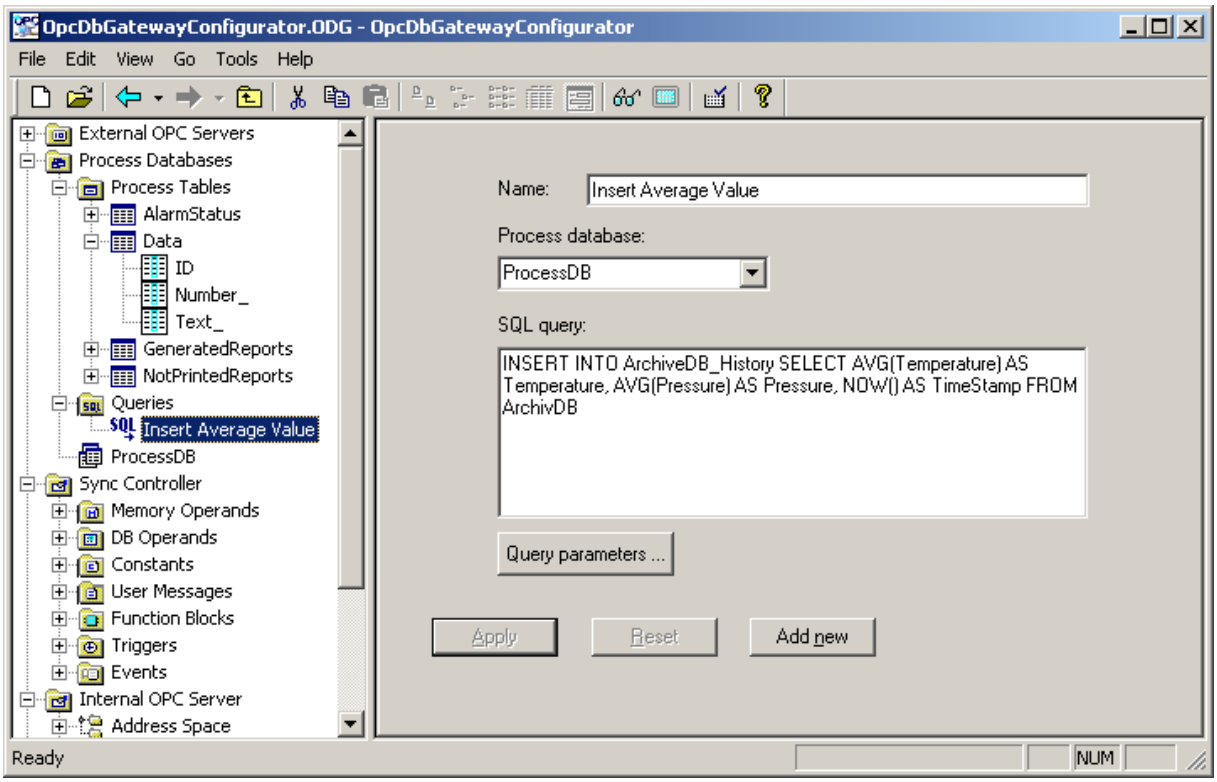


Figure 75: *Queries*

The configuration parameters of a query are following:

Process database	the process database where the query should be executed
SQL query	Any standard SQL command - SELECT, UPDATE, INSERT, DELETE ...

Table 14 - Queries parameters

How to parametrize an SQL query?

The SQL query can contain up to 6 parameters - %1%, %2%, ..., %6%.
Each parameter represents one memory operand. Before the query is executed the current value of the memory operand is inserted into defined position of the SQL query string. Afterwards is the SQL query executed.

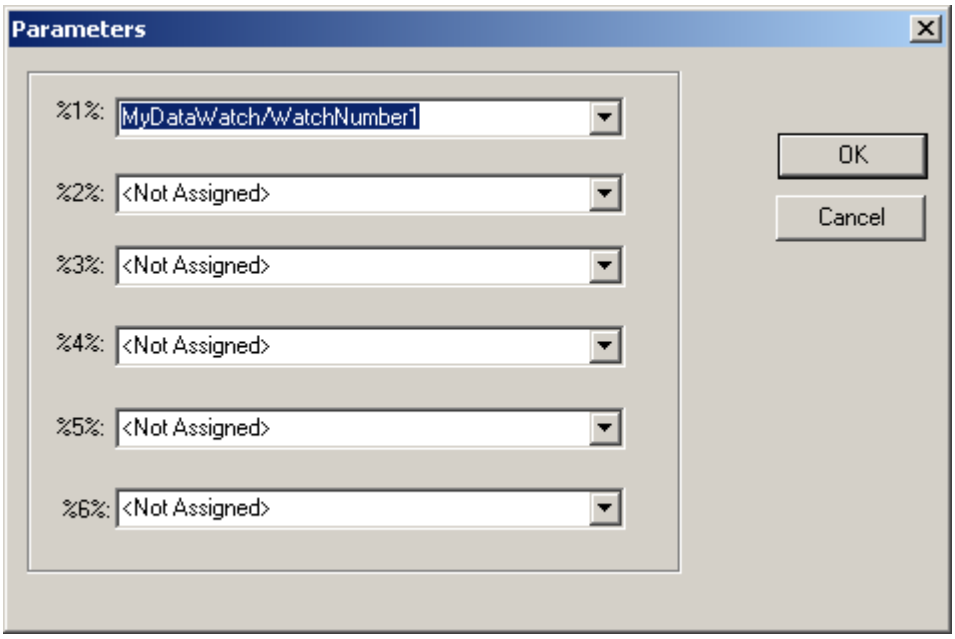


Figure 76: *Query parameters*

Remark: from ver.5 query is placed under related database in tree view
Related articles
Standard Query Language (SQL)

6.5 Sync Controller

The **Sync Controller** (or SyncController thread) executes synchronous events and the function block **Main**. The controller periodicaly reads the message queue where synchronous events are queued and executes them according to their priority. Afterwards it executes the main function block. The period of the controller is defined in the dialog box bellow.

The **Main** function block and the function blocks activated through synchronous events are executed during one period of the SyncController.

1. The Sync Controller detects which opc variables are used as inputs and outputs of those function blocks that should be executed.
2. The input opc variables are read from external opc servers.
3. The function blocks activated through the events are executed. The order of execution is defined by the priority of the event. The higher is the priority the earlier is the function block executed.
4. The function block **Main** is executed.
5. Finally, new values are written to output OPC variables.

In the folder **Sync Controller** in the Tree view the user can define all general parameters of the OpcDbGateway runtime.

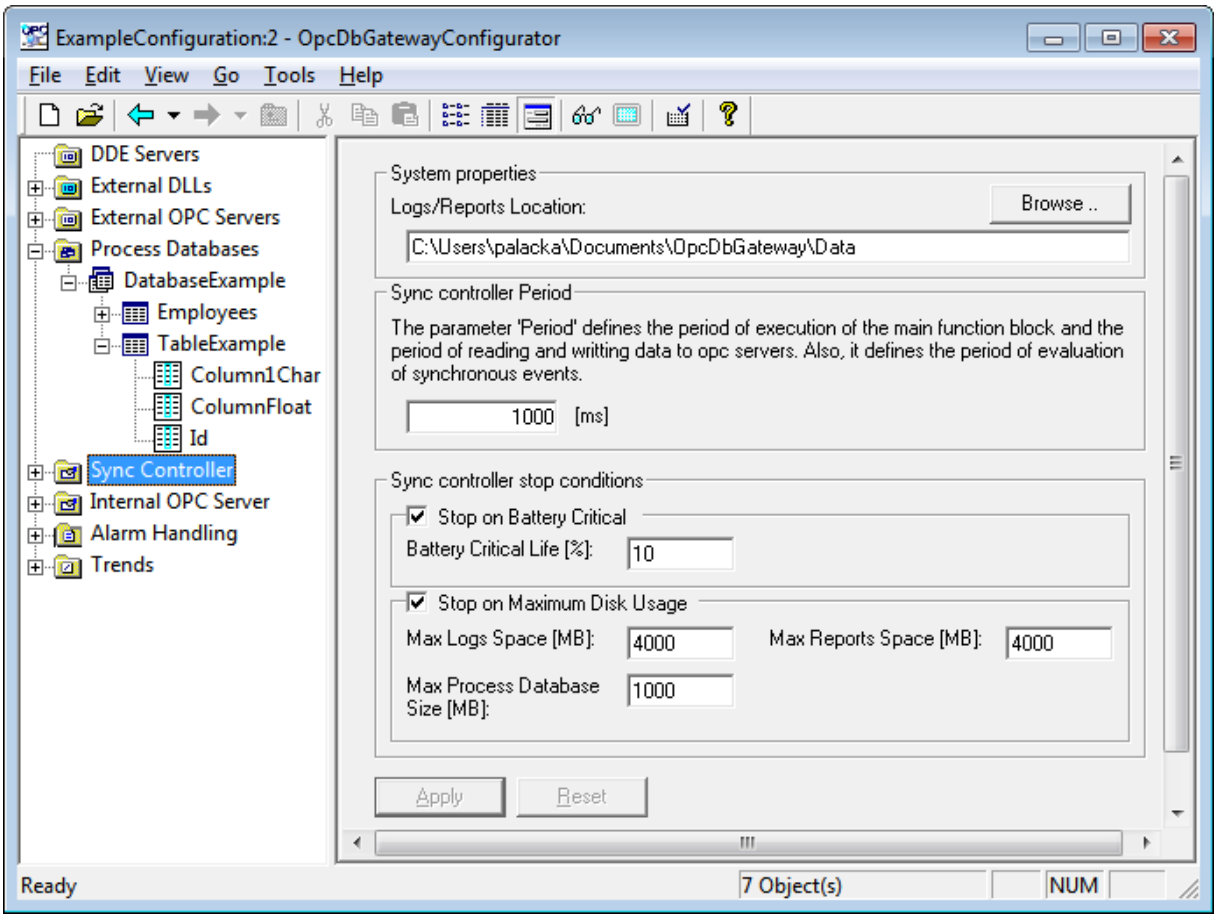


Figure: *Sync controllerr settings*

Here is the complete set of the parameters:

Logs/Reports Location	Specifies an output directory where all generated reports and log files will be placed.

Table 1 Logs/Reports Location

Sync. contrpller Period	This parameter defines the period of the synchronous controller

Table 2 - Sync. controller Period

Max Logs Space	The maximum allowed space used by log files.
Max Reports Space	The maximum allowed space used by reports
Battery Critical Life Percent	The critical value of the battery life percent status.
Max. Process Database size	The critical value of process Database size

Table 3 - Stop conditions

There are several **system variables** that provide information about the controller.

Period	The requested period of the controller [ms]
PeriodMeasured	The last measured period of the controller [ms] - the distance between two synchronization impulses
PeriodCounter	The number of synchronization impulses received by the controller since the start of the runtime
Cycle	The length of the execution of all synchro events and main function block in the last controller's cycle [ms]
CycleMax	The maximum measured value of Cycle since the start of the OpcDbGateway runtime
CycleMin	The minimum measured value of Cycle since the start of the OpcDbGateway runtime
CycleLimitCounter	This counter is incremented whenever the Cycle >= Period
CycleLimitPercentage	CycleLimitCounter /PeriodCounter * 100 [%]

Table 26 -System variables that provide information about controller

Related articles

Log files
Reports
Data persitence
Disk and memory monitor
Power status
Controller
Events
Function blocks

6.5.1 Memory operands

Memory operands are variables assigned to the internal memory of the OpcDbGateway runtime. They can be used as source or destination operands of function block commands.

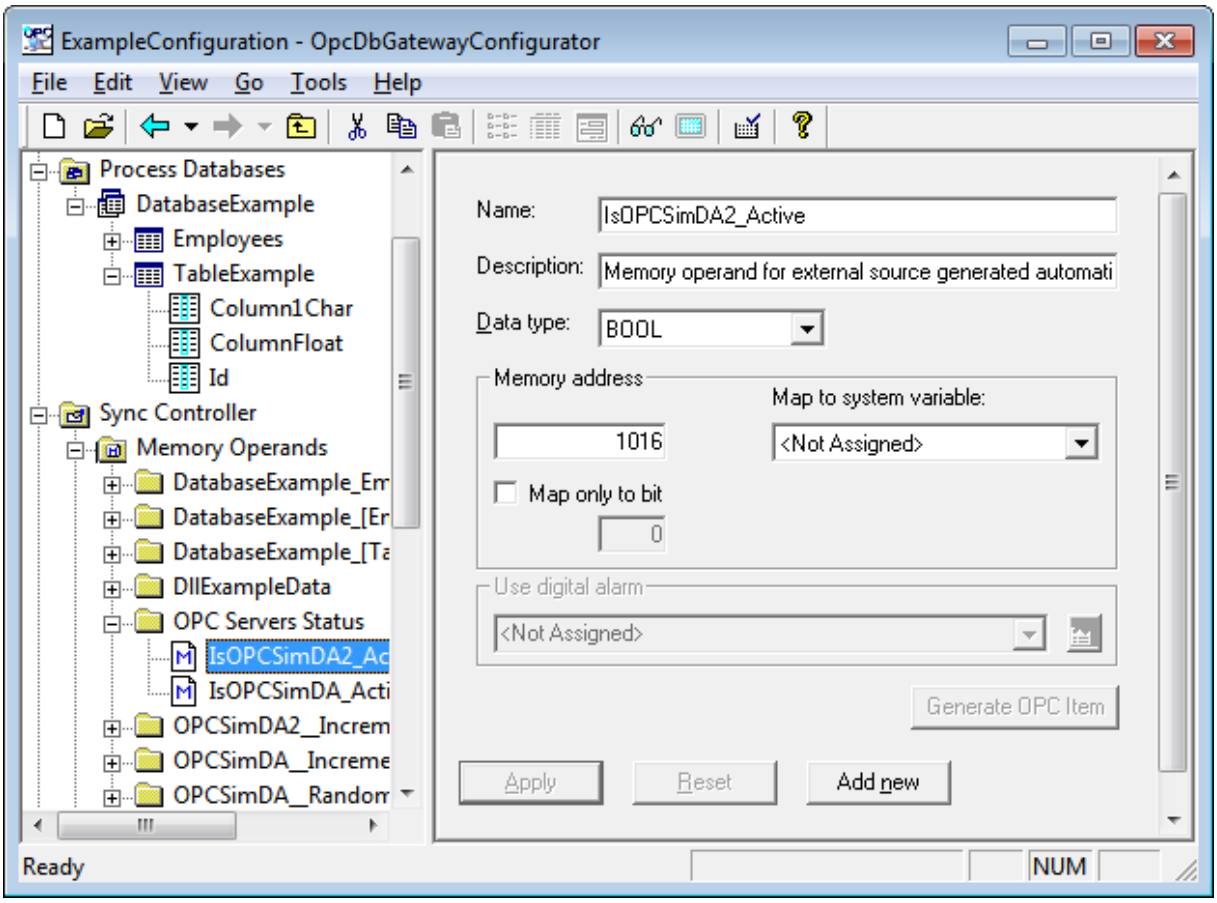


Figure: *Memory operand definition dialog box*

How to add a new memory operand?

Click with the right mouse button on the folder **Memory operands** and select the command **New⇨Folder** to which you want to store the memory operand or you can use already created folder. And than with the right mouse button on folder you can select command **New⇨Memory operands**. A new memory item will be created. The following parameters have to be defined:

DataType	The data type of the memory operand (bool, byte, currency, date, double, dword, float, integer, long, short, string, word)
Memory address	The address of the memory operand in the internal memory of the OpcDbGateway runtime
Bit	If the data type of the memory operand is BOOL it is possible to access bits (from 0 to 15) of the memory address.
Digital alarm	Assigns an alarm definition to the memory operand.

Table 11 - Memory operand parameters

How are memory operands related to opc items?

The memory operands are related to opc items of external opc servers through the memory

addresses.
We have for example an opc item **ServerX.Digitals.IO_1001** mapped to the memory address 2134.
We can define a new memory operand and assign it to the memory address 2134.
As a result of the relation, we can use the memory operand for reading from and writing to the opc item **ServerX.Digitals.IO_1001**.
It is possible to create memory operands related to opc items during mapping of opc items to the memory. This makes the configuration job faster. For more information see the article
Opc items mapping.

There is possible to define how many memory operands can be used in configuration as well as data persistence - if values of memory operands have to be saved to configuration database and to be used when runtime application is restarted.

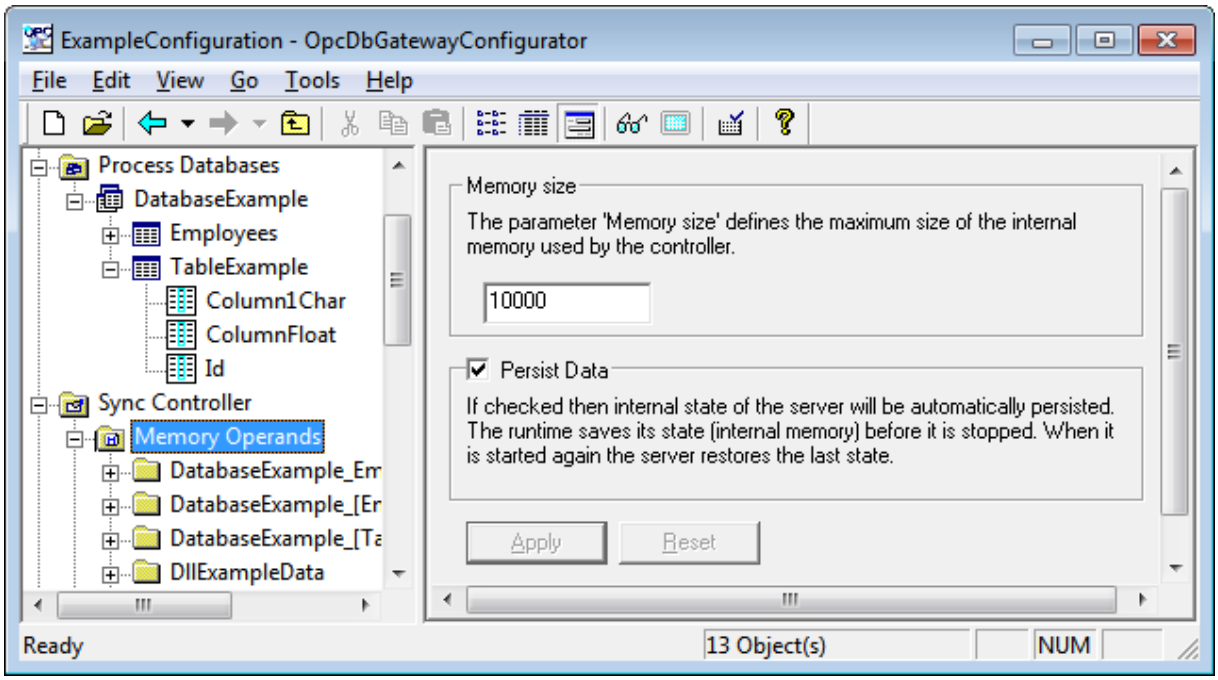


Figure: *Defining max. usable nunber of memory operands used in a configuration and persistence of memory operands when runtime is stopped*

Related articles
External OPC servers

6.5.2 DB Operands

DB Operands represent cells or columns of a database table. They can be used as source or destination operands in function block commands.

How to define a new DB Operand?

Click with the right mouse button on the folder **DB Operands** and select command **New⇨DB Operand**. A new **DB Operand** item will be inserted. Following parameters have to be defined:

DB Operands can be automatically created using wizard for mapping of database tables

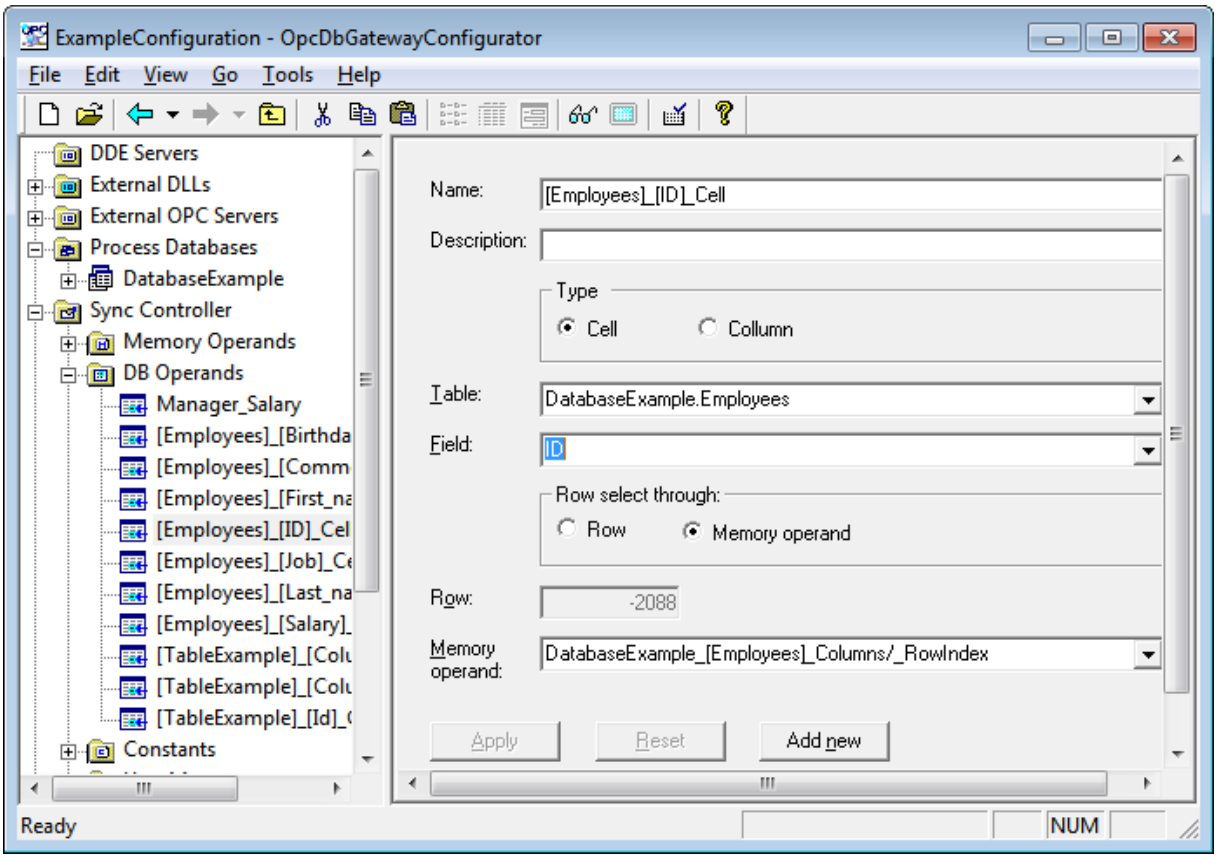


Figure 79: *DB Operands*

Table	specifies the table in the process database
Field	specifies the field in the table (column)
Row	specifies the row in the table (the row 1 is the first record in the table) If the parameter Row is equal to zero then the DB Operand represents all rows in the column otherwise it is just one cell.
Memory operands	Row is possible to define through the Memory operand.

Table 12 - DB Operand parameters

Related articles
Process tables

6.5.3 Constants

Constants don't change their value. They can be used as input operands in function block commands.

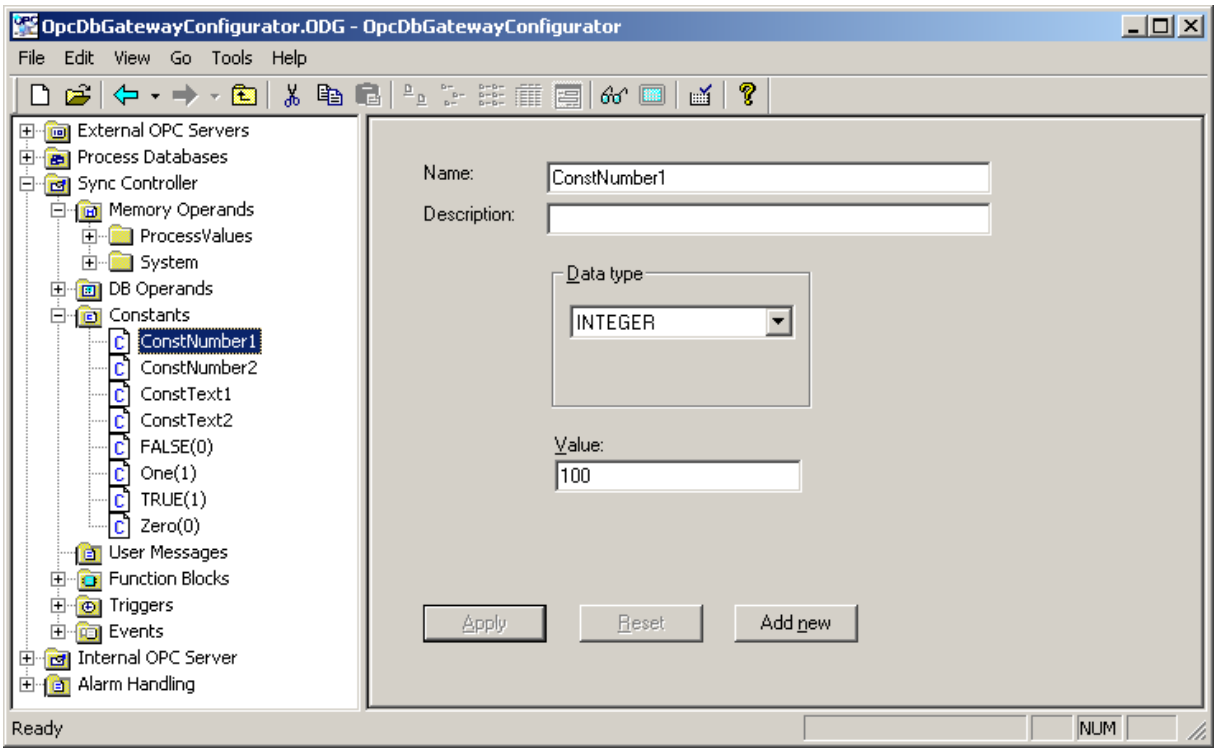


Figure 80: *Constants*

The parameters which are necessary to define a constant are following:

Data type	The data type of the constant (bool, byte, currency, date, double, dword, float, integer, long, short, string, word)
Value	The value of the constant

Table 13 - Constants parameters

Related articles
Commands

6.5.4 User messages

User messages are messages that can be written to a log file of the OpcDbGateway runtime.

Parametrization of user messagess
User messages can be parametrized. Parameters provide easy way of adding additional information to the user message according to the current value of one or more memory operands.

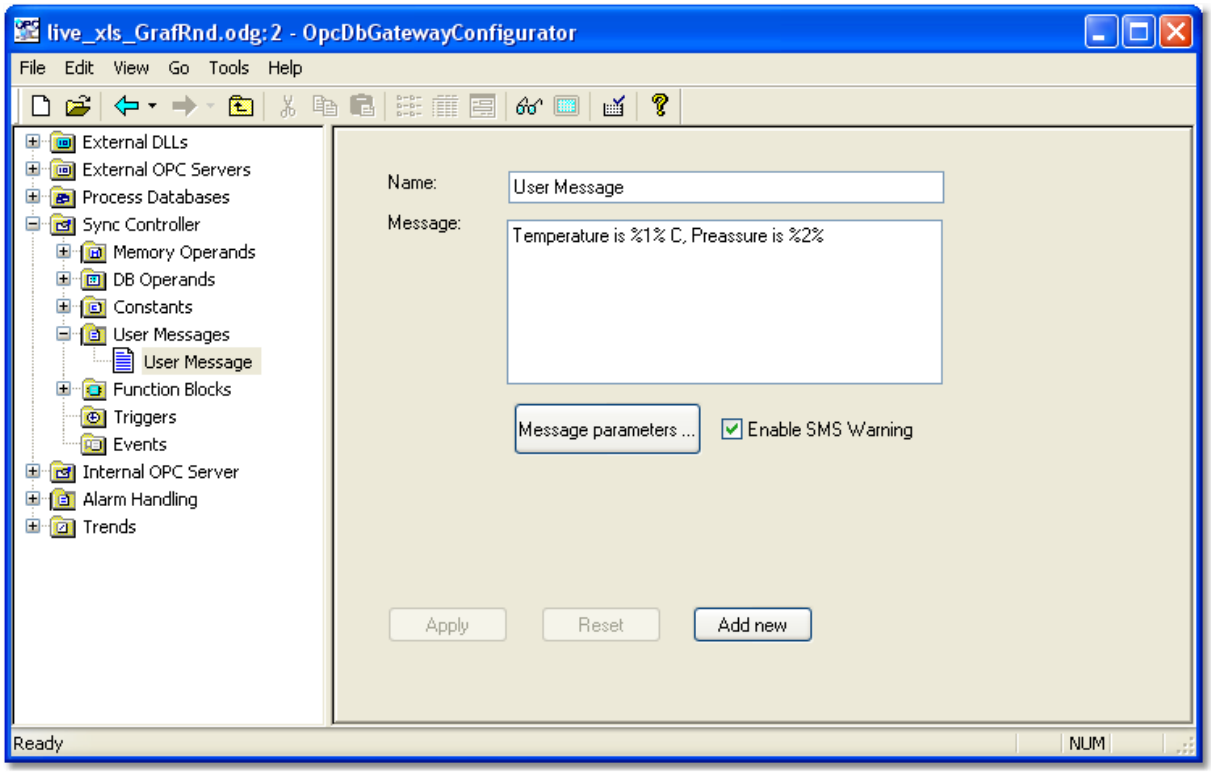


Figure 81: *User messages*

How to add a new user message?

Click with the right mouse button on the folder **User Mesages**. Choose command **New⇨UserMessages** from the context menu. A new User message item will be created. Now you can define the name and the text of the message.

How to parametrize a User message?

The user message can contain up to 6 parameters - %1%, %2%, ..., %6%.
Each parameter represents one memory operand. Before the user message is written to the log file the current value of the memory operand is inserted into defined position of the user message string. Afterwards is the message written to the log.

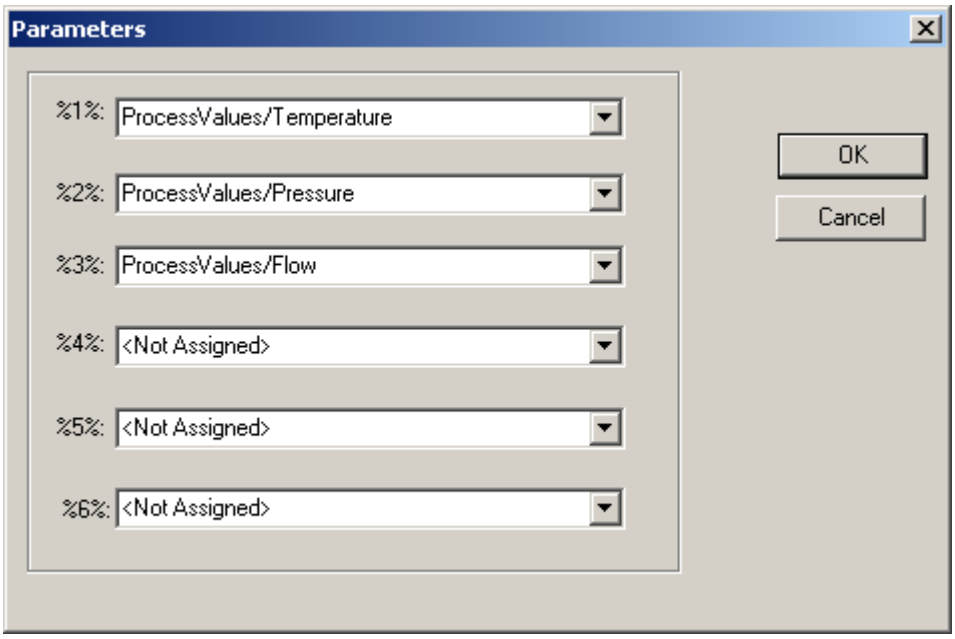


Figure 82: *User message parameters*

Related articles

Command_WRITE_MSG_TO_LOGFILE
Command_WRITE_MSG_TO_TABLE

6.5.5 Function blocks

Function block represents a group of commands that can be executed sequentialy.

The order of execution
The order of execution of commands is defined by the parameter **LineNr**.

A function block can be called
A function block can be called either from **another function block** or as a handler of an event **CallFunctionBlock**.

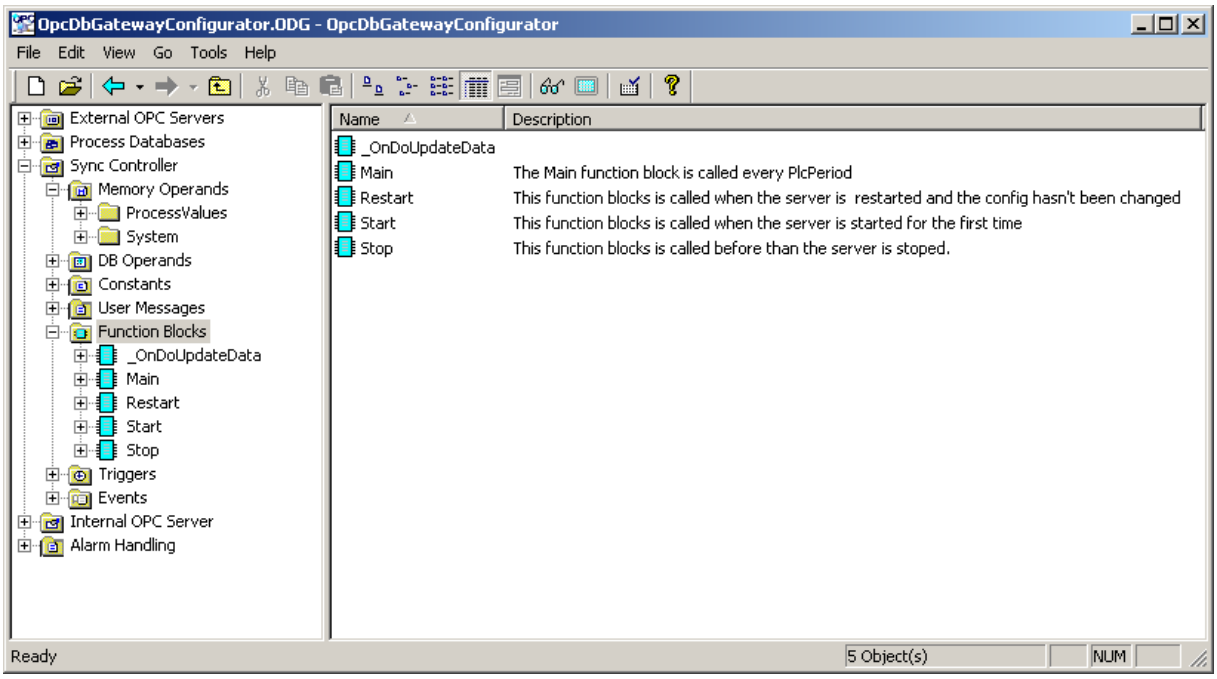


Figure 83: *Function block*

There are four standard function blocks:

Main	This function block is called in every period of the Synchronous Controller
Start	This function block should be used for initialization of memory operands, process tables and so on. It is called when the OpcDbGateway runtime is started for the first time. Also it is called when the configuration has been changed by the user since the last start/restart of the OpcDbGateway runtime.
Stop	This function block is called when the OpcDbGateway runtime is stopped either by the user or on power failure.
Restart	This function block is called when the OpcDbGateway runtime is stopped and started again without any change to the configuration database since the last start/restart.

Table 15 - Standard function blocks

How to add a new function block?

Click with the right mouse button on the folder **Function blocks** and choose the command **New⇨FunctionBlock**. A new **Function block** item will be inserted. Type the name and description of the function block.

Related articles

Commands

6.5.5.1 Commands

Command is an operation between two input operands. The result of the operation is stored into an output operand. The input operand can be memory operand, DB operand or a constant. The output operand can be either memory operand or DB Operand.

How to add a command to a function block?

Click with the right mouse button on the **function block** you would like to add the command to. Choose the command **New⇨Command** from the context menu. A new **Command** item will be inserted.

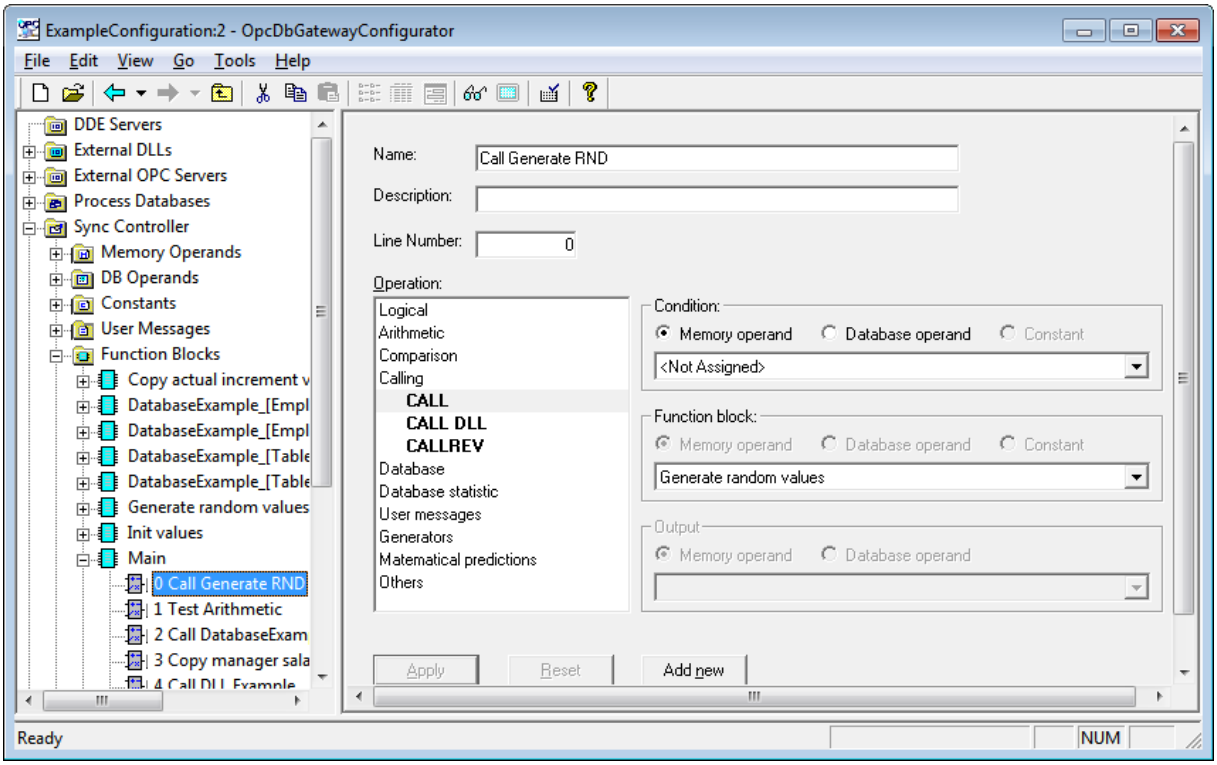


Figure: *Command dialog box*

LineNr	The line number of the command. This specifies the order of commands in a function block. The first is executed the command with the lowest number.
Operation	The operation defines what has to be done with the two input operands
Output type	The type of the output operand Mem ... memory operand DB ... database operand
Output	The output operand
Input1 type	The type of the first input operand Mem ... memory operand DB ... database operand Const ... constant
Input1	The first input operand
Input2 type	The type of the second input operand Mem ... memory operand DB ... database operand Const ... constant
Input2	The second input operand

Table - Command parameters

Here is the list of available operations :

Logical

AND	Logical AND
OR	Logical OR
NAND	Logical NAND
NOR	Logical NOR

Arithmetic

ADD	Addition
SUB	Subtraction
MUL	Multiplication
DIV	Division

Comparison

EQ	Is equal ?
NEQ	Is not equal?
GREATER	Is greater?
LOWER	Is lower?

Statistic

MIN	Finds minimum value in a column of a database table
MAX	Finds maximum value in a column of a database table
AVG	Calculates average of values in a column of a database table
COUNT	Counts the number of rows in a column of a database table
SUM	Calculates sum of values in a column of a database table

Database

WRITE_ARRAY_TO_TABLE	Adds new record to a table
WRITE_ARRAY_TO_TABLE_Ex	Adds new record to a table - for each value adds quality and timestamp
WRITE_ARRAY_TO_ACTUAL_TREND	Adds new record to a table - if the table is full deletes the oldest record
REMOVE_ALL_RECORDS	Removes all records from a table
COPY_COLUMN	Copies a column
COPY_TABLE	Copies a table
QUERY	Executes an SQL query
FIND_FIRST	Find the first value in column.
FIND_NEXT	Find the next value in column.

User messages

WRITE_MSG_TO_LOGFILE	Writes an user message to a log file
WRITE_MSG_TO_TABLE	Writes an user message to a process table - adds new record to the table

Regression, Extrapolation, Prediction

SLOPE	calculates slope (factor) of a column of a database table
EXTRAPOLATE	extrapolates values of a column of a database table
PREDICT_DX	predicts delta X

Others

SET	Copies value of one operand to another.
CALL	Calls a function block.
CALLREV	Calls a function block.
CALL DLL	Calls an external DLL.
IMPULS	Checks whether input operand has changed its state or not.
READ	Reads from an opc item and stores the value to a memory operand.
RND	This operation is a random number generator designed to generate a sequence of numbers.

6.5.5.1.1 Logical operations

6.5.5.1.1.1 AND

AND

Logical AND.

Result

Output = Input1 AND Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

If at least one of the operands has zero value then the output is zero. Otherwise the output value is nonzero.

Example

AND(bC, bA, bB) // bC = bA AND bB;

Related articles

Function blocks
Commands

6.5.5.1.1.2 OR

OR

Logical OR.

Result

Output = Input1 OR Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

If at least one of the operands has nonzero value then the output is nonzero. Otherwise the output value is zero.

Example

OR(bC, bA, bB) // bC = bA OR bB;

Related articles

Function blocks
Commands

6.5.5.1.1.3 NAND

NAND

Logical NAND.

Result

Output = Input1 NAND Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

If at least one of the operands has zero value then the output is one. Otherwise the output value is zero.

Example

NAND(bC, bA, bB) // bC = bA NAND bB;

Related articles

Function blocks
Commands

6.5.5.1.1.4 NOR

NOR

Logical NOR.

Result

Output = Input1 NOR Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

If at least one of the operands has nonzero value then the output is zero. Otherwise the output value is nonzero.

Example

NOR(bC, bA, bB) // bC = bA NOR bB;

Related articles

Function blocks
Commands

6.5.5.1.2 Aritmetic operations

6.5.5.1.2.1 ADD

ADD

Addition of input operands.

Result

Output = Input1 + Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

No

Example

ADD(C, A, B) // C = A + B;

Related articles

Function blocks
Commands

6.5.5.1.2.2 SUB

SUB

Subtracts the value of the second of the second operand from the value of the first operand.

Result

Output = Input1 - Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

No

Example

SUB(C, A, B) // C = A - B;

Related articles

Function blocks

Commands

6.5.5.1.2.3 MUL

MUL

Multiplication of input operands.

Result

Output = Input1 * Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

No

Example

MUL(C, A, B) // C = A * B;

Related articles

Function blocks
Commands

6.5.5.1.2.4 DIV

DIV

Divides the value of the first operand by the value of the second operand.

Result

Output = Input1 / Input2

Parameters

Input1 Memory operand, DB operand or Constant

Input2 Memory operand, DB Operand or Constant

Remarks

No

Example

DIV(C, A, B) // C = A / B;

Related articles
Function blocks
Commands

6.5.5.1.3 Comparison operations

6.5.5.1.3.1 EQ

EQ

Compares input operands.

Result

Output	if Input1 is equal to Input2 then the output is nonzero. Otherwise it is zero.

Parameters

Input1	Memory operand, DB operand or Constant
Input2	Memory operand, DB Operand or Constant

Remarks
No

Example

```
EQ(bC, A, B)                    // bC = (A == B);
```

Related articles
Function blocks
Commands

6.5.5.1.3.2 NEQ

NEQ

Compares input operands.

Result

Output	if Input1 is not equal to Input2 then the output is nonzero. Otherwise it is zero.

Parameters

Input1	Memory operand, DB operand or Constant
Input2	Memory operand, DB Operand or Constant

Remarks
No

Example
NEQ(bC, A, B) // bC = (A <> B);

Related articles
Function blocks
Commands

6.5.5.1.3.3 GREATER

GREATER

Compares input operands.

Result

Output	if Input1 is greater than Input2 then the output is nonzero. Otherwise it is zero.

Parameters

Input1	Memory operand, DB operand or Constant
Input2	Memory operand, DB Operand or Constant

Remarks
No

Example
GREATER(bC, A, B) // bC = (A > B);

Related articles
Function blocks
Commands

6.5.5.1.3.4 LOWER

LOWER

Compares input operands.

Result

Output	if Input1 is lower than Input2 then the output is nonzero. Otherwise it is zero.

Parameters

Input1	Memory operand, DB operand or Constant
Input2	Memory operand, DB Operand or Constant

Remarks
No

Example
GREATER(bC, A, B) // bC = (A < B);

Related articles
Function blocks
Commands

6.5.5.1.4 Statistic operations

6.5.5.1.4.1 MIN

MIN

Finds the minimum value in a process table column.

Result

Output	The minimum value - Memory or DB operand

Parameters

Input1	Source column - DB operand
Input2	Not used

Remarks
No

Example
MIN (MinVal, ColSrc, NULL)

Related articles
DB Operands
Function blocks
Commands

6.5.5.1.4.2 MAX

MAX

Finds the maximum value in a process table column.

Result

Output	The maximum value - Memory or DB operand

Parameters

Input1	Source column - DB operand
Input2	Not used

Remarks

No

Example

MAX (MaxVal, ColSrc, NULL)

Related articles

DB Operands
Function blocks
Commands

6.5.5.1.4.3 AVG

AVG

Calculates the average value from all values of a process table column.

Result

Output	The average value - Memory or DB operand

Parameters

Input1	Source column - DB operand
Input2	Not used

Remarks

No

Example

AVG (AvgVal, ColSrc, NULL)

Related articles

DB Operands
Function blocks
Commands

6.5.5.1.4.4 COUNT

COUNT

Counts the number of rows (values) in a process table column.

Result

Output	count - Memory or DB operand

Parameters:

Input1	Source column - DB operand
Input2	Not used

Remarks

No

Example

COUNT (Cnt, ColSrc, NULL)

Related articles

DB Operands
Function blocks
Commands

6.5.5.1.4.5 SUM

SUM

Calculates the sum from all values of a process table column.

Result

Output	the sum - Memory or DB operand

Parameters

Input1	Source column - DB operand
Input2	Not used

Remarks

No

Example

SUM (SumVal, ColSrc, NULL)

Related articles
DB Operands
Function blocks
Commands

6.5.5.1.5 Database operations

6.5.5.1.5.1 COPY_COLUMN

COPY_COLUMN

Copies a process table column.

Result

Output	Destination column - DB operand

Parameters

Input1	Source column - DB operand
Input2	Process table

Remarks
The source and the destination columns should be of the same datatype.

Example
COPY_COLUMN (ColDest, ColSrc, NULL)

Related articles
DB Operands
Function blocks
Commands

6.5.5.1.5.2 COPY_TABLE

COPY_TABLE

Copies a process table.

Result

Output	Destination process table

Parameters:

Input1	Source process table
Input2	Not used

Remarks

The source and destination tables should have the same structure.

Example

COPY_TABLE (TableDest, TableSrc, NULL)

Related articles

Process tables
Function blocks
Commands

6.5.5.1.5.3 WRITE_ARRAY_TO_TABLE

WRITE_ARRAY_TO_TABLE

Writes an array of memory operands into a process database table. The array is written as one record.

Result

Output	If the record is successfully written to the table the output is zero (FALSE). If the table is full then the output is nonzero (TRUE).

Parameters

Input1	ProcessTable
Input2	Memory operand

Remarks

The command has only two input parameters. Therefore the number of operands written to the table is defined by the number of columns of the process table without the column ID. The first memory operand, which is defined as Input2, is written to the second column after the column ID. The second operand, which is assigned to the next memory address, is written to the third column and so on.
If the table is full or in other words, if the number of records of the table is equal to the length of the table then the write operation fails and the output operand is nonzero.

Example

WRITE_ARRAY_TO_TABLE(C, TableA, B0)

Related articles

Function blocks
Commands

6.5.5.1.5.4 WRITE_ARRAY_TO_TABLE_Ex

WRITE_ARRAY_TO_TABLE_Ex

Writes an array of memory operands into a process database table. The array is written as one record. The behaviour of the command is the same as WRITE_ARRAY_TO_TABLE. The main difference is that from each memory operand are taken three values - value, quality and timestamp. Then they are written into three separatate columns of the process table.

Result

Output	If the record is successfully written to the table the output is zero (FALSE). If the table is full then the output is nonzero (TRUE).

Parameters

Input1	ProcessTable
Input2	Memory operand

Remarks

The command has only two input parameters. Therefore the number of operands written to the table is defined by the number of columns of the process table without the column ID divided by three - for each operand is represent by three values. The first memory operand, which is defined as Input2, is written to the second column after the column ID. The timestamp (datatype DATETIME) is written to the third column and the quality (datatype INTEGER) to the fourth. The second operand, which is assigned to the next memory address, is written to the fifth column, its timestamp to the sixth and the quality to the seventh and so on.
If the table is full or in other words, if the number of records of the table is equal to the length of the table then the write operation fails and the output operand is nonzero.

Example

WRITE_ARRAY_TO_TABLE_Ex(C, TableA, B0)

Related articles

Function blocks
Commands

6.5.5.1.5.5 WRITE_ARRAY_TO_ACTUALTREND

WRITE_ARRAY_TO_ACTUALTREND

Writes an array of memory operands into a process database table.

Result

Output	no output

Parameters

Input1	ProcessTable
Input2	Memory operand

Remarks

The command has only two input parameters. Therefore the number of operands written to the table is defined by the number of columns of the process table without the column ID. The first memory operand, which is defined as Input2, is written to the second column after the column ID. The second operand, which is assigned to the next memory address, is written to the third column and so on.
If the table is full then the oldest record is deleted and the new record is appended to the end of the table.

Example

WRITE_ARRAY_TO_ACTUALTREND (NULL, TableA, B0)

Related articles

WRITE_ARRAY_TO_TABLE
WRITE_ARRAY_TO_TABLE_Ex
Function blocks
Commands

6.5.5.1.5.6 REMOVE_ALL_RECORDS

REMOVE_ALL_RECORDS

Removes all records from a process table.

Result

Output	No output value

Parameters

Input1	ProcessTable
Input2	Not used

Remarks

No

Example

REMOVE_ALL_RECORDS (NULL, TableA, NULL)

Related articles

Function blocks
Commands

6.5.5.1.5.7 QUERY

QUERY

Executes an SQL query

Result

Output	No

Parameters

Input1	Query
Input2	Not used

Remarks

No

Example

QUERY (Null, query1, NULL)

Related articles

Queries
Function blocks
Commands

6.5.5.1.5.8 FIND_FIRST

FIND_FIRST

Find the first founding value in specified column. For find next value in specified column use command FIND_NEXT.

Result

Output	Memory or DB operand - which consists row number with founding value

Parameters

Input1	Column - DB operand where I want to find the founding value
Input2	Constant, Memory or DB operand whith value what I want to find in specified column.

Remarks

No

Example

FIND_FIRST (RowNumber, Column, FoundingValue)

Related articles

FIND_NEXT
DB Operands
Function blocks
Commands

6.5.5.1.5.9 FIND_NEXT

FIND_NEXT

Usually continue after FIND_FIRST command, but is possible to use it without that command. FIND_NEXT try to find next row from previous row in which is specified the value what to find.

Result

Output	Memory or DB operand - which consists row number with founding value

Parameters

Input1	Column - DB operand where I want to find the founding value
Input2	Constant, Memory or DB operand row number from which I want to continue finding

Remarks

No

Example

FIND_NEXT (RowNumber, Column, RowNumber)

Related articles

FIND_FIRST
DB Operands
Function blocks
Commands

6.5.5.1.6 UserMessages operations

6.5.5.1.6.1 WRITE_MSG_TO_LOGFILE

WRITE_MSG_TO_LOGFILE

Writes a user message to the OpcDbGateway runtime's log file

Result

Output	No

Parameters

Input1	User message
Input2	Not used

Remarks

No.

Example
WRITE_MSG_TO_LOGFILE (NULL, UserMsg1, NULL)

Related articles
User messages
Log files
Function blocks
Commands

6.5.5.1.6.2 WRITE_MSG_TO_TABLE

WRITE_MSG_TO_TABLE

Writes a user message to a table of a process database.

Result

Output	No

Parameters

Input1	User message
Input2	Process table

Remarks
The command adds new record to the table. It writes the current timestamp to the second column and the user message to the third column. The first column is ID.

Example
WRITE_MSG_TO_TABLE (NULL, UserMsg1, NULL)

Related articles
User messages
Function blocks
Commands

6.5.5.1.7 Regression, Extrapolation, Prediction

6.5.5.1.7.1 EXTRAPOLATE

EXRAPOLATE

Extrapolates data of a process table column.

Result

Output	The destination column where extrapolated data are stored - DB operand

Parameters:

Input1	The source column - DB operand
Input2	Defines the number of extrapolated values - Memory operand, DB operand, Constant.

Remarks

The formula **y = k*x + b** is used to calculate the extrapolated data.
The parameters **k, b** are calculated from the source column using the minimum quadrat method.

Example

EXTRAPOLATE (ColDest, ColSrc, 150)

Related articles

Function blocks
Commands

6.5.5.1.7.2 PREDICT_DX

PREDICT_DX

The row in the column represents an independent variable X. The values in each row are values of a dependent variable Y. The command PREDICT_DX calculates in how many rows from the last row will the value reach defined limit.

Result

Output	the delta X - memory or DB operand

Parameters

Input1	The source column - DB operand
Input2	The limit value

Remarks

No

Example

PREDICT_DX (dx, ColSrc, 1560)

Related articles

Function blocks

Commands

6.5.5.1.7.3 SLOPE

SLOPE

Calculates the slope from all values of a process table column.

Result

Output	The slope - Memory or DB operand

Parameters

Input1	Source column - DB operand
Input2	The number of values of the source column. The slope is calculated only from defined number of values.

Remarks

Todo

Example

SLOPE (SlopeVal, ColSrc, 100)

Related articles

DB Operands
Function blocks
Commands

6.5.5.1.8 Others

6.5.5.1.8.1 SET

SET

This command copies a value of an input operand to the output operand.

Result

Output	=Input1

Parameters:

Input1	Memory operand, DB operand or Constant
Input2	Not used

Remarks

No

Example

```
SET(A,B)                                    // A = B;
```

Related articles

Function blocks
Commands

6.5.5.1.8.2 CALL

CALL

This command calls a function block. Returns after all commands of the function block are executed.

Result

Output	Not used

Parameters

Input1	The condition.
Input2	The function block that should be executed if the condition is nonzero.

Remarks

The condition is represented by the value of either memory or DB operand. If neither of the operands is assigned then the condition is ignored and the function block is executed.

Example

```
LOWER(bCondition, A, B)                     // bCondition = A < B;
CALL    (NULL, bCondition, fbHelloWorld)      // if (bCondition) fbHelloWorld();
```

Related articles

Function blocks
Commands

6.5.5.1.8.3 CALLREV

CALLREV

This command calls a function block. Returns after all commands of the function block are executed.

Result

Output	Not used

Parameters

Input1	The condition.
Input2	The function block that should be executed if the condition is zero.

Remarks

The condition is represented by the value of either memory or DB operand. If neither of the operands is assigned then the condition is ignored and the function block is executed.

Example

```
CALLREV(NULL, bCondition, fbHelloWorld)        // if (bCondition == FALSE) fbHelloWorld();
```

Related articles

Function blocks
Commands

6.5.5.1.8.4 CALL DLL

CALL DLL

The operation calls an external DLL.

Result

Output	Memory operand If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

Parameters

Input1	External DLL
Input2	Memory operand The first item from an array of memory operands. These memory operands follow one after another as in the Process memory. The array of memory operands consists of two parts: • number of input memory operands, • number of output memory operands. the first input memory operand = Input2 the first output memory operand = Input2 + a number of input memory operands The contents of input memory operands is passed as input parameter (*lpInputs*) to the external DLL function DoProcessIO. In this function all input data are processed and stored to output parameter (*lpOutputs*). Afterwards, the function is finished and the output parameter values copied to output memory operands.

Remarks

The CALL DLL operation enables to call an external DLL as command of Function Block. In fact, the CALL DLL operation always calls the function DoProcessIO of external DLL. For more datails you can click on topic.

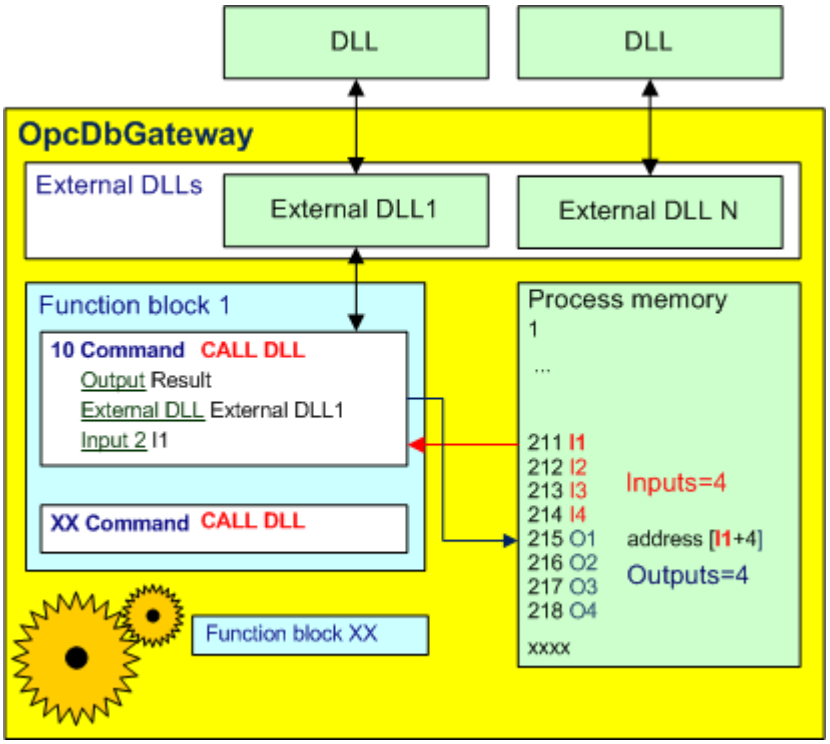


Figure 86: *The CALL DLL operation.*

Example

CALL DLL(Ret1, DllExample, IO)

Related articles

Function blocks
Commands
DoProcessIO
How to call an external DLL from OpcDbGateway?
External DLL module
Configuring External DLLs

6.5.5.1.8.5 IMPULS

IMPULS

Checks whether an input operand has changed its value from zero to nonzero or viceversa.

Result

Output	If the input operand has changed its value then the output operand is nonzero. Otherwise the output operand is zero.

Parameters

Input1	The input operand whose value should be checked.
Input2	The mode if TRUE(1) then the change from zero to nonzero will be checked. if FALSE(0) then the change from nonzero to zero will be checked.

Remarks

Todo

Example

IMPULS (bChange, Input1, TRUE)

Related articles

Function blocks
Commands

6.5.5.1.8.6 READ

READ

Reads value from an opc item and stores it to the memory operand.

Result

Output	No

Parameters

Input1	Memory operand
Input2	Not used

Remarks

Each opc item from a connected opc server is associated to one memory operand through a memory address.
Before this memory operand is used as input of one of the commands in a function block, the value of the associated opc item is read from the opc server. In some cases it is necessary to read opc items from the opc server without executing any operation above the memory operand. For this case should be used command READ, which in fact does nothing, but makes the programmer sure that associated opc item is read from the opc server.

Example

READ(NULL, A, NULL)

Related articles

OPC items mapping
Function blocks
Commands

6.5.5.1.8.7 RND

RND

This operation is a random number generator designed to generate a sequence of numbers.

Result

Output	A generated number.

Parameters

Input1	Memory operand, DB operand or Constant A minimal value for generated number.
Input2	Memory operand, DB Operand or Constant A maximal value for generated number.

Remarks

No

Example

RND(C, A, B) // C = RANDOM(A, B);

Related articles

Function blocks
Commands

6.5.6 Triggers

Triggers are used for starting of events. Each event object must have assigned one and only one trigger. In other words, the trigger defines conditions when an event should be triggered.

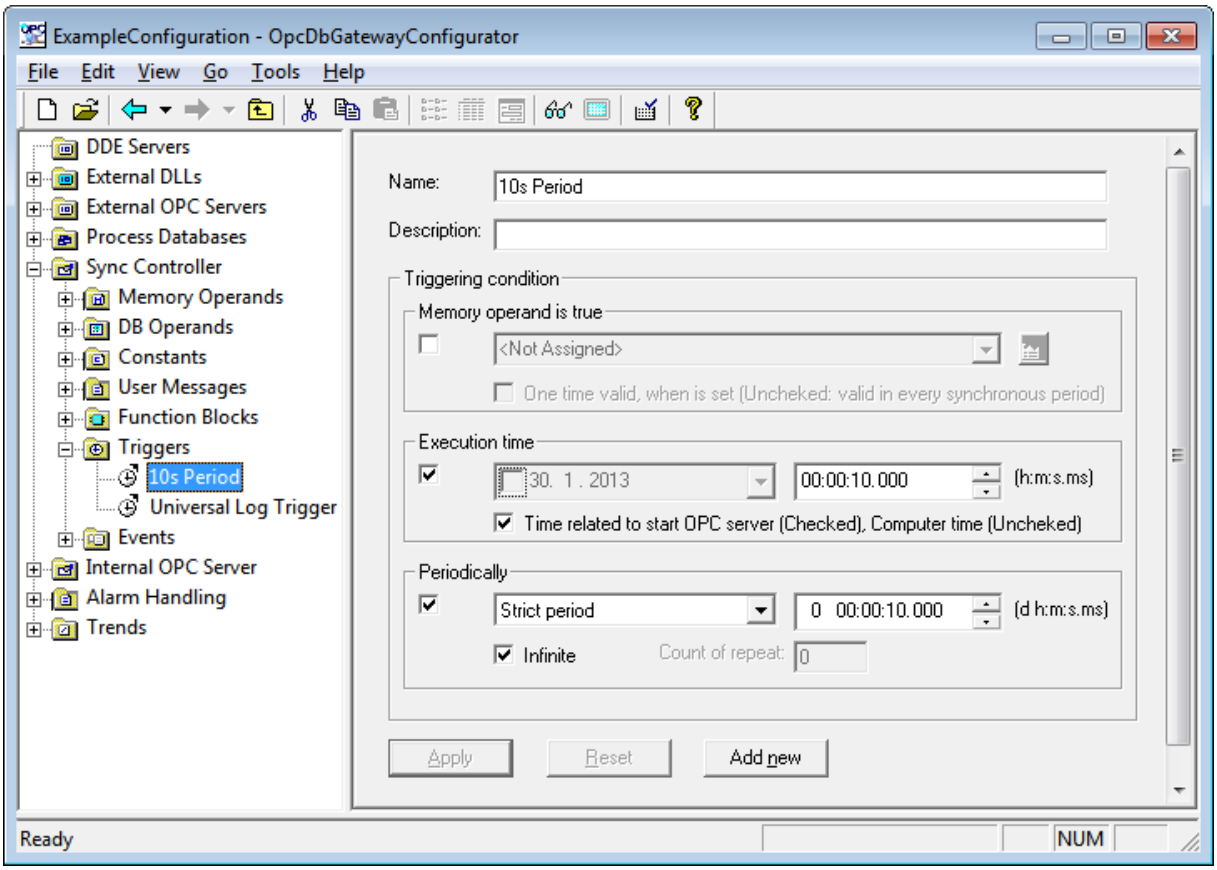


Figure: *Triggers*

How to add a new trigger?

Click with the right mouse button on the folder **Triggers**. Choose the command **New⇨Trigger** from the context menu.
A new **Trigger** item will be created. The following paramaters have to be defined for each trigger:

Related articles

Events

6.5.6.1 Triggers - functionality

Triggers are used for starting of **synchronous** (that are executed in synchronous thread) and **asynchronous** (that are executed in asynchronous thread) events. Each event has assigned one trigger object. If the trigger object is set then the event is started.

Triggers can be controlled either by a trigger variable or by a trigger timer or by both. Trigger timer can be used for one shot or periodical trigerring. Periodical trigerring can have defined count of repeats.

Trigger variable can be any boolean memory operand. The timer is defined by start time, period in seconds and the number of ticks. The timer defines when ticks should be sent to the trigger.

Triggers initiate events that are either in synchronous or asynchronous thread. Evaluation of trigger conditions is executed within special trigger thread. See figure bellow.

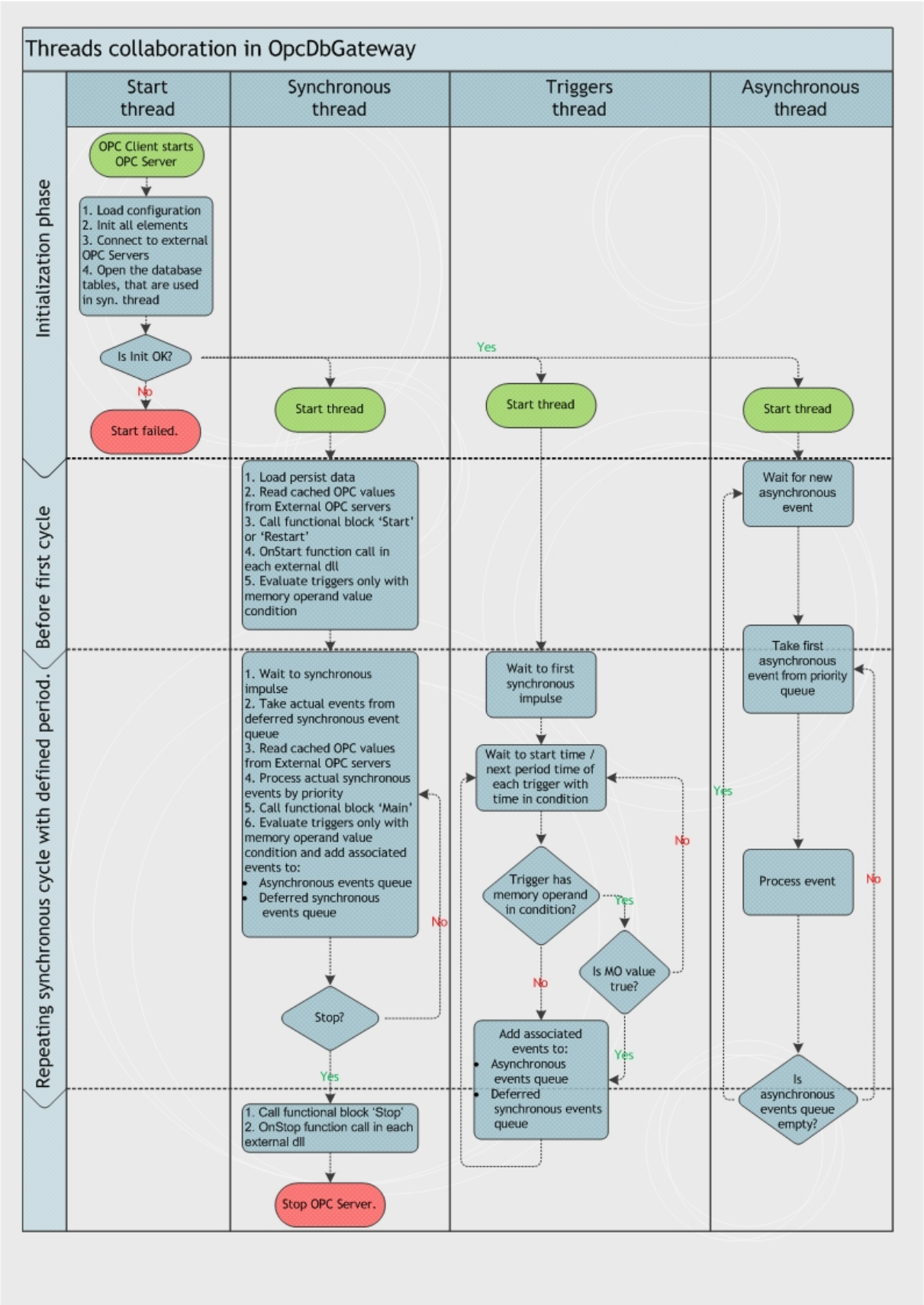


Figure: Trigger thread - timing in OpcDbGateway

Related articles
Events

6.5.7 Events

Events are one of the most important parts of the OpcDbGateway runtime. They exactly define what and when has to be done. Each event has assigned one trigger object. If the trigger is true then a defined action will be executed. Every event is trigerred using a trigger. One trigger can start altough more events.

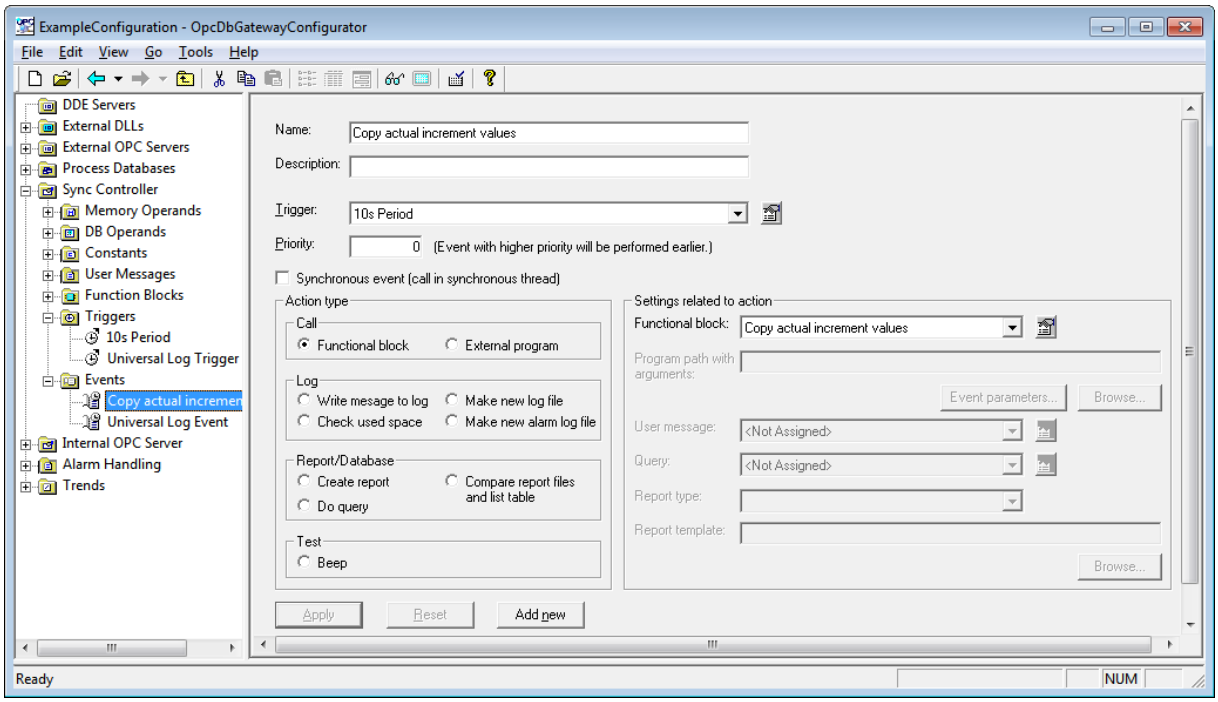


Figure: *Events dialog box*

Action type	**Functional block** - calls a function block and executes its commands **External program** - starts an external program **Create report** - generates a report Compare report files and report table **Do query** - executes a query **Make new log file** - closes current log file and creates a new one **Beep** - generates a beep sound (usefull for debugging) **Write msg to log** - writes an user message to a log file **Check used space** - checks the space used by logs, reports, process database and the size of used virtual memory **Make new alarm log file** - closes current alarmlog file and creates a new one **Beep** - for test purposes - watching execution of a concrete functional block by sound
Trigger	The trigger that triggers the event.
Function block	The function block which should be executed through the event **Call function block**
User message	The user message that should be written to a log file through the event **Write msg to log**.
Query	The SQL query that should be executed through the event **Do query** or **Generate report**
Priority	The priority defines the precedence of processing of events The events are evaluated one by one. If two or more events come at the same time, the event with higher priority is evaluated before other ones. 0 – lowest priority, 255 – highest priority
Program path	The executable program file (exe, script ...) which should be started
Report type	The type of the report that should be generated (TXT, CSV, HTML, XML)

Table 18 - Event parameters

Related articles

Triggers
Function blocks
Log files
Reports
Disk and memory monitor

6.5.7.1 Events - functionality

Events are two categories of events:

1. **Asynchronous events**
   - Call external program
   - Create report
   - Do query
   - Make new log
   - Beep
   - Write message to log
   - Check used space
   - Make new alarm log
   - Write message to alarm log
   - Call function block
2. **Synchronous events**
   - Call function block

Explaining the difference between these two categories of events

To explain the difference between these two categories of events we need to look deeper into the implementation of the event system.

There are three standalone threads running with a specific purpose:

- Trigger thread
- SyncController thread
- AsyncEvents thread

The Trigger thread periodicaly evaluates whether an event should be started or not. Each event has assigned one trigger which defines when the event should be started. If an event is triggered than it is inserted to a message queue. If it is a synchronous event then it is inserted to the queue **SyncQueue**. If it is an asynchronous event then it is inserted to the queue **AsyncQueue**. Also, The Trigger thread periodicaly sends a synchronization signal to the **SyncController** thread. The period of sending of the signal is equal to the period of the controller as defined in the Synchronous controller dialog.

The **AsyncEvents** thread is a simple message loop. It reads messages from the **AsyncQueue** and starts the requested actions one by one.

The **SyncController** thread is a little bit more complicated. When it receives a synchronization signal from the Trigger thread it reads the events from the queue **SyncQueue** and executes them. After all events are executed then the function block **Main** is executed. If the synchronization impuls comes and not all events are executed yet, then the Main function block is executed.

Each event has defined a priority. The priority defines how the events should be read from the message queues.
The rule is: The higher the priority is the earlier is the event read from the message queue.

The principle of processing of synchronous and asynchronous events is described as folowing
The synchronous events used the method of two priority queues.

The trigger thread prepares synchronous events to the priority queue 1. The **SyncController thread** reads and executes events from the priority queue 2 whenever it receives synchronization signal. The pointers to both queues are periodicaly exchanged at the beginning of each period. It can happen that in one period of the controller not all events are executed for time reasons. These not executed events are inserted to the queue1 before the pointers to the queues are exchanged.

There are two system variables that monitor the size of the message queues.

SyncQueueSize	The size of the message queue with synchronized events.
AsyncQueueSize	The size of the message queue with non-synchronized events.

Table 25 - System variables - s**ynchronous and asynchronous queue size**

Related articles

Triggers
System variables

6.5.8 Data persitence - functionality

The OpcDbGateway runtime implements very simple data persistence mechanism. When it is stopped either manually or on power failure it saves the contents of its **memory operands area** to the configuration database. On next start of the OpcDbGateway runtime, it will restore the memory from the configuration database. However, If the configuration database has been changed since the last start of the OpcDbGateway runtime then the memory is not restored.

The data persistence can be enabled or disabled in the Memory operands dialog.

6.6 Internal OPC Server

The internal OPC server folder inlcudes internal Address space of the OpcDbGateway, Conversions and Simulation signals.

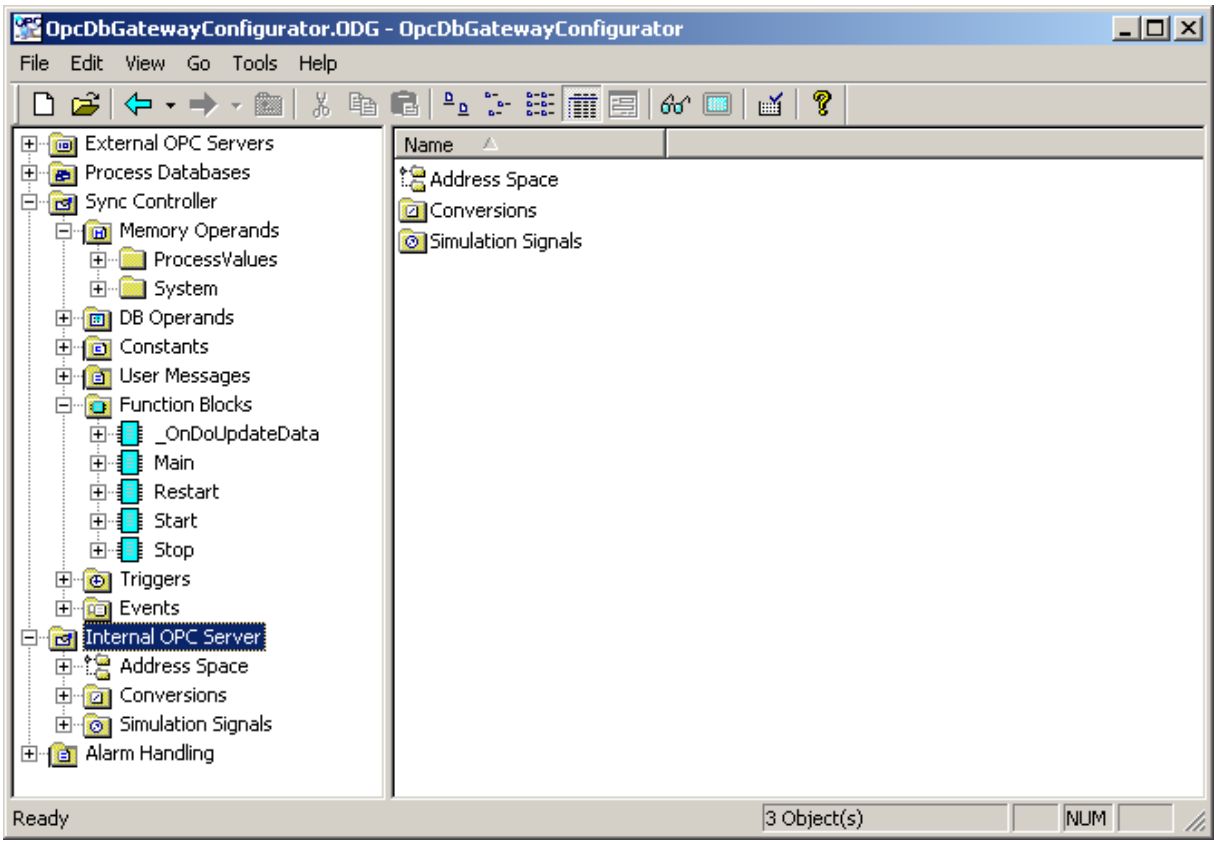


Figure 89: *Internal OPC Server*

6.6.1 Address space

Address space of an OPC server is a set of data items that any OPC client program can access.

The address space tree of the OpcDbGateway runtime has the following levels:

- Folders
- Data items

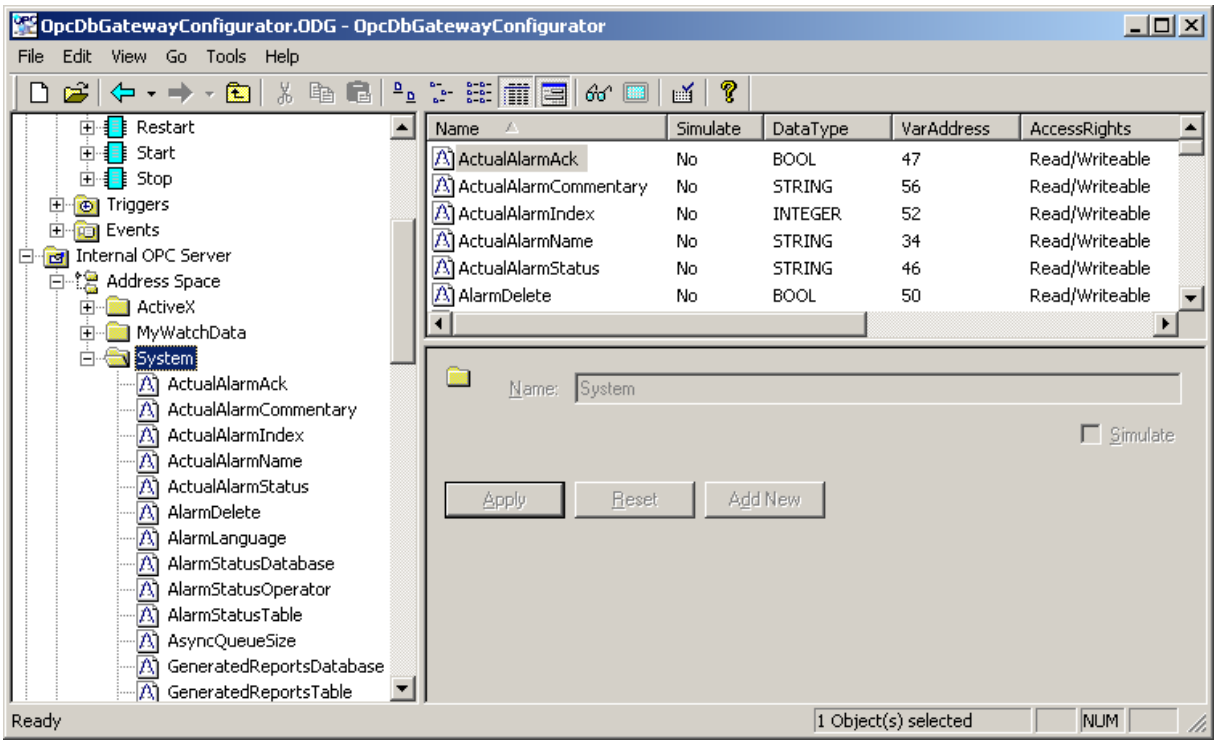


Figure 90: *Address space*

Folders
Folder is an object that can group items that logically belong together. You can design and use folders any way that is suitable to your application. Three levels of folders are supported in the OpcDbGateway runtime.

Data Items
Data Item represents one memory address of the OpcDbGateway runtime.
A symbolic name and description is associated with the data item. OPC Client can obtain the data item description.

New data item creation requires configuration of the following properties:

Access rights	The access rights of the data item readable readable/writeable
Data type	The data type of the dataitem
Simulate	if true the data item will be simulated
Simulation signal	the simulation signal
Manual	if true the data item will have constant value
Manual value	the manual value
Use conversion	if true, convert the value
Conversion	required conversion of the value
Memory address	defines the source of the data item - one address in the internal memory of the OpcDbGateway runtime

Table 19 - Dataitems parameters

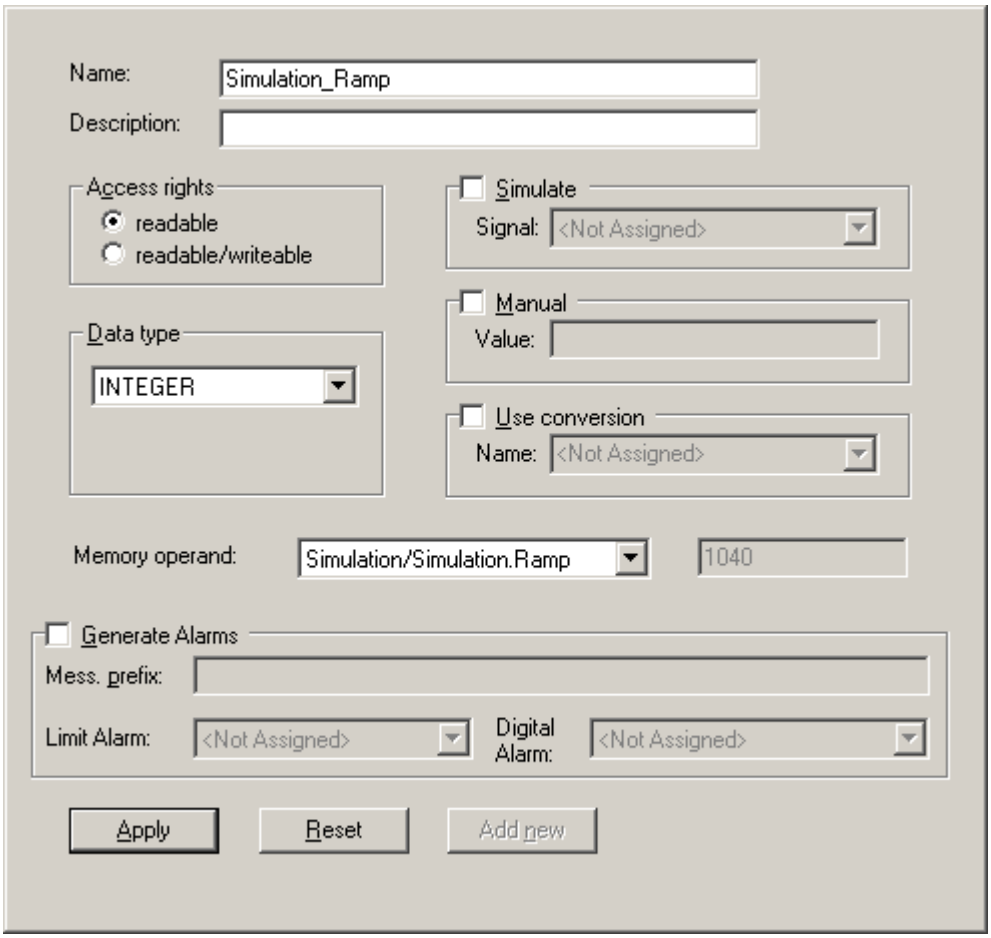


Figure 91: *Data item configuration*

Each data item could have associated alarm message. You have to define only the **Message prefix**, **Limit alarm** or **Digital Alarm**.

Related articles

Simulation signals
Conversions
OLE for Process Control (OPC)
Alarm system

6.6.2 Conversions

Using Conversions directory you can define your own conversions for converting of device data values from instrument range units (IR) to engineering units (EU).

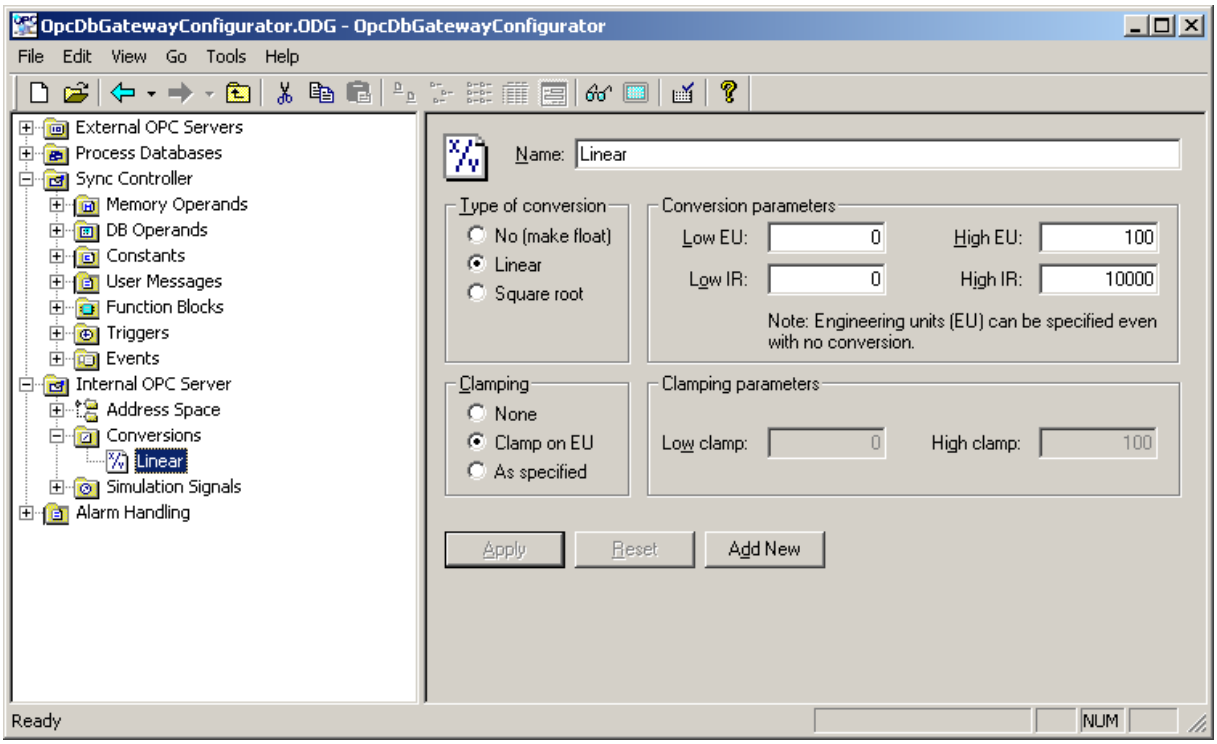


Figure 92: *Conversions*

To define a new conversion form you have to specify following properties:

Type of conversion	No conversion converts the data into float data type, but does not change the value itself. Linear or square root conversions keep a linear or square root relation between EU and IR.
Conversion parameters	Note that definition of range limits helps some client applications (e.g. when creating slider) and makes sense even when no conversion is specified.
Clamping	If clamping is on, the data value will be limited to its High clamp/EU value, when it exceeds the upper limit, and similarly with Low clamp parameter.

Table 20 - Conversion parameters

Related articles

Address space

6.6.3 Simulation signals

Using **simulation signals** directory, you can define your own simulation signals. The new signal can be created from predefined set of signals by changing of some of their parameters.

The predefined types of simulation signals are following:

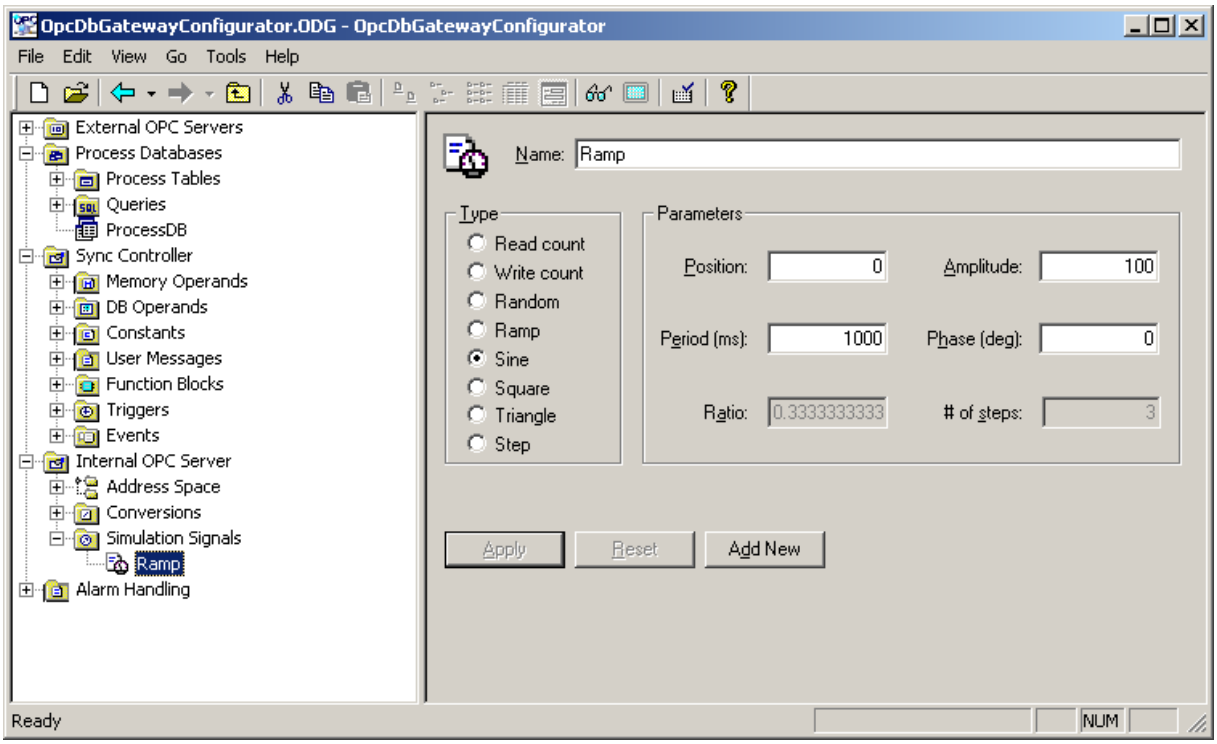


Figure 93: *Simulation signals*

Read count	Linear signal. The value is incremented by one every time when the item is read.
Write Count	Linear signal. The value is incremented when the item is written.
Random	Random signal. The value is generated within the Amplitude range starting with Position.
Ramp, Sine	Periodical signals. Their time behavior is influenced by Period and Phase parameters. Period specifies the signal frequency, while Phase moves the signal origin on the time axis.
Square, Triangle	Periodical signals. Their time behavior is influenced by Period, Phase and Ratio parameters. Ratio defines Triangle signal steepness, or Square signal H/L proportions.
Step	Periodical signal. Their time behavior is influenced by Period, Phase and # of steps parameters. # of steps parameter defines a number of steps the signal amplitude will be divided into.

Table 21 - Simulation signal parameters

Related articles

Address space

6.7 Alarm systems

Alarm systems enable to evaluate and store fulfilling or ending of alarm conditions as well as reaction of operating personnel on alarms.

Basic terms

Alarm source – it is a variable which indicates that an alarm has been activated
Quitting source – it is a variable which indicates that a operating personnel confirmed being informed about an alarm activiation

According to the type of alarm source variables alarms can be of the type:
- **Limit alarm** – for numeric alarm variables
- **Digital alarm** – for bool variables

Alarm severity – a number that is higher for more severe and smaller for less severe alarms.

OpcDbGateway offers **two database systems**:

1. **proprietary alarm system** based on values of memory operands - alarm sources and quitting source are memory operands. This system uses tables in process database to save not only actual status of alarm sources but also **alarms history**. It has also own visualisation within output view of the configurator.

2. alarm system based on OPC items implemented **according to the OPC AE standard.** It does not provide an alarm history saving in OpcDbGateway runtime application. (If required, it must be imlemented in OPC AE client application which is not delivered with OpcDbGateway.)

SAEAUT UNIVERSAL OPC Server offers only the 2nd type because it has not own process databases.

Related articles
Alarm system nfunctionalitty
Alarms - configuring
Alarm Viewer

6.7.1 Alarm system - functionality

The functionality of the **alarm system according to the OPC AE 1.01 standard** can be found in the specification maintained by OPC Foundation. Within OpcDbGateway, this alarm system is implemented as internal OPC DA client accessing OPC items in internal OPC server and providing OPC AE server interface. Programmer API to implement external OPC AE client applications is defined in mentioned OPC AE standard.

Proprietary alarm system is implemented according to the figure bellow.
It contains next parts:
- Alarm configuration within OpcDbGateway configurator – defining of memory operands for alarm and quitting sources, defining of alarm messages in 4 different languages, defining of alarm handling ways, defining if alarm messages have to be sent also trough SMS and short e-mails (SAEAUT SMS Service has to be installed to enable that). Alarm mesages can be parameterised by max. 5 memorz operands /message
- Alarm handling within OpcDbGateway runtime application
- Saving of status for every alarm source in database table of the process database with constant configured length. Table is created by default always when a new configuration is defined.

- Saving of alarm source history (when alarm was initiated, quitted, disappeared) in database table of the process database. Table is created by default always when a new configuration is defined.
- Alarm visualisation:
    - Within output view of the OpcDbGateway configurator
    - In alarm client applications (not delivered with OpcDbGateway) either using the same ActiveX as used in the output view or own implementation (as an OPC client) which uses OPC items to control functionality of the alarm handling in OpcDbGateway runtime and access process database to be able to display alarms status and history
- Logging of alarm history to alarm log files. Alarm log file is created by default when a new configuration is defined. To be able to limit file length, new alarm log files can be created using a configured event.

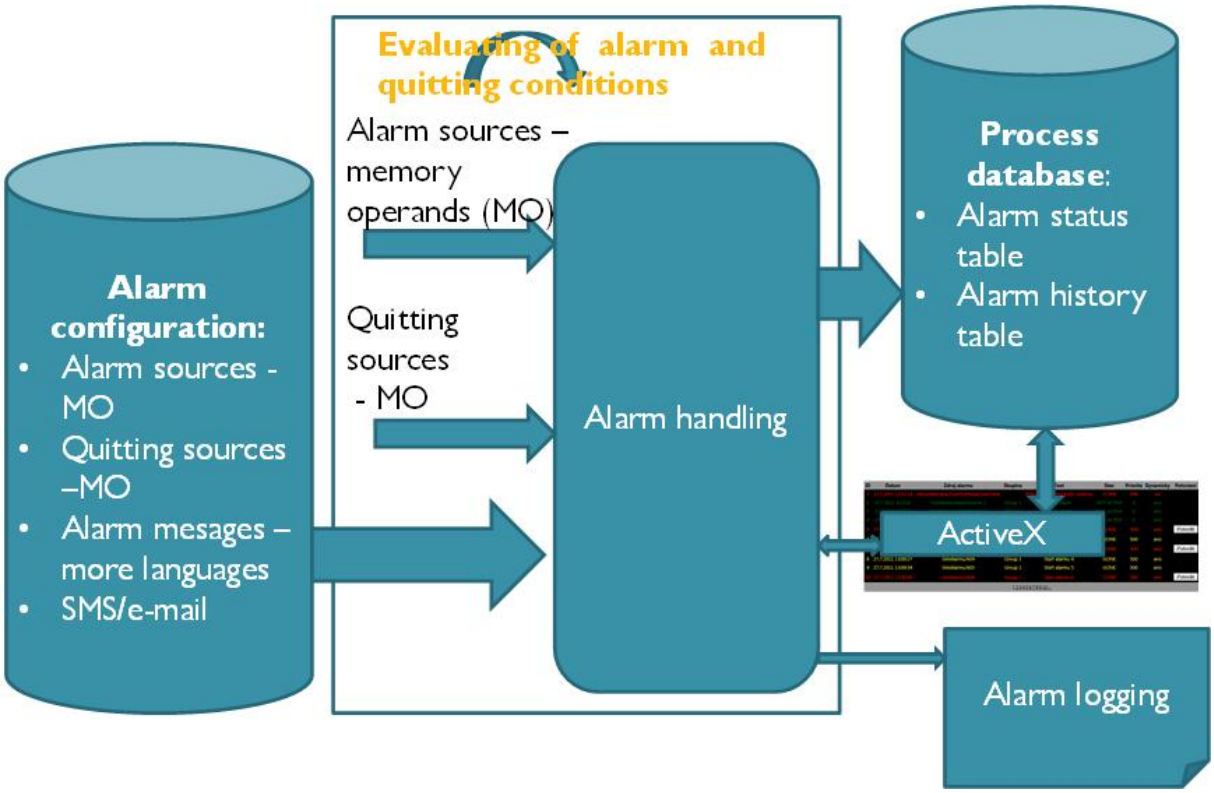


Figure: Proprietary alarm system in OpcDbGateway

Evaluation of the alarm conditions within proprietary alarm system is done in OpcDbGateway. The evaluation can be very sophisticated and so for alarm clients are all alarms presented as digital alarms.

Alarm client interfaces to proprietary alarm system

Client applications have to be able access following tables of the process database:

- AlarmStatus
- AlarmStatusHistory

To access process database, the connection string as defined in the configuration has to be used. However, to control alarm system in OpcDbGateway runtime would be risky through database driver. It should be provided by system variables used for this purpose. These system variables are by default mapped to the OPC items of the internal OPC server and so it is favourable to implement

own client applications as OPC clients. This way works also ActiveX - SaeOPCClient.ocx delivered with OpcDbGateway (placed in \Program Files\OpcDbgateway). System variables for alarm system control are memory operands (with indexes from 46 to 56) that can be mapped e.g. also to external dll's. Because of this, also alarm clients with other communication drivers can be used.
OPC items of the internal OPC server for alarm system control can be found in address space on the path System\Alarms.

Within SaeOPCClient.ocx one OPC item – ActualAlarmAck for quitting all alarms is used. Which concrete alarm is used is defined by OPC item ActualAlarmIndex. However, as alarm source and quitting source can be used whatever memory operand and so these variables need to be used in all types of alarm client applications.

Alarm Status Handling

The alarm can has one of four states:

- Inactive
- Come
- Acknowledged
- Gone

Inactive
Alarm conditions was not fulfilled yet.

Come
After fulfilling the alarm conditions.

Acknowledged
After quitting from some source.

Gone
When the alarm conditions are not more valid.

The transitions between these states are conditioned with next conditions:

AlarmDefinitionsMOP.Enable – if FALSE then transition conditions are evaluated, but none messages are written to log file. The default value is TRUE
AlarmDefinitionsMOP.ReturnToNormalEnable – if TRUE then transitions from states Come or Acknowledged to Inactive is enabled. If it is FALSE, then if the alarm condition is not more actual, transitions from the states Come or Acknowledged to Gone are executed. The default value is FALSE.
AlarmDefinitionsMOP. AckRequired – if FALSE then acknowledge is executed by server, it means that from status Inactive is executed transition direct to the status Acknowledged

Saving alarm Inquiries

In configuration databank have to be configured a table for process databank AlarmStatus, where the count of the rows match count of the alarm sources. This table has structure according to the Fig. 2.The table contains inquiries about all alarm sources and their states. This table can be used for example for viewing alarm states in alarm client.

Field Name	Data Type	Description
ID	Number	
Time_	Date/Time	
Date_	Date/Time	
Message	Text	
Status	Text	
Priority	Number	
Group_	Text	
Deleted	Yes/No	
Comentar	Text	
Color_	Number	
StatusInt	Number	

Figure 46: *The table AlarmStatus*

The changing of the alarm states will be logged to the alarm log file. For every alarm source can be defined up to 2 messages one for the event when alarm condition is changed to fulfilled (AlarmDefinitionsMOP.AlarmMessageID), and one when alarm condition is changed to not fulfilled (AlarmDefinitionsMOP.ReturnToNormalMessageID).
These messages are saved to the table AlarmStatus. As the alarm messages can be saved in the table AlarmMessages in maximal 4 languages, it will be possible to change the actual language of the messages trough system variable of the runtime.

In addition inquiry about every transition between states for each alarm is saved to log file in such a way:

```
I 2002-03-12 10:14:00.218  Alarm name: nnnnnn…,Group: gggggg…, Previous
state: INACTIVE, New state: ACTIVATED
```

To use the possibility to view the filtered alarm logging, it can be useful save this inquiries to the databank table too.

Related articles

Static alarms
Dynamic alarms

6.7.1.1 Static alarms

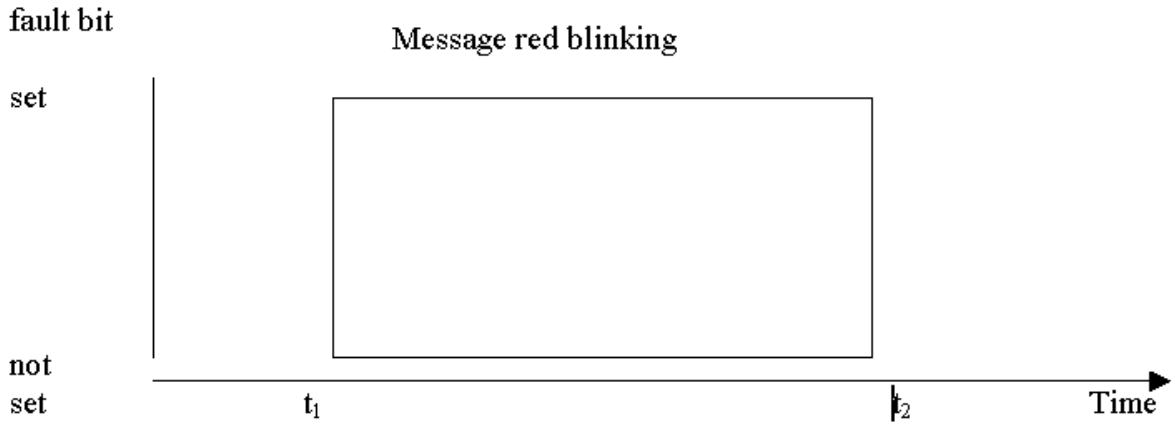


Figure 47: *Static alam*

time point t_1: fault bit set
time point t_2: fault bit reset

Displaying of the alarm message at the time t_1: Fault bit set

Channel nr..:	Message	Status	Time
1	Message Nr.1 (red blinking)	come	t_1

Displaying of the alarm message at the time t_2: Fault bit reset

Channel nr..:	Message	Status	Time
1	Message Nr.1 (green)	gone	t_2

Displaying of the alarm message after the time t_2

Channel nr..:	Message	Status	Time
1	No message (green)	not active	> t_2

Related articles
Dynamic alarms

6.7.1.2 Dynamic alarms

The fault bit is reset before the alarm isacknowledge

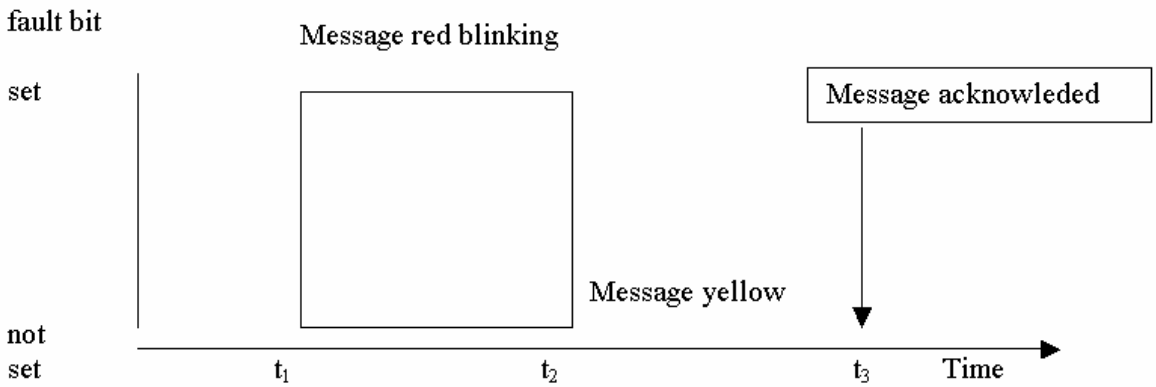


Figure 48: *The fault bit is reset before the alarm is acknowledged*

time point t_1: fault bit set
time point t_2: fault bit reset
time point t_3: alarm acknowledged

Displaying of the alarm message at the time t_1: Fault bit set

Channelnr.:	Message	Status	Time
1	Message Nr.1 (red blinking)	come	t_1

Displaying of the alarm message at the time t_2: Fault bit reset

Channelnr.:	Message	Status	Time
1	Message Nr.1 (yellow)	gone	t_2

Displaying of the alarm message at the time t_3: Alarm acknowledged

Channelnr.:	Message	Status	Time
1	Message Nr. 1 (green)	acknowledged	t_3

Displaying of the alarm message after the time t_3

Channelnr.:	Message	Status	Time
1	No message (green)	not active	> t_3

Fault bit is reset after the alarm is acknowledged

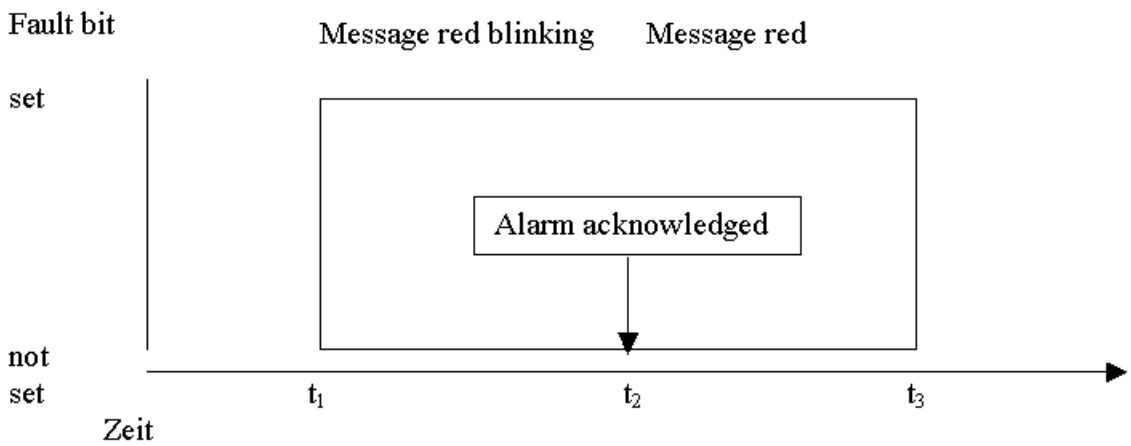


Figure 49: *The fault bit is reset after the alarm is acknowledged*

time point t_1: fault bit set
time point t_2: alarm acknowledged
time point t_3: fault bit reset

Displaying of the alarm message at the time t_1: Fault bit set

Channelnr.:	Message	Status	Time
1	Message Nr.1 (red blinking)	come	t_1

Displaying of the alarm message at the time t_3: Alarm acknowledged

Channelnr.:	Message	Status	Time
1	Message Nr.1 (red)	acknowledged	t_2

Displaying of the alarm message at the time t_3: Fault bit reset

Channelnr.:	Message	Status	Time
1	Message Nr.1 (green)	gone	t_3

Displaying of the alarm message after the time t_3

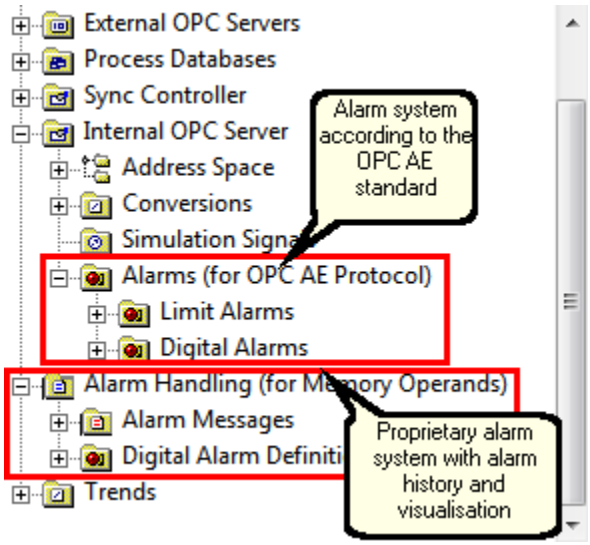
Channelnr.:	Message	Status	Time
1	No message (green)	not active	> t_3

Related articles

Static alarms

6.7.2 Alarms - configuring

Items configured for alarm system according to the OPC AE specification are in the configurator tree view placed under item Internal OPC server because this system works only with items mapped to the address space of the internal OPC server.
Items configured for proprietary alarm system are in the configurator tree view placed under item Alarm Handling (for Memory Operands)



1.
Figure: Alarm systems in tree view of the configurator.

6.7.2.1 Proprietary alarm system

6.7.2.1.1 Alarm messages

Alarm messages are messages that can be written to an alarm log file of the OpcDbGateway runtime. Alarm messages can be parametrized. Parameters provide easy way of adding additional information to the message according to the current value of one or more memory operands.

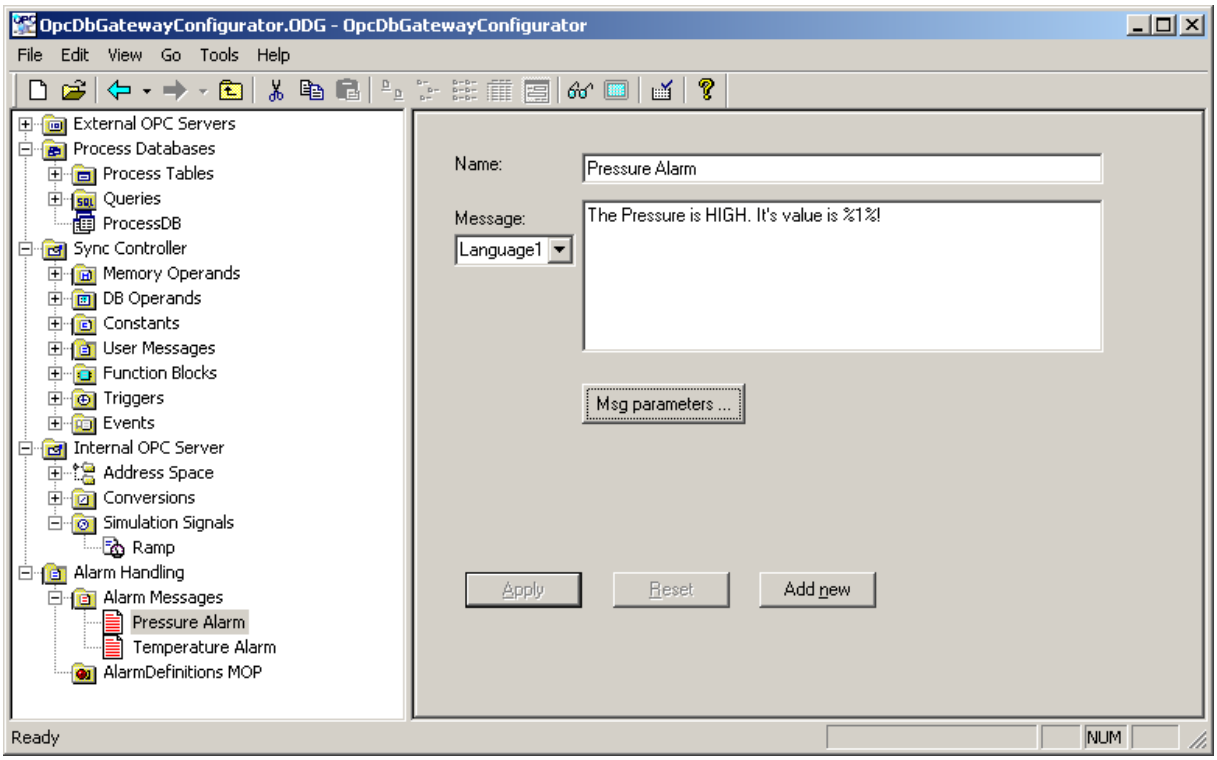


Figure 95: *Alarm messages*

An alarm message has following parameters:

Language	The message text can be written in four different languages.
MessageText	The text of the message.

Table 22 - Alarm message configuration parameters

Parametrazition of Alarms
The message can contain up to 6 parameters - %1%, %2%, ..., %6%. Each parameter represents one memory operand.
Before the message is written to the alarm log file the current value of the memory operand is inserted into defined position of the message string. Afterwards is the message written to the log.

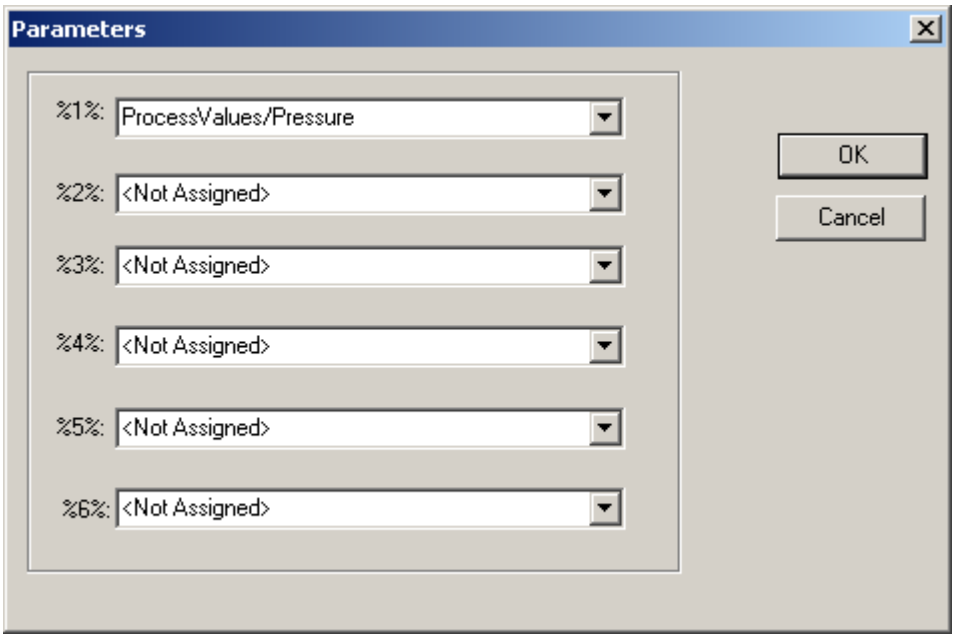


Figure 96: *Alarm message text parameters*

Related articles

Alarm definitions MOP

6.7.2.1.2 Alarm definitions MOP

Alarm definitions MOP define the behaviour of alarms - the value activate the alarm, the alarm message, the severity of the alarm or the acknowledge memory operand. Each alarm definition can be assigned to one or more memory operands.
The alarm messages are written to a log file. Also, the current state of all defined alarms is kept in the table AlarmStatus.

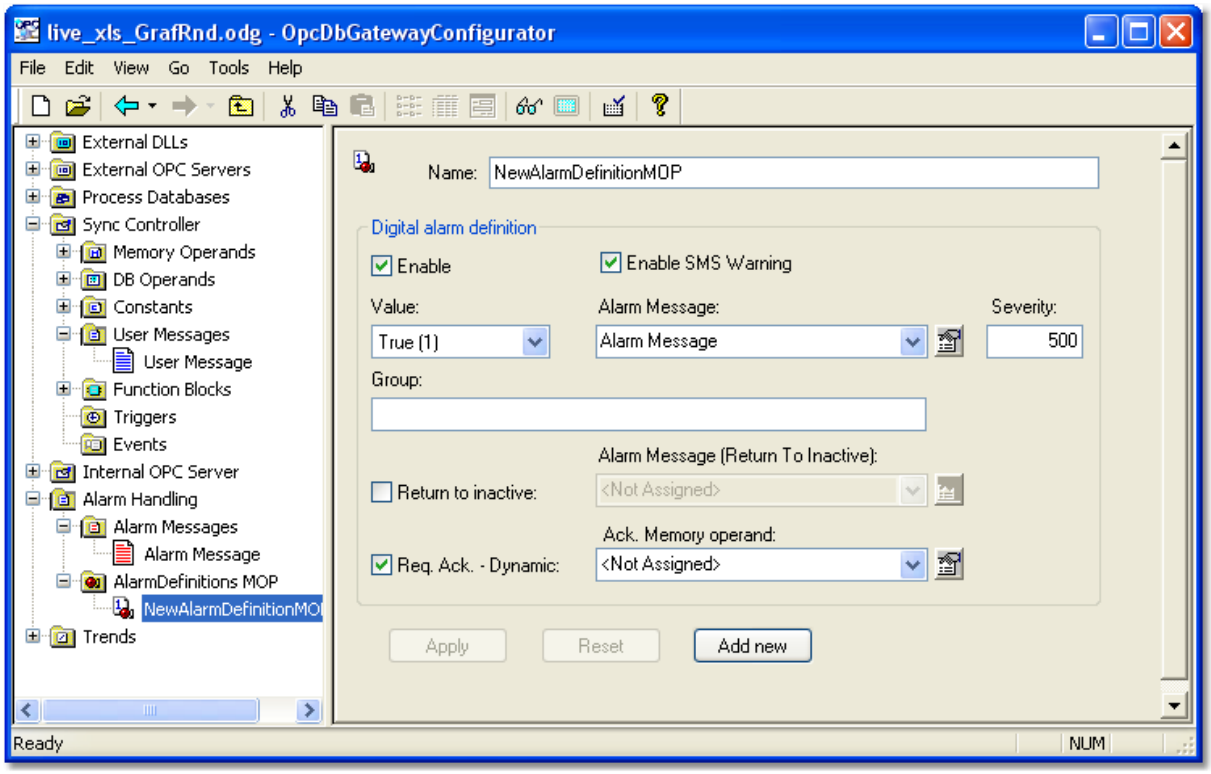
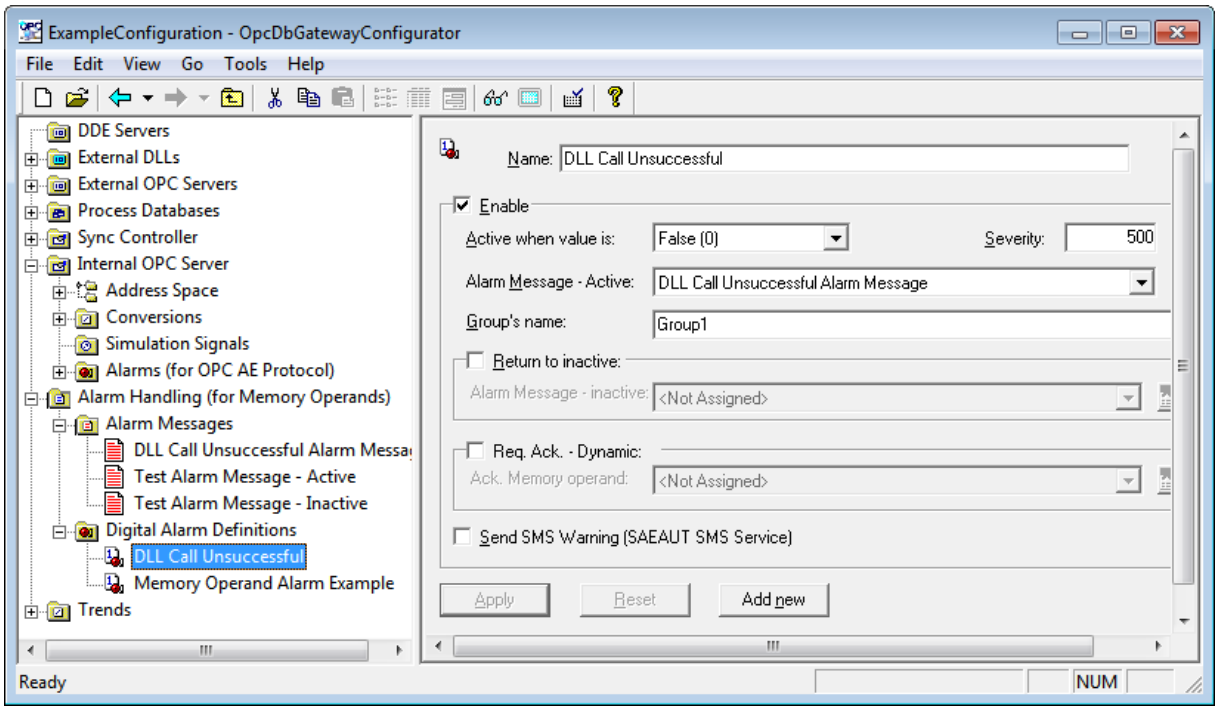


Figure 1: *Alarm definitions MOP*

Figure 3: *Limit alarm definitions*

Enable	If true the evaluation of the alarm is enabled
Value	Alarm value of the memory operand
Alarm Message	The message which is displayed when the alarm conditions are fullfilled
Severity	The urgency of the alarm 1 lowest severity 1000 highest severity
Return to inactive	If true the alarm can have status Gone without acknowledging it
Alarm Message (Return To Inactive)	Info message displayed when the alarm became inactive
Req. Ack - Dynamic	If true then the alarm must be acknowledged through another memory operand If the acknowledge is not required then acknowledge is made automaticaly by the OpcDbGateway runtime
Ack. Memory operand	The memory operand for acknowledging of the alarm
Enable SMS Warning	If true the SMS warning is sent.

Table 1 - Alarm definition parameters



Alarm messages
Alarm system
Memory operands

6.7.2.2 Alarm system (OPC AE standard)

Alarm system based on OPC items is functioning according to the OPC AE 1.1 standard

In case that an OPC item of the internal OPC server has t o have an alarm definition it is necessary chose this posibility in OPC item dialog box and chose proper alarm definition of the type limt alarm or digital alarm. They have to be configured in advance.

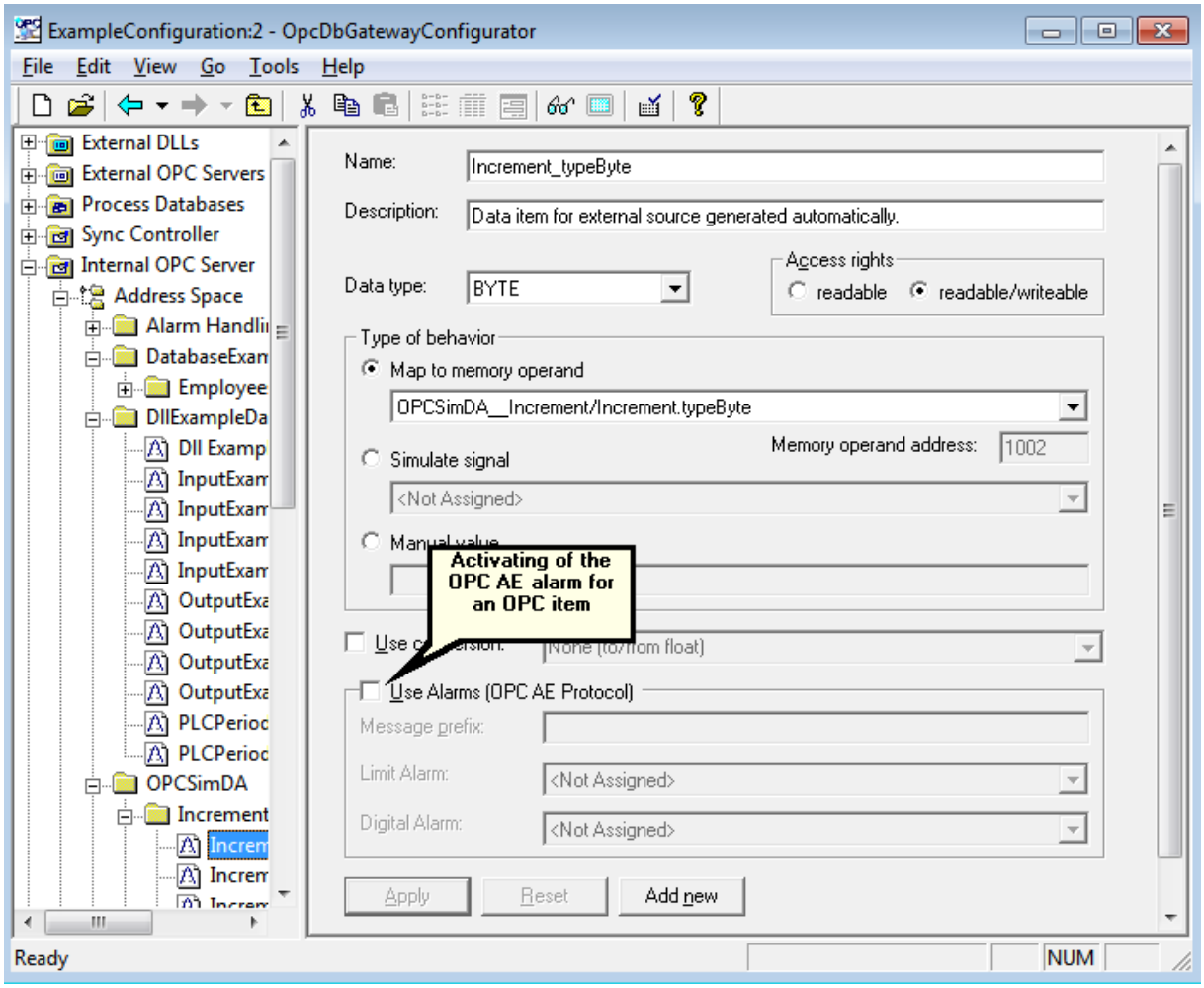


Figure: *Activating of OPC AE alarm.*

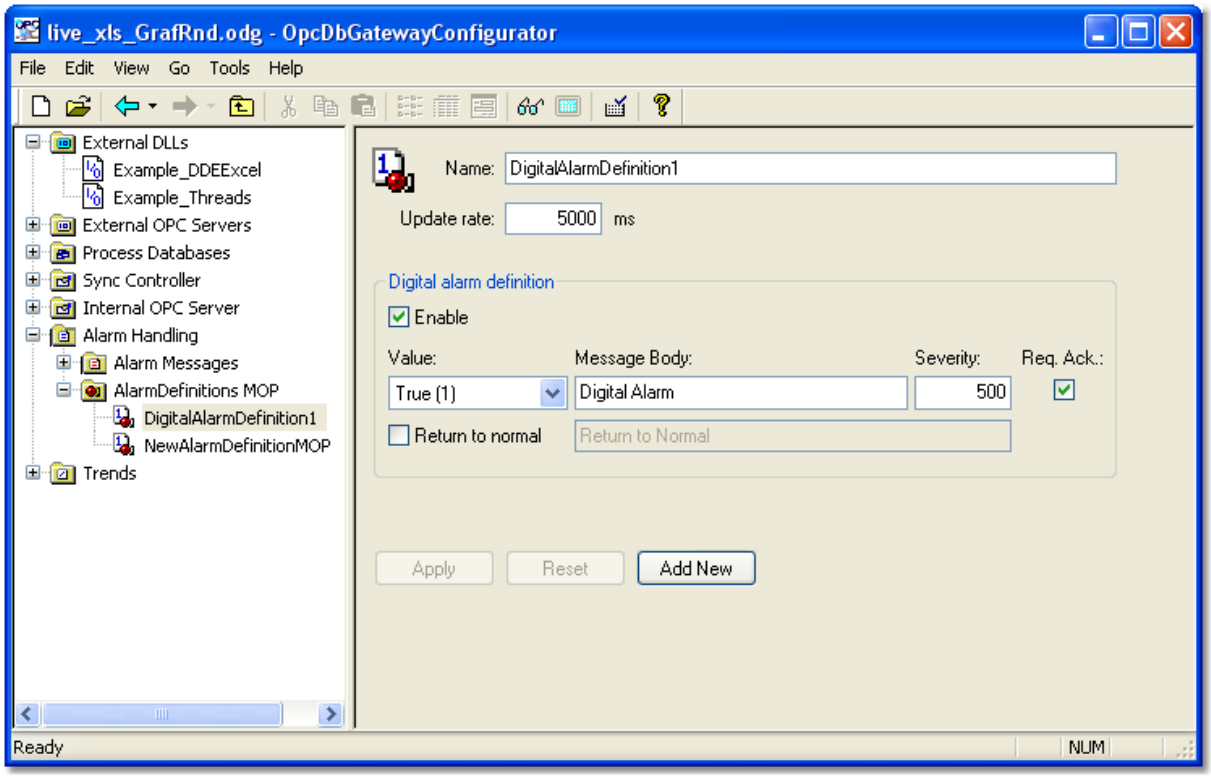


Figure: *Digital alarm definitions*

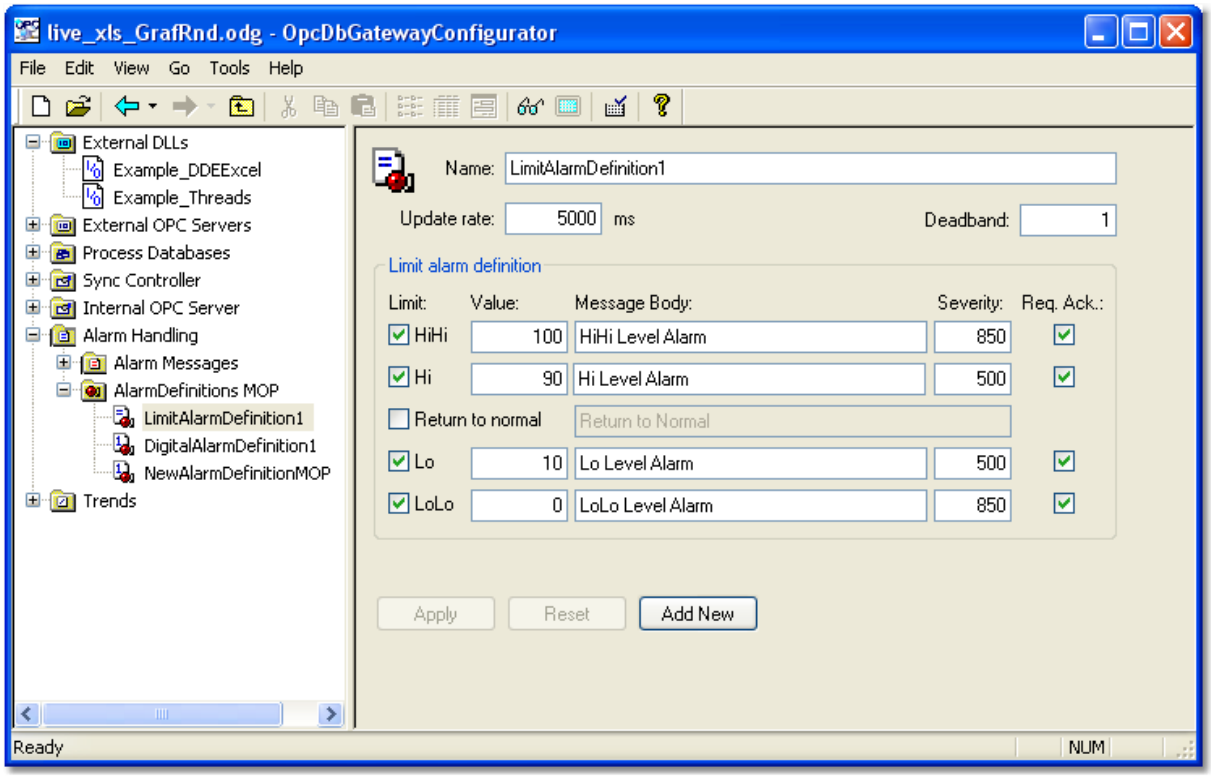


Figure: *Limit alarm definitions*

Enable	If true the evaluation of the alarm is enabled
Value	Alarm value of the memory operand
Alarm Message	The message which is displayed when the alarm conditions are fullfilled
Severity	The urgency of the alarm 1 lowest severity 1000 highest severity
Return to inactive	If true the alarm can have status Gone without acknowledging it
Alarm Message (Return To Inactive)	Info message displayed when the alarm became inactive
Req. Ack - Dynamic	If true then the alarm must be acknowledged through another memory operand If the acknowledge is not required then acknowledge is made automaticaly by the OpcDbGateway runtime
Ack. Memory operand	The memory operand for acknowledging of the alarm
Enable SMS Warning	If true the SMS warning is sent.

Table 1 - Alarm definition parameters

Related articles
Alarm messages
Alarm system
Memory operands

6.8 Historic trends - What's Historic Trend

OpcDbGateway can collect and process data from various kinds of OPC Servers. But sometimes it is necessary not only to process these data, but also to save them for future statistical processing. To provide this, a historic trend tool was added to OpcDbGateway. Historic Trends allows to store memory operands into a database with a given saving period. It means that a new table is created in the database with columns for every selected memory operand (and for ID and timestamp too). At every time period the values of these memory operands are saved into the table together with the current timestamp (a new row in the table is added). The backup mechanism of the trend was added too. It allows, with the given period, to make a copy of trend's database into a XML, HTML or CSV file. These files may be used as a backup copy of the database (CSV and XML files can be exported back to a mdb file), or as a report files, for the specific time or period. Everytime a new file is created and stored in the directory ..\Data\REPORT\"File type"\"Backup's event name"\.. (see Figure 1).

More details about Historic trends it is possible see in the following articles:

- Historic Trends Wizard
- Trend View

Name	Ext	Size	↓Date	Attr
[..]		<DIR>	16.11.2006 10:00	----
TrendTest_Backup_Event_061116_100030	HTML	3 276	16.11.2006 10:00	-a--
TrendTest_Backup_Event_061116_100000	HTML	3 161	16.11.2006 10:00	-a--
TrendTest_Backup_Event_061116_095930	HTML	3 156	16.11.2006 09:59	-a--
TrendTest_Backup_Event_061116_095900	HTML	3 175	16.11.2006 09:59	-a--
TrendTest_Backup_Event_061116_095830	HTML	3 247	16.11.2006 09:58	-a--
TrendTest_Backup_Event_061116_095800	HTML	3 168	16.11.2006 09:58	-a--
TrendTest_Backup_Event_061116_095730	HTML	3 153	16.11.2006 09:57	-a--
TrendTest_Backup_Event_061116_095700	HTML	3 172	16.11.2006 09:57	-a--
TrendTest_Backup_Event_061116_095630	HTML	3 242	16.11.2006 09:56	-a--
TrendTest_Backup_Event_061116_095600	HTML	3 179	16.11.2006 09:56	-a--

Figure 100: *Example of backup files*

Related articles

Historic Trends Wizard
Trend View

6.8.1 Historic Trends Wizard

Historic trends are created by using a wizard. It allows a simple and fast creation, the only parameters that the user has to insert are:

- Trend's name
- Database
- Saving period
- Memory operands
- Backup parameters (not necessary)

Historic trends mechanism use only items and tools provided by OpcDbGateway (events, triggers, …). User can create it's own trend mechanism also with using queries, triggers and events, but wizard make this process more easily. The items created by wizard can be changed manually later.

The wizard consists of four pages:

- Page 1 – Introduction and description
- Page 2 – Memory Operands selection
- Page 3 – Main trend's parameters
- Page 4 – Backup settings

How to start wizard

There are two ways how to start a wizard:

1. From main menu, choose **Tools** → **Wizard** → **Create Historic Trends**

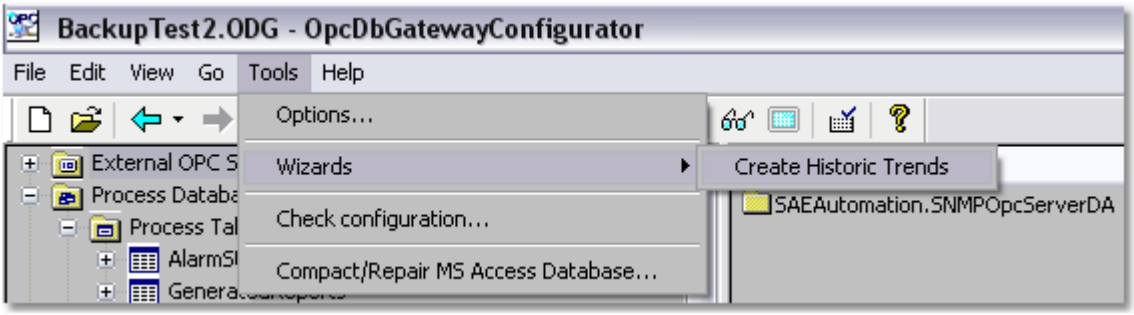


Figure 101: *Starting trend's wizard from main menu*

2. From the tree view, right click on trend's group, and choose **New → Trends**

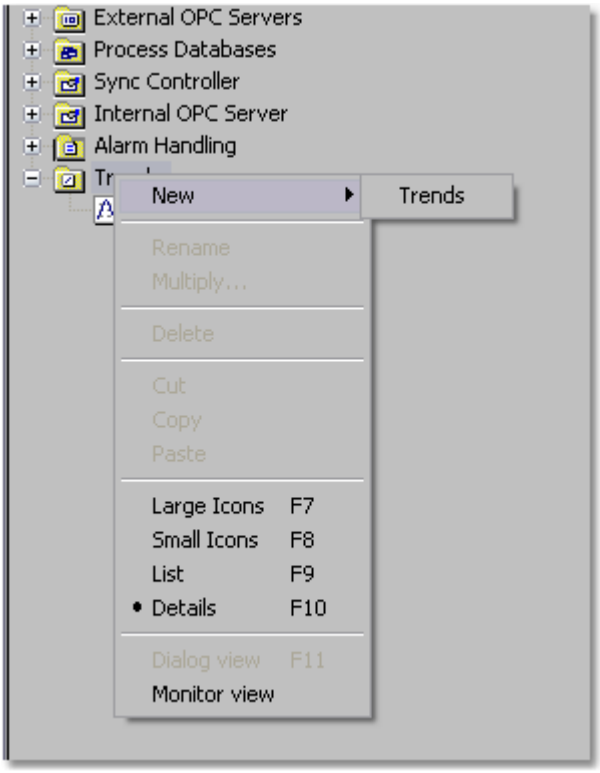


Figure 70: *Starting trend's wizard from local menu*

Page 1 – Introduction and description

The first wizard page contains only information about:

- what's historic trend
- what is created using this wizard
- what's necessary before starting a wizard

The two only things, that are not created by this wizard, and must be done before manually by the user are:

- The database must be created
- Memory operands must be created and mapped

If the user is not sure, how to make this, the buttons ? are placed on this page, that will open the proper page of OpcDbGateway help.

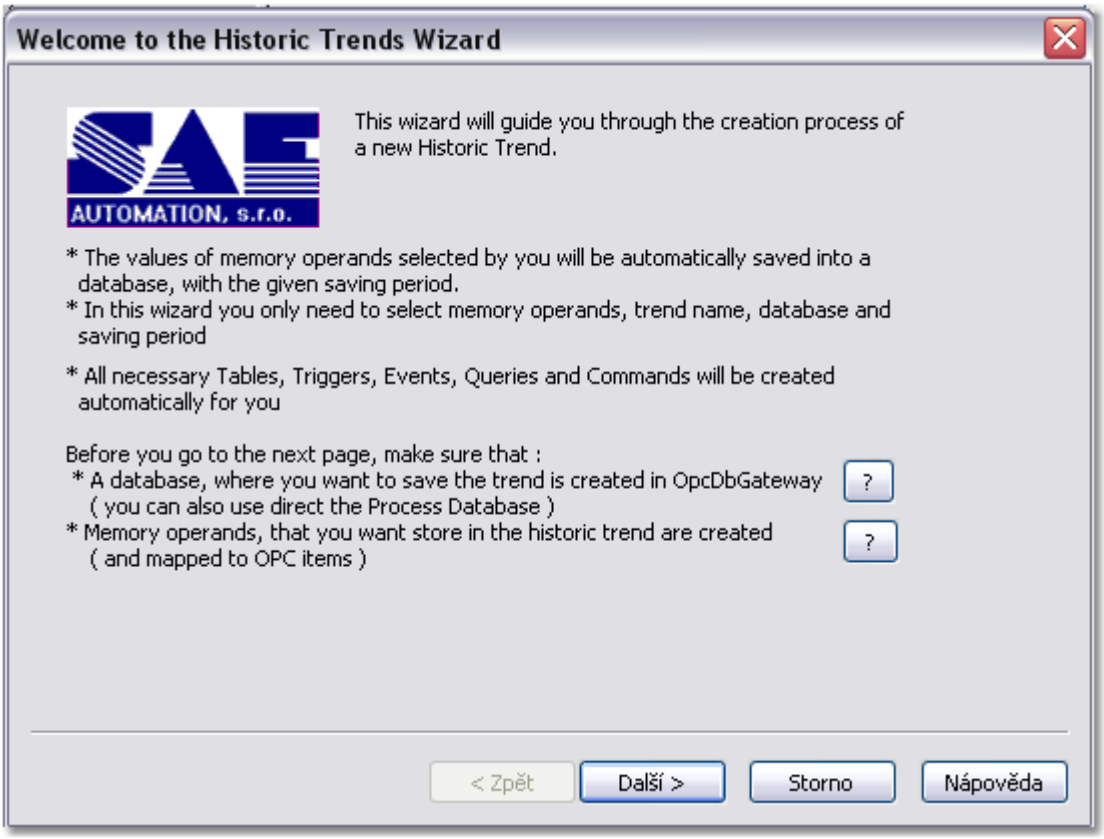


Figure 71: *First wizard page - Basic information*

Page 2 – Memory Operands selection

The second page is used to select memory operands that will be saved into a database. The combo box on the left contains the list of all memory operands, on the right there are selected memory operands. Maximal number of selected operands is 6. Use double click to remove a memory operand from the list.

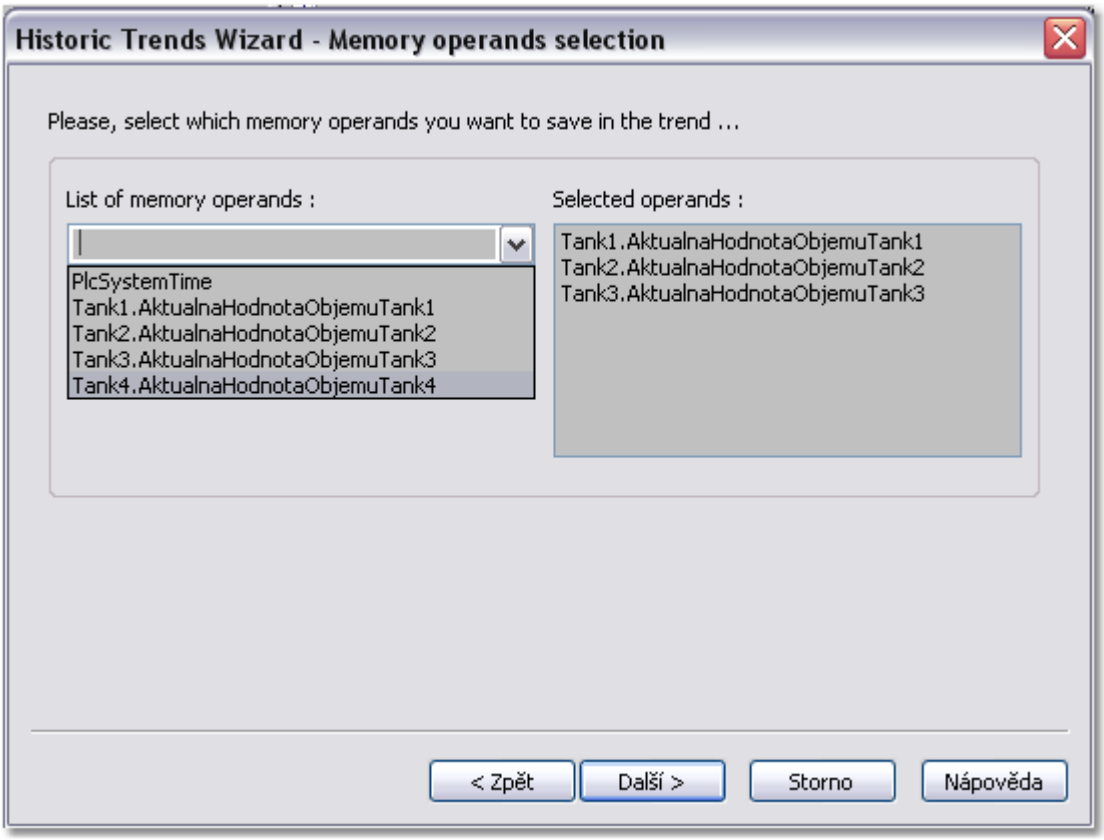


Figure 72: *Second wizard page - Memory operands*

Page 3 – Main trend's parameters

In this page it is necessary to set three main trend's parameters:

- Trends name (the created table will have the same name)
- Database, where the table will be located
- Saving period (in milliseconds)

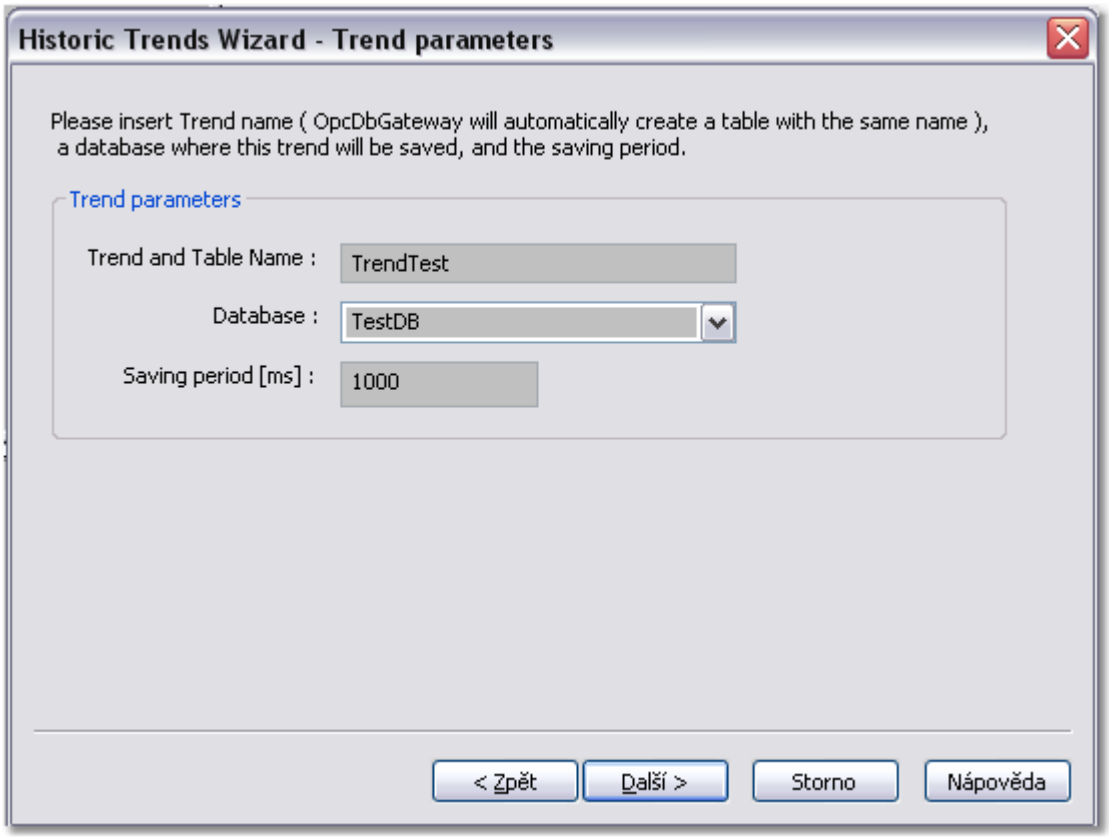


Figure 73: *Third wizard page - Main parameters*

Page 4 – Backup settings

It is the last page of the wizard and is used to set a backup of the trend. Backup is not necessary, you must check the checkbox "Use Backup", if you want to use it. There are only two things to set on this page:

- Backup period – user can set backup period in days, or in number of iterations (number of records in the table)
- File type – the type of backup file. There are three supported types: XML, HTML and CSV

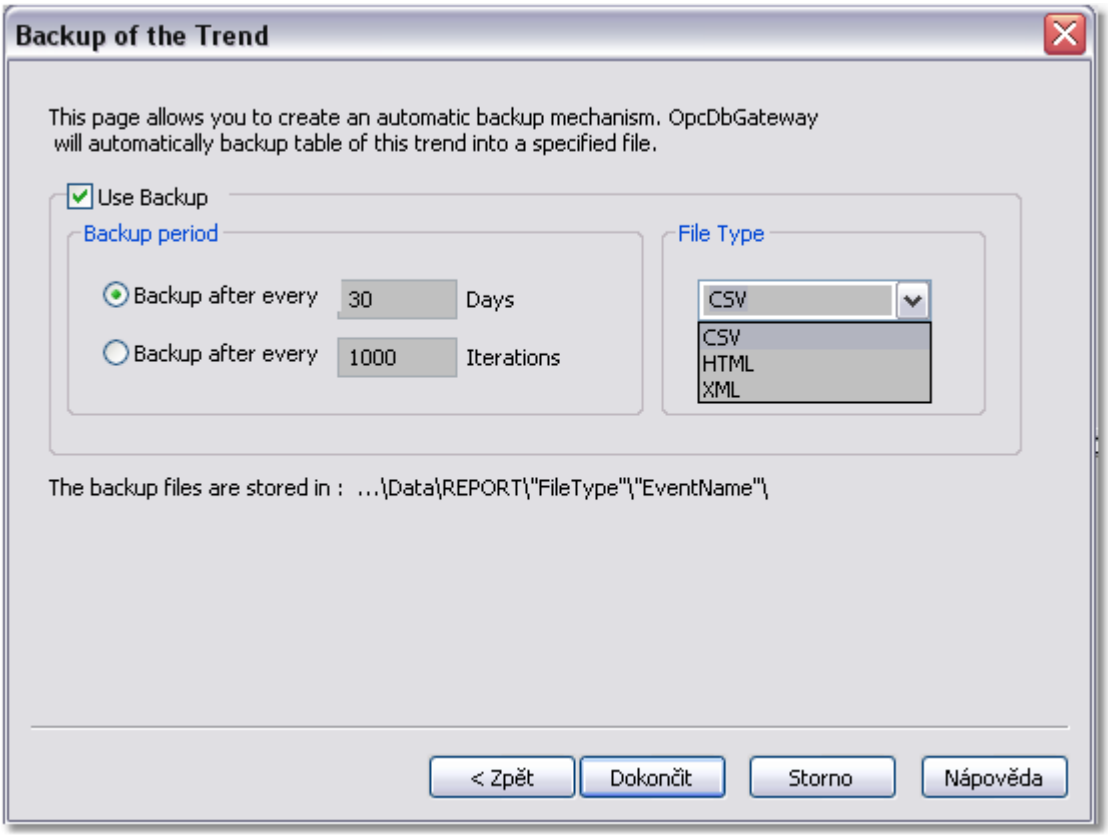


Figure 74: *Last wizard page - Backup settings*

Related articles

Historic trends - What's Historic Trend
Trend View

6.8.2 Trend View

Trend is only an informative view about the selected trend. It contains all information about the trend, backup, and all OpcDbGateway items used to create this trend.

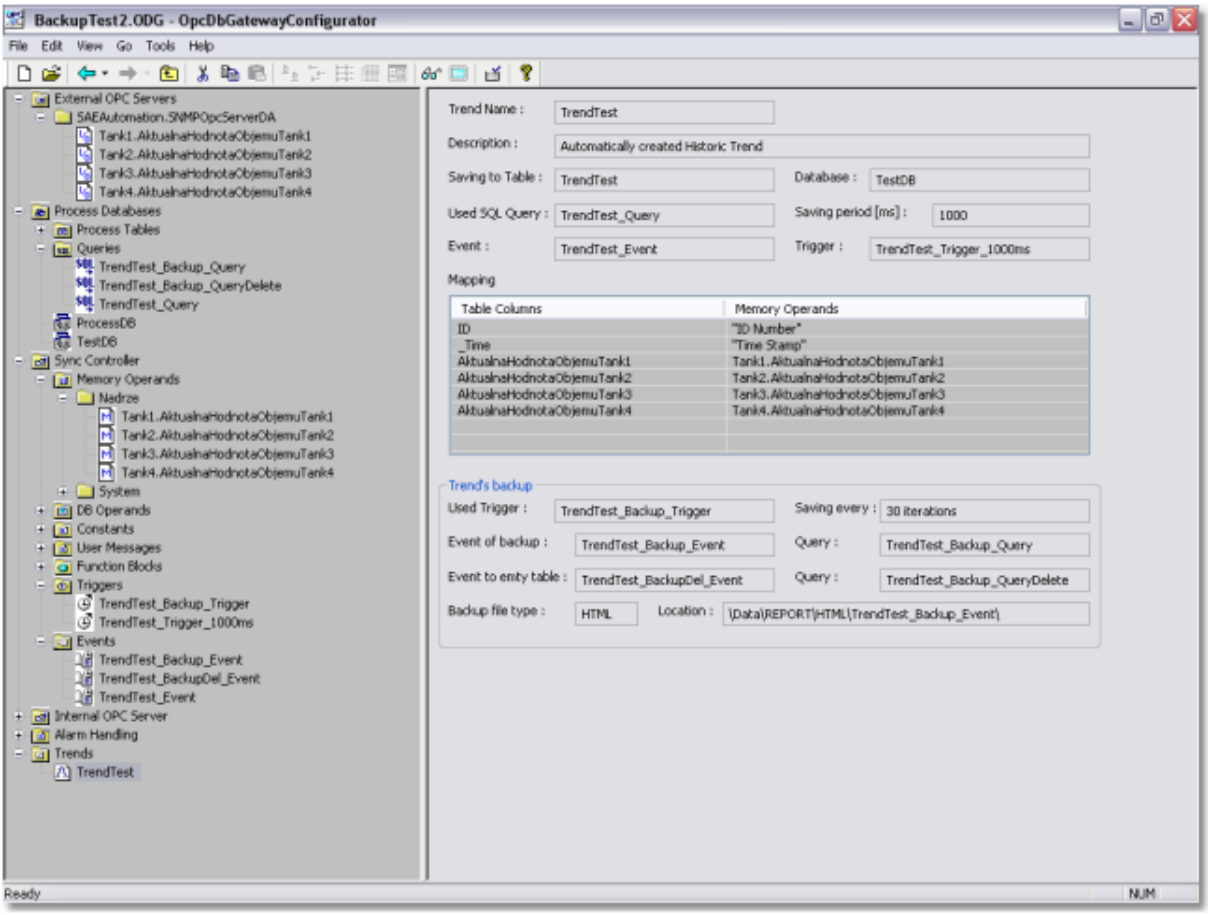


Figure 102: *Trend's view*

To create the functionality of the trend (and backup), many OpcDbGateway items are created automatically. For a trend with the name "TrendName", following items are created:

Trends
TrendName
New trend record is inserted, it contains the information about used events.

Tables
TrendName
The table, where the values are stored. It has the same name as the trend.

Columns
ID
ID number of the record

_Time
Time stamp

Operand1 … OperandN
One column for every memory operand. The columns have the same names as memory operands (just not including folder's path).

Triggers

TrendName_Trigger_Nms
Trigger used by the Trend

TrendName_Backup_Trigger
Trigger used by backup. Its period is in days, or in milliseconds (when user select backup period as number of iterations, in this case is period the number of iterations multiple by the trend's trigger time).

Events
TrendName_Event
Event used by trend, it is called by TrendName_Trigger_Nms, and it execute query TrendName_Query.

TrendName_Backup_Event
This event is used to make backup file. It's called by TrendName_Backup_Trigger, it is switch to "Create report" mode, and the query to select the values from the table is TrendName_Backup_Query

TrendName_BackupDel_Event
It is used to empty trend's table, after a backup was made into a backup file. To do that is used query TrendName_Backup_QueryDelete. It is also called by trigger TrendName_Backup_Trigger.

Queries
TrendName_Query
This query makes a trend's record into table, saves ID, timestamp and values of memory operands.

TrendName_Backup_Query
This query is used by the backup mechanism, called by TrendName_Backup_Event. It just select all records from the table.

TrendName_Backup_QueryDelete
This query is used to empty trend's table.

Related articles
Historic trends - What's Historic Trend
Historic Trends Wizard

OpcDbGateway and SAEAUT Universal OPC Server

Part VII

7 System variables

System variables are variables used internally by the OpcDbGateway runtime. They provide following information:

- current status of the server
- size of the space used by log files and reports
- current system time
- current power status
- time and speed meassurements

The first 100 addresses of the OpcDbGateway runtime are reserved for the system variables.

For each system variable exists one data item in the folder **System** in the internal OPC server's address space.
Any opc client can read information from these data items. Also, the OPC ActiveX controls such as ReportViewer or AlarmViewer use the system variables to retrieve information about reports and alarms.

It is possible to use system variables in function blocks e.g. the current system time. We can define for each system variable one memory operand.

The list of all system variables is in the following table:

Memory address	System variable	Data type	Access rights
1	SyncQueueSize	LONG	Read only
2	AsyncQueueSize	LONG	Read only
3	TimeLog	BOOL	Read / write
4	TraceLog	BOOL	Read / write
5	Period	LONG	Read / write
6	PeriodMeassured	LONG	Read only
7	PeriodCounter	LONG	Read only
8	Cycle	LONG	Read only
9	CycleMax	LONG	Read only
10	CycleMin	LONG	Read only
11	CycleLimitCounter	LONG	Read only
12	SystemTime	DOUBLE	Read only
13	CycleLimitPercentage	DOUBLE	Read only
14	UsedLogsSpace	DOUBLE	Read only
15	UsedReportsSpace	DOUBLE	Read only
16	AvailMemory	DOUBLE	Read only
17	UsedProcessDBSpace	DOUBLE	Read only
18	LogsFull	BOOL	Read only
19	ReportsFull	BOOL	Read only
20	VirtualMemoryFull	BOOL	Read only
21	ProcessDBFull	BOOL	Read only
22	ACLineStatus	INT	Read only
23	BatteryFlag	INT	Read only
24	BatteryLifePercent	INT	Read only
25	BatteryLifeTime	LONG	Read only
26	BatteryFullLifeTime	LONG	Read only
27	Stop	BOOL	Read / write
28	Start	BOOL	Read / write
29	Status	LONG	Read only
30	StatusDescription	STRING	Read only
31	StopReason	LONG	Read only
32	StopReasonDescription	STRING	Read only
33	LastErrorDescription	STRING	Read only
34	ActualAlarmName	STRING	Read / write
35	GeneratedReportsDatabase	STRING	Read only
36	GeneretedReportsTable	STRING	Read only
37	NotPrintedReportsDatabase	STRING	Read only
38	NotPrintedReportsTable	STRING	Read only
39	NewGeneratedReport	STRING	Read only
40	ReportOperator	STRING	Read / write
41	ReportName	STRING	Read / write
42	PrintCommand	BOOL	Read / write
43	PrintStatus	BOOL	Read / write
44	PrintConfirm	BOOL	Read / write
46	ActuaAlarmStatus	STRING	Read / write
47	ActualAlarmAck	BOOL	Read / write
48	NextAlarm	BOOL	Read / write
49	PreviousAlarm	BOOL	Read / write
50	AlarmDelete	BOOL	Read / write

Table 24 - System variables

7.1 Disk and memory monitor

The OpcDbGateway runtime can monitor the size of the space used by log files and reports. Then the information about the used space is available in the system variables:

UsedLogsSpace	[MB] The size of the space used by log files
UsedReportsSpace	[MB] The size of space used by reports
AvailVirtualMemory	[MB] The size of available virtual memory
LogsFull	Is set TRUE if UsedLogsSpace exceeds limit value
ReportsFull	Is set TRUE if UsedReportsSpace exceeds limit value
VirtualMemoryLow	Is set TRUE if VirtualMemoryLow reaches allowed minimum

Table 27 - Disk and memory monitor

The allowed maximum for logs and reports and the minimum virtual memory can be defined in the General settings dialog.

Monitoring of the used space might consume the processor time, especially when there is a huge amount of log files and reports.
For this reason it can be configured and optimized by proper setting of the configuration parameters. This chore is done only when an event **CheckUsedSpace** is received by the OpcDbGateway runtime. A cyclic trigger can be created so that it will cause that this event will be periodically sent to the runtime.

7.2 Power status

The OpcDbGateway runtime monitors the power supply system. If the battery charge reaches a defined low limit value then it can stop the OpcDbGateway runtime before the power fails.

The current status of the power supply is available in following system variables:

ACLineStatus	AC power status. 0 Offline 1 Online 255 Unknown status
BatteryFlag	Battery charge status. 1 High 2 Low 4 Critical 8 Charging 128 No system battery 255 Unknown status
PLC_BatteryLifePercent	Percentage of full battery charge remaining. 0 to 100, or 255 if status is unknown. All other values are reserved.
PLC_BatteryLifeTime	Number of seconds of battery life remaining, or –1 if remaining seconds are unknown.
PLC_BatteryFullLifeTime	Number of seconds of battery life when at full charge, or –1 if full battery lifetime is unknown.

Table 28 - Power status

The General settings dialog contains two parameters **Stop On Battery Critical** and **Battery Critical Life Percent**.
If the parameter **Stop On Battery Critical** is checked then the OpcDbGateway runtime monitors the status of the battery.
If the value of the system variable **BatteryFlag** is **CRITICAL** or **BatteryLifePercent** is lower than the user-defined **Battery Critical Life Percent** then the OpcDbGateway runtime saves the content of the memory to the configuration database and stoppes its activity.

Related articles

General settings
Data persitence

OpcDbGateway and SAEAUT Universal OPC Server

Part VIII

8 Data logging

There are two types of log files:
- standard log files
- alarm log files

The purpose of the standard log file is to store various informations such as:
- start and stop of the OpcDbGateway runtime
- runtime errors
- runtime warnings (system overhead, ...)
- event statistics
- user messages (can be parameterised with memory operands)

The purpose of the alarm log file is to store:
- alarm messages

New log files are created when:
- the OpcDbGateway runtime is started
- an event **Make new log file** (or 'Make new alarm log file') are triggered (for example every 24hours)
- the OpcDbGateway runtime is stopped

The contents of each log file is protected by a checksum which is appended at the end of each file.
Log files can be viewed using log view or whatever text editor.
Log files are created in folder defined in the Sync controller dialog box**.**

Related articles
Logging in OpcDbGateway and SAEAUT UNIVERSAL OPC Server
Standard log files
Alarm log files

8.1 Standard log files

Structure of the log file name
The name of each log file has the following structure:

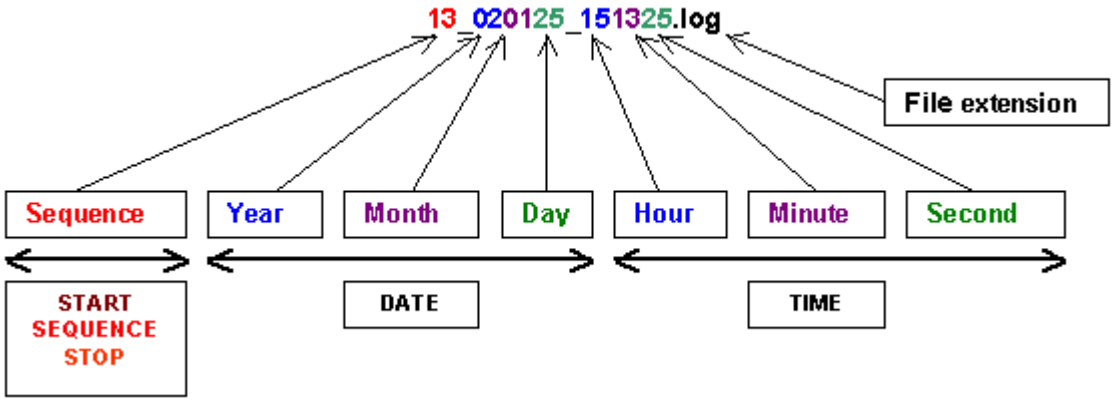


Figure 42: *The structure of the log file name*

Related articles

Log file header
Alarm log files

8.1.1 Log file header

The log file header contains following information:

COMPUTER NAME

The NetBIOS name of the local computer. This name is established at system startup, when the system reads it from the registry.

OPERATING SYSTEM

Platform	Identifies the operating system platform.
Version	Identifies the version number of the operating system.
Service pack	Indicates the latest Service Pack installed on the system. If no Service Pack has been installed, the string is empty.
Build number	Identifies the build number of the operating system.

MEMORY INFORMATION

TotalPhys	Total size, in mega bytes, of physical memory.
AvailPhys	Size, in mega bytes, of physical memory available.
TotalPageFile	Total possible size, in mega bytes, of the paging file. Note that this number does not represent the actual physical size of the paging file on disk.
AvailPageFile	Size, in mega bytes, of space available in the paging file. The operating system can enlarge the paging file from time to time. The dwAvailPageFile member shows the difference between the size of current committed memory and the current size of the paging file — it does not show the largest possible size of the paging file.
TotalVirtual	Total size, in mega bytes, of the user mode portion of the virtual address space of the calling process.
AvailVirtual	Size, in mega bytes, of unreserved and uncommitted memory in the user mode portion of the virtual address space of the calling process.

PRODUCT

Name	Product name
Version	Product version
Company name	Company name
Copyright	Copyright
Modul file name	A file path and a file name of the application.

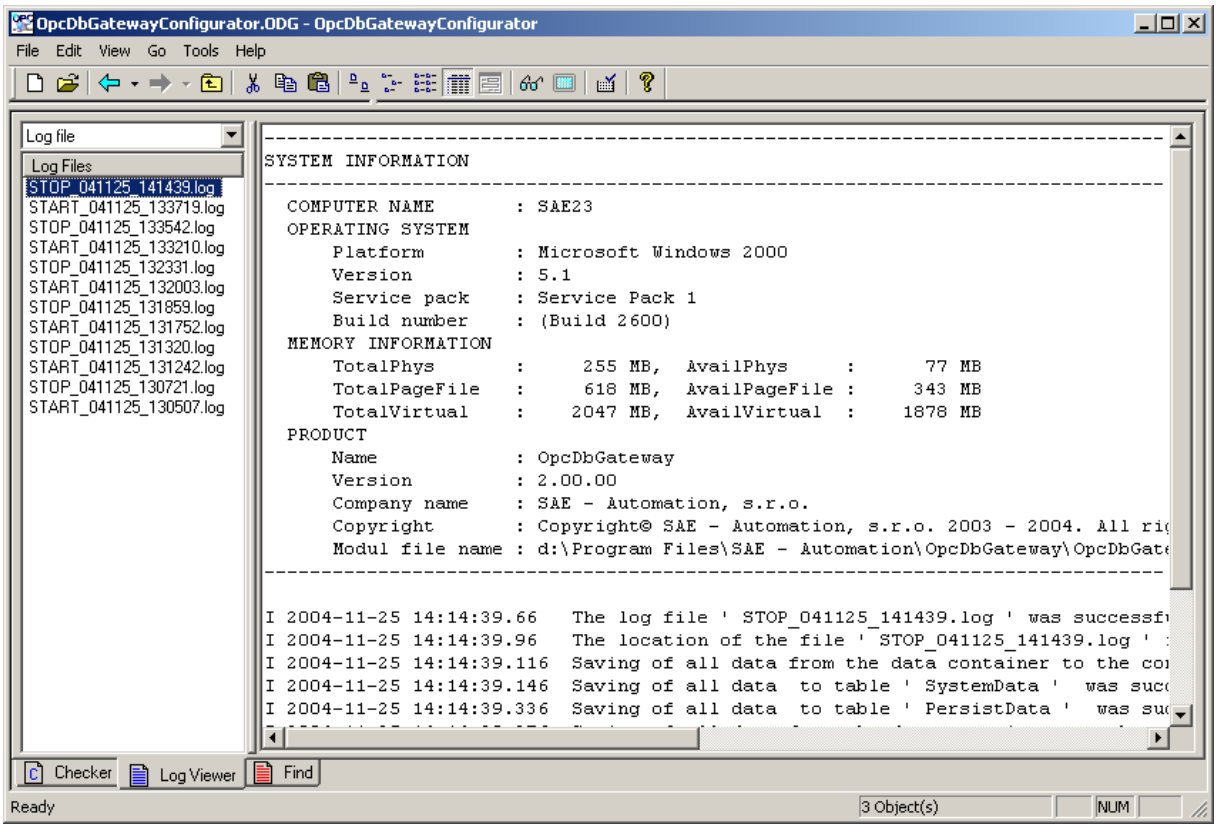


Figure 43: *Log file*

8.2 Alarm log files

The structure of the alarm log file name
The name of each alarm log file has the following structure:

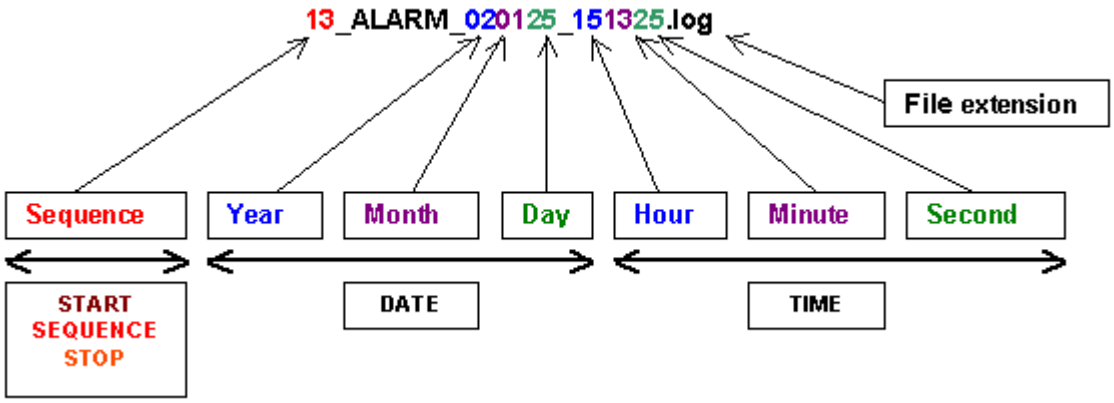


Figure 44: *The structure of the alarm log file name*

Related articles
Standard log files
Log file header

OpcDbGateway and SAEAUT Universal OPC Server

Part IX

9 Reports

The OpcDbGateway runtime can generate reports in several formats:

- text files (*.txt)
- csv files (*.csv)
- html files (*.html)
- xml files (*.xml)

To generate a report we need to define

1. When it should be generated – a trigger
2. Event used to generate the report
3. What data should be inserted to the report – an SQL query
4. The format of the report – TXT, CSV, …
5. The template file for generating of the report (optional)

The event type **create report** is used for generating of a user defined report. After the event has been triggered an **SQL query** is executed and data are selected from a database. Afterwards, the selected data are inserted into the template file and saved in the user defined folder (see Synchronous controller dialog).

The SQL query for generating reports

The SQL query used for generating of reports is a simple SELECT command which should select either all or a subset of records from a database table. The result set of the query is used as source data for generating of the report.

Example1:
SELECT * FROM tb1
The result set should contain all data from the database table tbl1.

Example2:
SELECT col1, col2 FROM tbl1
The result set should contain all data from the fields col1 and col2 of the table tbl1

Example3:
SELECT * FROM tbl1 WHERE (col1 > 5)
The result set should contain data from the table tb1 where the values in the field col1 are greater than 5.

Report types

The output format of the report is defined by the parameter **Report type**. If the 'default' type is selected then the output format is defined by the paremeter **Default report type** in the **General settings** folder.

Template files

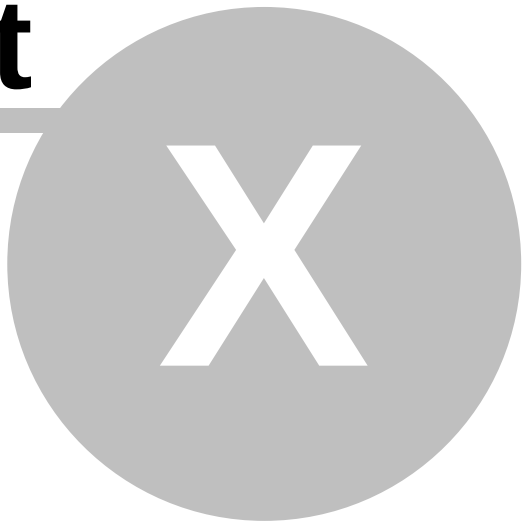
The simplest formats such as TXT, CSV, HTML or XML do not use templates.

Related articles

Events

OpcDbGateway and SAEAUT Universal OPC Server

Part



10 Examples

There are example configurations delivered with OpcDbGateway and SAEAUT Universal OPC Server.

The *ExampleConfiguration.ODG* configuration delivered with SAEAUT UNIVERSAL OPC Server differs from the configuration with the same name delivered with OpcDbGateway only by removed functionality (databases, connecting external OPC servers, proprietary alarms) that are not supported in SAEAUT UNIVERSAL OPC Serve.

Demo configurations for DDE functionality *SystemToExcel.ODG* and *DDETestBook.ODG* are installed only if DDE client enhancement is installed. Description please see in DDE client for OpcDbGateway and SAEAUT Universal OPC Server help file or on line.

10.1 Configuration ExampleConfiguration.ODG

The demo configuration *ExampleConfiguration.ODG* enables learning and testing of following features:

1. Communication of the OpcDbGateway with external devices that use external OPC servers as communication drivers
2. Communication of the OpcDbGateway with external application using enhancing DLL's
3. Communication of the OpcDbGateway with databases
4. Mapping of data items from devices, external applications and cells from database tables to memory operands
5. Working with alarms - defining of memory operands alarm activation and alarm acknowledgment
6. Mapping of memory operands to OPC items of the internal OPC server
7. Using of function blocks to perform operations with memory operands
8. Using triggers to initialize events
9. Defining of events to call function blocks and start external programmes
10. Working with trends

To understand the example and to be able to create own configurations, it is necessary to learn basic principles of working with OpcDbGateway described in
Integration of applications effectively. OpcDbGateway – configuring and programming, overview.

The demo configurations *SystemToExcel.ODG and DDETestBook* enables learning and testing of connecting external devices and applications to the OpcDbGateway using DDE client enhancement. Details please see in the DDE client on line help.

10.1.1 Using external OPC servers as communication drivers

By installation of the OpcDbGateway and SAEAUT Universal OPC Server OPC server SAEAutomation OPCSimDA is installed as well. It is supposed to be used only for demo and testing purposes. SAEAutomation OPCSimDA is used **to simulate connection to an external device over OPC server**. It offers changing data items on the simulated device as well as possibility to write to those data items.

Within Configurator, you can see configured connection to the simulation OPC server in the tree view under *External OPC Servers*. There are configured connections to the same SAEAutomation OPCSimDA OPC server - *OPCSimDA* and *OPCSimDA2.* You can try to configure also your own connection to this server. In runtime, connection to this sever is provided by internal OPC client built into OpcDbGateway.

By configuring of connection to external OPC servers you can browse for available servers or write OPC server id directly. External OPC Servers can be placed on local host or whatever computer in workgroup or network domain.

As next step you can create OPC groups see e.g. the group *OPCSimDA->Increment* that can have defined e.g. way of communication with OPC server synchronous/asynchronous update rate...

For every OPC group you can choose OPC items by browsing of address space of the external OPC and provide automatic mapping to memory operands and to OPC items of the internal OPC server.

10.1.2 Using enhancing DLL's

Enhancing Dll's enable to enhance functionality of the OpcDbGateway with own programmed modules that can be used to implement own functionality, to encapsulate ActiveX and dll's from devices vendors to communicate with their devices and even to implement almost all project functionality using programming instead of pure configuring possibilities provided by OpcDbGateway Configurator.

There is example with source codes – *Exampel1.dll* of very easy functionality implemented within projects for Visual Studio 2005 or 2012 that you can open from Start menu. They can be used as templates for your own projects.

Within configuration, the dll calling must be provided from a function block. The *Example1.dll* is called from the function block *Main.*

After creating your own enhancing dll, it is necessary to copy it to the directory where OpcDbGateway is installed - e.g. C:\Program Files\OpcDbGateway\ExternalDll.
Calling of the dll must be configured in your configuration.

10.1.3 Communication with databases

OpcDbGateway and SAEAUT Universal OPC Server can connect to many different databases using **configured connection strings.** When new configuration is created, default database with the name *ProcesDB* containing tables *AlarmStatus* and *AlarmStatusHistory* is created automatically. Within the demo configuration, also connection to another database on the local host – *DatabaseExample* with tables *Employees* and *TrendTable* is created.

Using configurator, you will be able to configure access to other databases on local or remote hosts, to create new tables on connected databases or map tables existing on connected databases to the configuration. You can also define queries for connected databases and use them in configuration e.g. using events.

10.1.4 Mapping to memory operands

The memory operands area is the place where an information exchange between different external devices, databases and computation modules implemented as enhancing DLL's is executed. By using of the OpcDbGateway configurable functionality, we work with memory operands and provide processing and data exchange between external devices, databases, program modules and the computing machine of the OpcDbGateway itself.

10.1.5 Internal OPC server

There is an OPC server providing access to memory operands within OpcDbGateway runtime.

There is possibility to define OPC items within Address space of the internal OPC server and chose to which memory operands they have to be mapped. Creating of OPC items can be done also automatically within Configurator. E.g. by configuring of the access to external OPC servers it is possible to choose that related memory operands and OPC items of the internal OPC server will be created automatically.

Address space of the internal OPC server can be structured by folders. It enables to estimate which folders from memory operands configuration contain related OPC items.
E.g. memory operands from folder *OPCSimDA_Increment* are mapped to the folder *OPCSimDA ->Increment*.

On the level of internal OPC server, different value manipulations can be defined. E. g. OPC item can be simulated using simulation signals defined in the related folder or converted using one of conversions. To the OPC items alarms according to OPC AE specification can be defined. (These are different than alarms defined for memory operands.)

There is folder with the name *System* in the address space that is created automatically by default in every new configuration. Using variables from this folder, it is possible to monitor or control internal functionality of the runtime. (Memory operands to these variables are not explicitly created, but if you need a memory operand you can create it.)

10.1.6 Commands, function blocks

Data processing in OpcDbGateway- basics

Data processing is:

- implicit that does not need configuring or programming, (e.g. transfer of unchanged data from mapped external OPC servers to internal OPC server.)
- explicit - executed by commands that are organised in function blocks (FB) and enabling processing of data from external devices applications and databases in OpcDbGateway runtime application.

Data processing can be configured using configurable commands or programmed within enhancing DLL's.

There are default function blocks that are created in every new configuration - MAIN, START, RESTART, STOP, Write Universal Log Message. (Some of them can be let dummy – without commands).

There are function blocks that are executed always only one time – by start (FB: START, RESTART) or stop (FB: STOP) of the OpcDbGateway runtime application. There is also FB MAIN that is executed in every cycle of the OpcDbGateway synchronous controller. Function blocks can be called from another FB using commands CALL or CALLREV. All FB called from FB MAIN are called cyclically (because of cyclic implicit functionality of the FB MAIN).

There can be also FBs that are executed as events activated by a trigger. Within demo application following FBs are called this way: Copy actual increment values, Write Universal Log Message.

Functionality programmed within enhancing dll is called using command CALLDLL that can be called from any FB. (In the demo configuration it is called from the FB MAIN).

FB hierarchy in demo configuration

There is following hierarchy of the FB calls within demo application:

FB: *Start*
    FB: *Init*
        Commands: 1-8
        FB: *Call Init Alarm Example*
Stop
    Nothing
Restart

FB: *Init*

Main

FB: *Generate random values*
Command: *Test arithmetics*
FB: *Call DatabaseExample_[Employees]_Copy_DB_To_MO*
Command: Copy manager salary
Command: Call DLL Example

Function blocks called as events:

Copy actual increment values
Write Universal Log Message

Functionalities demonstrated by FBs in the demo application

Init – initialization of memory operands by start and restart and calling FB *Call Init Alarm Example* to initialise memory operands used by alarming example.

Generate random values – generates random values for memory operands *RND/Random1* – 4.

MAIN - command *Test arithmetic*: makes sum of values of memory operands *Test athmetics/Input 1* and *Input 2* and put result to memory operand *Test athmetics/Output*. As operands Input1 and 2 are writeable, this functionality can be tested using OPC or web client.

MAIN –Call DatabaseExample_[Employees]_Copy_DB_To_MO – values of records in database *Emplyees* are copied to memory operands and can be read over associated OPC items of the internal OPC server. There is one MO with special meaning *DatabaseExample_[Employees]_Columns/_RowIndex*. This MO is automatically (or per hand) created by mapping of database columns to DB operands and related MO operands. It is used to define row in database table from that are values copied to related MO's. Function block *Call DatabaseExample_[Employees]_Copy_DB_To_MO* together with *DatabaseExample_[Employees]_Copy_MO_To_DB* have been created automatically using wizard called from main menu in configurator – *Tools->Wizards->CreateMapping* to database table. The wizard created automatically also database operands, memory operands and OPC items related to columns of the table.

MAIN – command *Copy manager salary* is used to show the possibility of reading one cell from database table using related DB operand - *Manager salary* to memory operand *Manager salary.* DB operands can be created by software wizard for mapping of database tables to configuration.

MAIN – command *Call DLL Example* –provides cyclic calling of the functionality programmed within enhancing dll – Example1.dll. It is used to show how programmed and configured functionality can cooperate.

10.1.7 Triggers and events

Events in OpcDbGateway and SAEAUT Universal OPC Server are initialised by triggers. Events can provide:

- Executing of function blocks,
- call an external program
- provide functionality related to configured logging
- provide activities related to configured reporting or different queries on databases

- easy testing functionality (beep).

Events can run as **synchronous** – running in the same program thread as FB *MAIN*, or **asynchronous** – running in distinct asynchronous thread.

Triggers used to initialise events can be of following types:

- based on **value of a memory operand** – trigger is activated when configured MO has value TRUE,
- based **on time** - trigger is activated when defined time elapsed.

Triggers based on time can be periodic or one time valid. Periods of triggers can be either related to start of the synchronous controller (strict period keeping) or related to calendar – period need not be always the same – e.g. according to nr. of days in a month.

Triggers based on memory operands can provide reset of the memory operand after one period of the synchronous controller (related event will be activated only one time), or can be set all the time till reset from outside (related event will be activated in every period of the synchronous controller).

Within demo following events are used:

Copy actual increment values - is used to show how to configure copying of data values from one external OPC server to other. It provides calling of FB with the same name in asynchronous thread when trigger - *10s Period trigger* is initiated -periodically every 10 s. Within this FB ,values from external OPC server *SimDA* - group *Increment* are written to external OPC server *SimDA2* group *Increment_write* using memory operands to that related OPC items are mapped.

Run Browser with first page - is used to show how to **start external applicatio**n from OpcDbGateway. The external application is started 5 s after start of the SAEAUT SCADA runtime by trigger *Run Browser after 5s.* Using application cmd.exe with file name with extension html as parameter, it is provided that the file will be opened in default web browser.

Run IExplorer - it provides start of the external application MS Internet Explorer when trigger of the type value is activated. It is alike functionality as by previous event, but at this time can be activated also from outside using memory operand Triggers/IExplorer and OPC item with the same name.

TrendsTable_Event - it is used to activate periodic writing to the trend.

Universal Log Event - it is used to provide writing a message to log file. It is activated by memory operand, that can be set e.g. from enhancing dll. This way can enhancing dll use logging functionality of the runtime application.

.

10.1.8 Alarms

The **source of alarm** in the proprietary alarm system of OpcDbGateway is **memory operand (MO)**. There are 2 MO within demo application:

Alarm Handling Example/Alarm Input - this memory operand is mapped to OPC item and so changing its value from outside, it is possible to experiment with alarming in the demo configuration. Alarm definition *Memory Operand Alarm Example* that is associated to this MO contains also definition of the memory operand *Alarm Handling Example/ AlarmAck* that is used to confirm (acknowledge)

activated alarm from outside.

DLLExampleData/Dll Example Return - this memory operand has associated alarm definition *DLL Call Unsuccessful.* This alarm is not supposed to be acknowledged and so has not associated MO for that.

10.1.9 Trends

Trend in OpcDbGateway stands for periodic writing of values of a chosen set of memory operands to a database table. As to the memory operands data from external devices applications and databases and even results of the internal computations can be mapped, it is efficient and universal way of logging structured data from all those sources to database.

This activity can be configured using standard configuration means – triggers, events, SQL queries. The whole configuring process would be relatively complicated. Because of this, within Configurator a software wizard (started from main mane Tools->Wizards->Create Historic Trends) , that makes configuration of trends much easier, has been created. What is provided by the wizard is shown in Figure 1 where you can see the start view of the wizard.
.
After running the runtime you can see in Configurator values of the trend written to the table created by wizard.

10.1.10 Demo summary

Demo configuration (and whatever other configuration of the OpcDbGateway and SAEAUT Universal OPC Server) has following parts :

- configuring of external data sources - external dll's - *External DLL Example*, external OPC servers - *OPCSimDA* and *OPCSIMDA2*, process databases - *ProcessDB* and *DatabaseExample*
- configuring of data processing - all items under Sync Controller
- configuring of connection to client applications - over *Internal OPC Server*
- configuring of higher level functionality as *Alarm handling for memory operands* and Trends

Control flow by data processing

By configuring, the control flow picture is created automatically (It can be displayed using main menu of the Configurator *View->Graphic project viewer*). Triggers, events, function blocks are displayed as rectangles within the control flow. By click on the rectangle, it is possible to get fast to the related entity and edit its settings.

There are 2 parallel lines representing functionality executed cyclically (*PLC cycle*) during period of the synchronous controller in synchronous thread. There is only function block *MAIN* executed within synchronous thread in the demo application (FB called from FB *MAIN* are not shown in the control flow picture). In case that we want also another function blocks to be executed within synchronous thread (repeatedly or not) they can be called as synchronous events.

There are shown asynchronous events that run in different - asynchronous thread. In the control flow we can see that they are not displayed between horizontal lines of the synchronous cycle. Every from them is started by a trigger and provides executing of an internal activity – call FB, execute query on a database external activity (not represented by rectangle) or external activity – call an external program.

We can see also on time executed FBs (*START, RESTART* and *STOP*) in the control flow picture.

Conclusion

Demo application does not cover all aspects and possibilities offered by the OpcDbGateway and SAEAUT Universal OPC Server. For example, as SCADA systems works often with big amounts of data points on external data sources it is necessary to understand the possibilities of mapping those data points to configuration.

OpcDbGateway and SAEAUT Universal OPC Server

Part XI

11 Appendices

11.1 Documents downloads and white Papers

OpcDbGateway on the web
OpcDbGateway blog
Buy on line
On line help
OpcDbGateway - documents, videos, white papers and downloads
SAEAUT UNIVERSAL OPC Server - documents, videos, white papers and downloads
DDE client for OpcDbGateway and SAEAUT UNIVERSAL OPC Server

11.2 Standard Query Language (SQL)

SQL is a language used in querying, updating, and managing relational databases. SQL can be used to retrieve, sort, and filter specific data to be extracted from the database. OpcDbGateway can be connected to databases from different vendors. Their implementations of SQL can differ from language described bellow. Please use vendor specific implementation for concrete databases.

11.2.1 SELECT Statement

SELECT

Retrieves an information from the database as a set of records.

Syntax
SELECT [*predicate*] { * | *table*.* | [*table*.]*field1* [AS *alias1*] [, [*table*.]*field2* [AS *alias2*] [, ...]]}
FROM *tableexpression* [, ...] [IN *externaldatabase*]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
The SELECT statement has these parts:

predicate
One of the following predicates: ALL, DISTINCT, DISTINCTROW, TOP Predicates. You use the predicate to restrict the number of records returned. If none is specified, the default is ALL.
*
Specifies that all fields from the specified table or tables are selected.
table
The name of the table containing the fields from which records are selected.
field1, *field2*
The names of the fields containing the data you want to retrieve. If you include more than one field, they are retrieved in the order listed.
alias1, *alias2*
The names to use as column headers instead of the original column names in *table*.
tableexpression
The name of the table or tables containing the data you want to retrieve.
externaldatabase

The name of the database containing the tables in *tableexpression* if they are not in the current database.

Remarks
To perform this operation, a database engine searches the specified table or tables, extracts the chosen columns, selects rows that meet the criterion, and sorts or groups the resulting rows into the order specified. SELECT statements do not change data in the database.The minimum syntax for a SELECT statement is:
SELECT *fields* FROM *table*

If a field name is included in more than one table in the FROM clause, precede it with the table name and the **.** (dot) operator. In the following example, the Department field is in both the Employees table and the Supervisors table. The SQL statement selects departments from the Employees table and supervisor names from the Supervisors table:
SELECT Employees.Department, Supervisors.SupvName
FROM Employees INNER JOIN Supervisors
WHERE Employees.Department = Supervisors.Department;

If you want a different field name or a name is not implied by the expression used to generate the field, use the AS clause to provide an alternate name for the **Field** object
SELECT BirthDate AS Birth FROM Employees;

Whenever you use **aggregate functions** or queries that return ambiguous or duplicate **Field** object names, you must use the AS clause to provide an alternate name for the **Field** object. The following example uses the title HeadCount to name the returned **Field** object in the resulting **Recordset** object:
SELECT COUNT(EmployeeID) AS HeadCount FROM Employees;

You can use the other clauses in a SELECT statement to further restrict and organize your returned data :
FROM Clause
WHERE Clause
GROUP BY Clause
HAVING Clause
ORDER BY Clause

Related articles
DELETE Statement
UPDATE Statement
INSERT INTO Statement
SELECT .. INTO Statement

11.2.1.1 ORDER BY Clause

ORDER BY Clause

Sorts a query's resulting records on a specified field or fields in ascending or descending order.

Syntax
SELECT *fieldlist*
FROM *table*
WHERE *selectcriteria*
[ORDER BY *field1* [ASC | DESC][, *field2* [ASC | DESC]][, ...]]]

A SELECT statement containing an ORDER BY clause has these parts:

fieldlist
The name of the field or fields to be retrieved along with any field-name aliases, SQL aggregate functions, selection predicates (ALL, DISTINCT, DISTINCTROW, or TOP), or other SELECT statement options.
table
The name of the table from which records are retrieved. For more information, see the FROM clause.
selectcriteria
Selection criteria. If the statement includes a WHERE clause, the orders values after applying the WHERE conditions to the records.
field1, *field2*
The names of the fields on which to sort records.

Remarks
ORDER BY is optional. However, if you want your data displayed in sorted order, then you must use ORDER BY.

Example
The default sort order is ascending (A to Z, 0 to 9). Both of the following examples sort employee names in last name order:
SELECT LastName, FirstName FROM Employees ORDER BY LastName
SELECT LastName, FirstName FROM Employees ORDER BY LastName ASC;

Example 2
To sort in descending order (Z to A, 9 to 0), add the DESC reserved word to the end of each field you want to sort in descending order. The following example selects salaries and sorts them in descending order:
SELECT LastName, Salary FROM Employees ORDER BY Salary DESC, LastName

11.2.1.2 WHERE Clause

WHERE Clause

Specifies which records from the tables listed in the FROM clause are affected by a SELECT, UPDATE, or DELETE statement.

Syntax
SELECT *fieldlist*
FROM *tableexpression*
WHERE *criteria*

A SELECT statement containing a WHERE clause has these parts:
fieldlist
The name of the field or fields to be retrieved along with any field-name aliases, selection predicates (ALL, DISTINCT, DISTINCTROW, or TOP), or other SELECT statement options.
tableexpression
The name of the table or tables from which data is retrieved.
Criteria
An expression that records must satisfy to be included in the query results.

Use various expressions to determine which records the SQL statement returns. Use the WHERE clause to eliminate records you do not want grouped by a GROUP BY clause. A WHERE clause can contain up to 40 expressions linked by logical operators, such as **And** and **Or**. When you enter a field name that contains a space or punctuation, surround the name with brackets ([ ]). For example, a customer information table might include information about specific customers :
SELECT [Customer's Favorite Restarant]

When you specify the *criteria* argument, date literals must be in U.S. format. For example, May 10, 1996, is written 10/5/96 in the United Kingdom and 5/10/96 in the United States. Be sure to enclose your date literals with the number sign (#).

Example 1
For example, the following SQL statement selects all employees whose salaries are more than $21,000:
SELECT LastName, Salary FROM Employees WHERE Salary > 21000

Example 2
Use various expressions to determine which records the SQL statement returns. For example, the following SQL statement selects all employees whose salaries are more than $21,000 and less than $25,000:
SELECT LastName, Salary FROM Employees WHERE Salary > 21000 AND Salary < 25000

Example 3
To find records dated May 10, 1996 in a United Kingdom database, you must use the following SQL statement:
SELECT * FROM Orders WHERE ShippedDate = #5/10/96#

11.2.1.3 FROM Clause

FROM Clause

Specifies the tables or queries that contain the fields listed in the SELECT statement.

Syntax
SELECT *fieldlist*
FROM *tableexpression* [IN *externaldatabase*]

A SELECT statement containing a FROM clause has these parts:
fieldlist
The name of the field or fields to be retrieved along with any field-name aliases, SQL aggregate functions, selection predicates (ALL, DISTINCT, DISTINCTROW, or TOP), or other SELECT statement options.
tableexpression
An expression that identifies one or more tables from which data is retrieved. The expression can be a single table name, a saved query name, or a compound resulting from an INNER JOIN, LEFT JOIN, or RIGHT JOIN.
externaldatabase
The full path of an external database containing all the tables in *tableexpression.*

Example
The following example shows how you can retrieve data from the Employees table:
SELECT LastName, FirstName FROM Employees

11.2.1.4 GROUP BY Clause

GROUP BY Clause

Combines records with identical values in the specified field list into a single record. A summary value is created for each record if you include an SQL aggregate function, such as **Sum** or **Count**, in the SELECT statement.

Syntax
SELECT *fieldlist*
FROM *table*
WHERE *criteria*
[GROUP BY *groupfieldlist*]

A SELECT statement containing a GROUP BY clause has these parts:
fieldlist
The name of the field or fields to be retrieved along with any field-name aliases, SQL aggregate functions, selection predicates (ALL, DISTINCT, DISTINCTROW, or TOP), or other SELECT statement options.
table
The name of the table from which records are retrieved. For more information, see the FROM clause.
criteria
Selection criteria. If the statement includes a WHERE clause, the groups values after applying the WHERE conditions to the records.
groupfieldlist
The names of up to 10 fields used to group records. The order of the field names in *groupfieldlist* determines the grouping levels from the highest to the lowest level of grouping.

Remarks
GROUP BY is optional.
Summary values are omitted if there is no SQL aggregate function in the SELECT statement.

Example
This example creates a list of unique job titles and the number of employees with each title:
SELECT Title, Count([Title]) AS Tally FROM Employees GROUP BY Title

11.2.1.5 HAVING Clause

HAVING Clause

Specifies which grouped records are displayed in a SELECT statement with a GROUP BY clause. After GROUP BY combines records, HAVING displays any records grouped by the GROUP BY clause that satisfy the conditions of the HAVING clause.

Syntax
SELECT *fieldlist*
FROM *table*
WHERE *selectcriteria*
GROUP BY *groupfieldlist*
[HAVING *groupcriteria*]

A SELECT statement containing a HAVING clause has these parts:
fieldlist
The name of the field or fields to be retrieved along with any field-name aliases, SQL aggregate functions, selection predicates (ALL, DISTINCT, DISTINCTROW, or TOP), or other SELECT statement options.
table
The name of the table from which records are retrieved. For more information, see the FROM clause.
selectcriteria
Selection criteria. If the statement includes a WHERE clause, the groups values after applying the WHERE conditions to the records.

groupfieldlist
The names of up to 10 fields used to group records. The order of the field names in *groupfieldlist* determines the grouping levels from the highest to the lowest level of grouping.
groupcriteria
An expression that determines which grouped records to display.

Remarks
HAVING is optional.
HAVING is similar to WHERE, which determines which records are selected.
After records are grouped with GROUP BY, HAVING determines which records are displayed.
A HAVING clause can contain up to 40 expressions linked by logical operators, such as **And** and **Or**.

Example
This example selects the job titles assigned to more than one employee in the Washington region.

SELECT Title, Count(Title) as Total FROM Employees WHERE Region = 'WA' GROUP BY Title HAVING Count(Title) > 1

11.2.1.6 ALL, DISTINCT, DISTINCTROW, TOP Predicates

ALL, DISTINCT, DISTINCTROW, TOP Predicates

Specifies records selected with SQL queries.

Syntax
SELECT [ALL | DISTINCT | DISTINCTROW | [TOP *n* [PERCENT]]]
FROM *table*

A SELECT statement containing these predicates has the following parts:

ALL
By default, displays all the rows in the query results. The following two examples are equivalent and return all records from the Employees table: SELECT ALL * FROM Employees ORDER BY EmployeeID; SELECT * FROM Employees ORDER BY EmployeeID;

DISTINCT
Omits records that contain duplicate data in the selected fields. To be included in the results of the query, the values for each field listed in the SELECT statement must be unique. For example, several employees listed in an Employees table may have the same last name. If two records contain Smith in the LastName field, the following SQL statement returns only one record that contains Smith: SELECT DISTINCT LastName FROM Employees; If you omit DISTINCT, this query returns both Smith records. If the SELECT clause contains more than one field, the combination of values from all fields must be unique for a given record to be included in the results. The output of a query that uses DISTINCT is not updatable and does not reflect subsequent changes made by other users.

DISTINCTROW
Omits data based on entire duplicate records, not just duplicate fields. For example, you could create a query that joins the Customers and Orders tables on the CustomerID field. The Customers table contains no duplicate CustomerID fields, but the Orders table does because each customer can have many orders. The following SQL statement shows how you can use DISTINCTROW to produce a list of companies that have at least one order but without any details about those orders: SELECT DISTINCTROW CompanyName FROM Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID ORDER BY CompanyName; If you omit DISTINCTROW, this query produces multiple rows for each company that has more than one order. DISTINCTROW has an effect only when you select fields from some, but not all, of the tables used

in the query. DISTINCTROW is ignored if your query includes only one table, or if you output fields from all tables.

TOP *n* [PERCENT]
Returns a certain number of records that fall at the top or the bottom of a range specified by an ORDER BY clause. Suppose you want the names of the top 25 students from the class of 1994: SELECT TOP 25 FirstName, LastName FROM Students WHERE GraduationYear = 1994 ORDER BY GradePointAverage DESC; If you do not include the ORDER BY clause, the query will return an arbitrary set of 25 records from the Students table that satisfy the WHERE clause. The TOP predicate does not choose between equal values. In the preceding example, if the twenty-fifth and twenty-sixth highest grade point averages are the same, the query will return 26 records. You can also use the PERCENT reserved word to return a certain percentage of records that fall at the top or the bottom of a range specified by an ORDER BY clause. Suppose that, instead of the top 25 students, you want the bottom 10 percent of the class: SELECT TOP 10 PERCENT FirstName, LastName FROM Students WHERE GraduationYear = 1994 ORDER BY GradePointAverage ASC; The ASC predicate specifies a return of bottom values. The value that follows TOP must be an unsigned Integer. TOP does not affect whether or not the query is updatable.

table
The name of the table from which records are retrieved.

11.2.2 UPDATE Statement

UPDATE Statement

Creates an update query that changes values in fields in a specified table based on specified criteria.

Syntax
UPDATE *table*
SET *newvalue*
WHERE *criteria*;

The UPDATE statement has these parts:
table
The name of the table containing the data you want to modify.
newvalue
An expression that determines the value to be inserted into a particular field in the updated records.
Criteria
An expression that determines which records will be updated. Only records that satisfy the expression are updated.

Remarks
UPDATE is especially useful when you want to change many records or when the records that you want to change are in multiple tables. You can change several fields at the same time.

Important
- UPDATE does not generate a result set. Also, after you update records using an update query, you cannot undo the operation. If you want to know which records were updated, first examine the results of a select query that uses the same criteria, and then run the update query.
- Maintain backup copies of your data at all times. If you update the wrong records, you can retrieve them from your backup copies.

Example

The following example increases the Order Amount values by 10 percent and the Freight values by 3 percent for shippers in the United Kingdom:
UPDATE Orders SET OrderAmount = OrderAmount * 1.1,Freight = Freight * 1.03 WHERE ShipCountry = 'UK'

11.2.3 INSERT INTO Statement

INSERT INTO Statement

Adds a record or multiple records to a table. This is referred to as an append query.

Syntax
Multiple-record append query:
INSERT INTO *target* [(*field1*[, *field2*[, ...]])] [IN *externaldatabase*]
SELECT [*source*.]*field1*[, *field2*[, ...]
FROM *tableexpression*
Single-record append query:
INSERT INTO *target* [(*field1*[, *field2*[, ...]])]
VALUES (*value1*[, *value2*[, ...])

The INSERT INTO statement has these parts:
target
The name of the table or query to append records to.
field1, *field2*
Names of the fields to append data to, if following a *target* argument, or the names of fields to obtain data from, if following a *source* argument.
externaldatabase
The path to an external database. For a description of the path, see the IN clause.
source
The name of the table or query to copy records from.
tableexpression
The name of the table or tables from which records are inserted. This argument can be a single table name or a compound resulting from an INNER JOIN, LEFT JOIN, or RIGHT JOIN operation or a saved query.
value1, *value2*
The values to insert into the specific fields of the new record. Each value is inserted into the field that corresponds to the value's position in the list: *value1* is inserted into *field1* of the new record, *value2* into *field2*, and so on. You must separate values with a comma, and enclose text fields in quotation marks (' ').

Remarks
You can use the INSERT INTO statement to add a single record to a table using the single-record append query syntax as shown above. In this case, your code specifies the name and value for each field of the record. You must specify each of the fields of the record that a value is to be assigned to and a value for that field. When you do not specify each field, the default value or **Null** is inserted for missing columns. Records are added to the end of the table.
You can also use INSERT INTO to append a set of records from another table or query by using the SELECT ... FROM clause as shown above in the multiple-record append query syntax. In this case, the SELECT clause specifies the fields to append to the specified *target* table.
If you append records to a table with an AutoNumber field and you want to renumber the appended records, do not include the AutoNumber field in your query. Do include the AutoNumber field in the query if you want to retain the original values from the field.
To find out which records will be appended before you run the append query, first execute and view the results of a select query that uses the same selection criteria.
An append query copies records from one or more tables to another. The tables that contain the

records you append are not affected by the append query.
Instead of appending existing records from another table, you can specify the value for each field in a single new record using the VALUES clause. If you omit the field list, the VALUES clause must include a value for every field in the table; otherwise, the INSERT operation will fail. Use an additional INSERT INTO statement with a VALUES clause for each additional record you want to create.

Example
This example selects all records in a hypothetical New Customers table and adds them to the Customers table. When individual columns are not designated, the SELECT table column names must match exactly those in the INSERT INTO table.
INSERT INTO Customers SELECT * FROM [New Customers]

Example 2
This example creates a new record in the Employees table.
INSERT INTO Employees (FirstName,LastName, Title) VALUES ('Harry', 'Washington', 'Trainee')

11.2.4 DELETE Statement

DELETE Statement

Creates a delete query that removes records from one or more of the tables listed in the FROM clause that satisfy the WHERE clause.

Syntax
DELETE [*table*.*]
FROM *table*
WHERE *criteria*

The DELETE statement has these parts:
table
The optional name of the table from which records are deleted.
table
The name of the table from which records are deleted.
criteria
An expression that determines which records to delete.

Remarks
DELETE is especially useful when you want to delete many records.
When you use DELETE, only the data is deleted; the table structure and all of the table properties, such as field attributes and indexes, remain intact.
You can use DELETE to remove records from tables that are in a one-to-many relationship with other tables. A delete query deletes entire records, not just data in specific fields. If you want to delete values in a specific field, create an update query that changes the values to **Null**.

Important
- After you remove records using a delete query, you cannot undo the operation. If you want to know which records were deleted, first examine the results of a select query that uses the same criteria, and then run the delete query.
- Maintain backup copies of your data at all times. If you delete the wrong records, you can retrieve them from your backup copies.

Example
This example deletes all records for employees whose title is Trainee. When the FROM clause includes only one table, you do not have to list the table name in the DELETE statement.

DELETE * FROM Employees WHERE Title = 'Trainee'

11.2.5 SELECT .. INTO Statement

SELECT...INTO Statement

Creates a make-table query.

Syntax
SELECT *field1*[, *field2*[, ...]] INTO *newtable* [IN *externaldatabase*]
FROM *source*

The SELECT...INTO statement has these parts:
field1, *field2*
The name of the fields to be copied into the new table.
newtable
The name of the table to be created. It must conform to standard naming conventions. If *newtable* is the same as the name of an existing table, a trappable error occurs.
externaldatabase
The path to an external database. For a description of the path, see the IN clause.
source
The name of the existing table from which records are selected. This can be single or multiple tables or a query.

Remarks
You can use make-table queries to archive records, make backup copies of your tables, or make copies to export to another database or to use as a basis for reports that display data for a particular time period. For example, you could produce a Monthly Sales by Region report by running the same make-table query each month.

Notes
- You may want to define a primary key for the new table. When you create the table, the fields in the new table inherit the data type and field size of each field in the query's underlying tables, but no other field or table properties are transferred.
- To add data to an existing table, use the INSERT INTO statement instead to create an append query.
- To find out which records will be selected before you run the make-table query, first examine the results of a SELECT statement that uses the same selection criteria.

Example
This example selects all records in the Employees table and copies them into a new table named Emp Backup.
SELECT Employees.* INTO [Emp Backup] FROM Employees

11.2.6 SQL Expressions

SQL Expressions
An SQL expression is a string that makes up all or part of an SQL statement.

Related articles
Between

In
Like

11.2.6.1 Like Operator

Like Operator

Compares a string expression to a pattern in an SQL expression.

Syntax
expression **Like** "*pattern*"

The **Like** operator syntax has these parts:
expression
SQL expression used in a WHERE clause.
pattern
String or character string literal against which *expression* is compared.

Remarks
You can use the **Like** operator to find values in a field that match the pattern you specify. For *pattern*, you can specify the complete value (for example, Like "Smith"), or you can use wildcard characters to find a range of values (for example, Like "Sm*").
In an expression, you can use the **Like** operator to compare a field value to a string expression. For example, if you enter Like "C*" in an SQL query, the query returns all field values beginning with the letter C. In a parameter query, you can prompt the user for a pattern to search for.

The following example returns data that begins with the letter P followed by any letter between A and F and three digits:
Like "P[A-F]###"

The following table shows how you can use **Like** to test expressions for different patterns.

Kind of match	Pattern	Match (returns True)	No match (returns False)
Multiple characters	a*a *ab*	aa, aBa, aBBBa abc, AABB, Xab	aBC aZb, bac
Special character	a[*]a	a*a	aaa
Multiple characters	ab*	abcdefg, abc	cab, aab
Single character	a?a	aaa, a3a, aBa	aBBBa
Single digit	a#a	a0a, a1a, a2a	aaa, a10a
Range of characters	[a-z]	f, p, j	2, &
Outside a range	[!a-z]	9, &, %	b, a
Not a digit	[!0-9]	A, a, &, ~	0, 1, 9
Combined	a[!b-m]#	An9, az0, a99	abc, aj0

Table 34 - The following table shows how you can use Like to test expressions for different patterns

Example
This example returns a list of employees whose names begin with the letters A through D.

SELECT LastName,FirstName FROM Employees WHERE LastName Like '[A-D]*'

11.2.6.2 In Operator

In Operator

Determines whether the value of an expression is equal to any of several values in a specified list.

Syntax
expr [**Not**] **In(***value1, value2, . . .***)**

Remarks
The **In** operator syntax has these parts:
expr
Expression identifying the field that contains the data you want to evaluate.
value1, *value2*
Expression or list of expressions against which you want to evaluate *expr*.

If *expr* is found in the list of values, the **In** operator returns **True**; otherwise, it returns **False**. You can include the **Not** logical operator to evaluate the opposite condition (that is, whether *expr* is not in the list of values).

Example
This example selects all customers from Brazil, Argentina, and Venezuela.
SELECT * FROM Customers WHERE Country IN ('Brazil', 'Argentina', 'Venezuela')

11.2.6.3 Between

Between...And Operator

Determines whether the value of an expression falls within a specified range of values. You can use this operator within SQL statements.

Syntax
expr [**Not**] **Between** *value1* **And** *value2*

The **Between...And** operator syntax has these parts:
expr
Expression identifying the field that contains the data you want to evaluate.
value1, *value2*
Expressions against which you want to evaluate *expr*.

Remarks
If the value of *expr* is between *value1* and *value2* (inclusive), the **Between...And** operator returns **True**; otherwise, it returns **False**. You can include the **Not** logical operator to evaluate the opposite condition (that is, whether *expr* lies outside the range defined by *value1* and *value2*).

Example
This example identifies the products with 35 or fewer units in stock.
SELECT ProductID, ProductName FROM Products WHERE UnitsInStock BETWEEN '0' AND '35'

11.2.7 SQL Aggregate functions

SQL Aggregate Functions
Using the SQL aggregate functions, you can determine various statistics on sets of values. You can use these functions in a SQL statement.

Related articles
Avg Function
Count Function
First, Last Functions
Min, Max Functions
Sum Function

11.2.7.1 Sum

Sum Function

Returns the sum of a set of values contained in a specified field on a query.

Syntax
Sum(*expr***)**

The *expr* placeholder represents a string expression identifying the field that contains the numeric data you want to add or an expression that performs a calculation using the data in that field. Operands in *expr* can include the name of a table field, a constant.

Remarks
The **Sum** function totals the values in a field. For example, you could use the **Sum** function to determine the total cost of freight charges.
The **Sum** function ignores records that contain **Null** fields.

Example
The following example shows how you can calculate the sum of the products of UnitPrice and Quantity fields:
SELECT Sum(UnitPrice * Quantity) AS [Total Revenue] FROM [Order Details]

11.2.7.2 Count

Count Function

Calculates the number of records returned by a query.

Syntax
Count(*expr***)**

The *expr* placeholder represents a string expression identifying the field that contains the data you want to count or an expression that performs a calculation using the data in the field. Operands in *expr* can include the name of a table field. You can count any kind of data, including text.

Remarks
You can use **Count** to count the number of records in an underlying query. For example, you could use **Count** to count the number of orders shipped to a particular country.
Although *expr* can perform a calculation on a field, **Count** simply tallies the number of records. It does not matter what values are stored in the records.
The **Count** function does not count records that have **Null** fields unless *expr* is the asterisk (*) wildcard character. If you use an asterisk, **Count** calculates the total number of records, including those that contain **Null** fields. **Count(*)** is considerably faster than **Count(**[*Column Name*]**)**. Do not enclose the asterisk in quotation marks (' ').
If *expr* identifies multiple fields, the **Count** function counts a record only if at least one of the fields is not **Null**. If all of the specified fields are **Null**, the record is not counted. Separate the field names with an ampersand (&).

Example
This example counts the discontinued items in the **Products** table in the **Northwind** database.
SELECT COUNT(ProductID) AS "Total Discontinued" FROM Products WHERE Discontinued = 'True'

Example 2
The following example calculates the number of records in the Orders table.
SELECT Count(*) AS TotalOrders FROM Orders

Example 3
The following example shows how you can limit the count to records in which either ShippedDate or Freight is not **Null**:
SELECT Count('ShippedDate & Freight') AS [Not Null] FROM Orders

11.2.7.3 Avg

Avg Function

Calculates the arithmetic mean of a set of values contained in a specified field on a query.

Syntax
Avg(*expr***)**

The *expr* placeholder represents a string expression identifying the field that contains the numeric data you want to average or an expression that performs a calculation using the data in that field. Operands in *expr* can include the name of a table field, a constant.

Remarks
The average calculated by **Avg** is the arithmetic mean (the sum of the values divided by the number of values). You could use **Avg**, for example, to calculate average freight cost.
The **Avg** function does not include any **Null** fields in the calculation.
You can use **Avg** in a query expression based on an SQL query.

Example
This example uses the Orders table to calculate the average freight charges for orders with freight charges over $100.
SELECT Avg(Freight) AS [Average Freight] FROM Orders WHERE Freight > 100

11.2.7.4 First, Last

First, Last Functions

Return a field value from the first or last record in the result set returned by a query.

Syntax
First(*expr***)**
Last(*expr***)**

The *expr* placeholder represents a string expression identifying the field that contains the data you want to use or an expression that performs a calculation using the data in that field. Operands in *expr* can include the name of a table field, a constant.

Remarks
The **First** and **Last** functions simply return the value of a specified field in the first or last record, respectively, of the result set returned by a query. Because records are usually returned in no particular order (unless the query includes an ORDER BY clause), the records returned by these functions will be arbitrary.

Example
This example uses the Employees table to return the values from the LastName field of the first and last records returned from the table.
SELECT First(LastName) as First, Last(LastName) as Last FROM Employees

11.2.7.5 Min, Max

Min, Max Functions

Return the minimum or maximum of a set of values contained in a specified field on a query.

Syntax
Min(*expr***)**
Max(*expr***)**

The *expr* placeholder represents a string expression identifying the field that contains the data you want to evaluate or an expression that performs a calculation using the data in that field. Operands in *expr* can include the name of a table field, a constant.

Remarks
You can use **Min** and **Max** to determine the smallest and largest values in a field based on the specified aggregation, or grouping. For example, you could use these functions to return the lowest and highest freight cost. If there is no aggregation specified, then the entire table is used.

Example
This example uses the Orders table to return the lowest and highest freight charges for orders shipped to the United Kingdom.
SELECT Min(Freight) AS [Low Freight],Max(Freight)AS [High Freight] FROM Orders WHERE ShipCountry = 'UK'

Index

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- I -

- J -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

Endnotes 2... (after index)

sae-automation@saeautom.sk,
tel.:+421-(0)42-445 07 01, fax:+421-(0)42-445 07 02,
Address: str. Trencianska 19, 018 51 Nová Dubnica, Slovakia.