Application for Communication

applications & TOOLS

Individual Signaling System Based on OPC Alarm and Events in Collaboration with the SIMATIC NET OPC Alarm and Event Server

Application Example with Code

SIEMENS

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are correctly used. These Application Examples do not relieve you of the responsibility of safely and professionally using, installing, operating and servicing equipment. When using these Application Examples, you recognize that Siemens cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice. If there are any deviations between the recommendations provided in these Application Examples and other Siemens publications – e.g. Catalogs – then the contents of the other documents have priority.

Warranty, liability and support

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act ("Produkthaftungsgesetz"), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract ("wesentliche Vertragspflichten"). However, claims arising from a breach of a condition which goes to the root of the contract shall be limited to the foreseeable damage which is intrinsic to the contract, unless caused by intent or gross negligence or based on mandatory liability for injury of life, body or health. The above provisions do not imply a change in the burden of proof to your detriment.

For questions about this document please use the following e-mail address:

mailto:csweb@ad.siemens.de

Foreword

Objective of the application

The international OPC Alarm & Events standard allows easy access to process messages and the display and processing of messages from systems of different manufacturers in a Windows application.

This application shows

- how a simple and individual signaling system can be programmed based on OPC Alarm & Events,
- how the SIMATIC NET OPC Alarm & Event server has to be configured to receive messages from S7 controllers and
- how alarm blocks can be called in the PLC from STEP 7.

Main contents of this application

The following main points are described in this application:

- Explanations of the operating principle of OPC Alarm & Events.
- Program-related handling of the OPC A&E interface of the SIMATIC NET OPC server and visualization of the messages of a SIMATIC S7 in a .NET application.
- Creating a STEP 7 program for generating the messages.
- Configuring PC stations with the SIMATIC NET OPC server and configuring the SIMATIC NET OPC A&E server.

Delimitation

This application does not include a description of the following points:

- Configuration and programming basics of the SIMATIC stations with STEP 7.
- Principle of operation of .NET and language syntax of C#.
- COM/DCOM basic technology and its configuration.

Basic knowledge of these topics is required.

Document structure

The documentation of this application is divided into the following main parts.

Part	Description
Application Description	This part provides a general overview of the contents. You are informed on the used components (standard hardware and software components and the specially created user software).
Principles of Operation and Program Structures	This part describes the detailed functional sequences of the involved hardware and software components, the solution structures and – where useful – the specific implementation of this application. It is required to read this part if you want to familiarize with the interaction of the solution components to use these components, e.g., as a basis for your own developments.
Structure, Configuration and Operation of the Application	This part takes you step by step through structure, important configuration steps, startup and operation of the application.
Appendix and Literature	This chapter provides further information such as bibliographic references, glossaries, etc.

Reference to Automation and Drives Service & Support

This entry is from the internet application portal of Automation and Drives Service & Support. The link below takes you directly to the download page of this document.

http://support.automation.siemens.com/WW/view/en/26548467

Table of Contents

Application Description

Content

This part of the document provides you with an overview of the processing of event and alarm messages from SIMATIC S7 controllers using the standardized OPC Alarms and Events interface. You are informed on the used components (standard hardware and software components and the specially created user software).

The displayed performance data illustrates the performance capability of this application.

1 Automation Problem

1.1 Overview

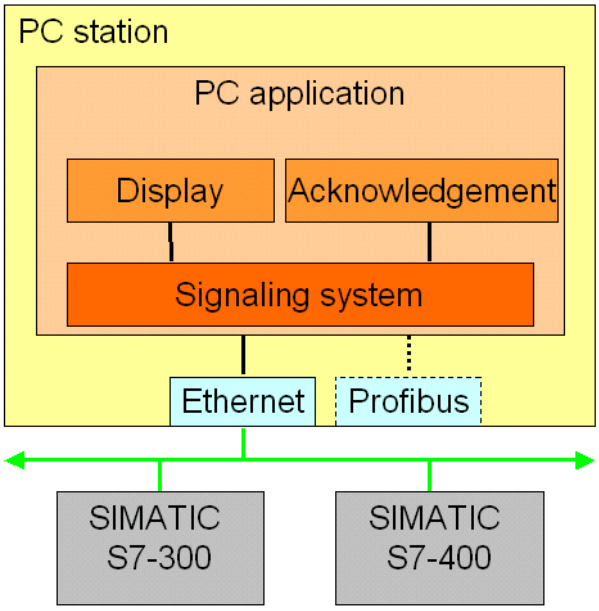
Introduction

To realize a data link, standardized mechanisms are preferably used today in order to ensure that such a data exchange remains independent of the used bus system or protocol or even manufacturer. A standardized mechanism for connecting different subsystems is also to be used for the exchange of event and alarm messages.

Overview of the automation problem

The figure below provides an overview of the automation problem.

Figure 1-1



Description of the automation problem

In an automation system, different devices are used to control the process. In this case, these devices are two SIMATIC S7 controllers. Figure 1-1 shows the involved components.

The automation problem consists of acquiring event messages and alarms from the automation systems in a central signaling system on a PC station. The PC station collects the alarms and displays them in the correct sequence. Furthermore, the PC station can acknowledge alarms if this is required. Automation systems and PC station are to communicate via Ethernet.

A communications link to the PC stations via Profibus is basically also possible; however, it will not be discussed here to simplify matters.

1.2 Requirements

Automation problem requirements

The alarm and event mechanism is not used for cyclic transmission of large data volumes; important events from the controller are signaled without causing unnecessary communication load on the controller by polling the PC station.

Controller requirement

The controller is to be capable of actively sending a message from the user program in the event of unexpected events without requiring that the controller be polled by the PC station.

The controller must have a communication option to a central signaling system, preferably via Ethernet.

PC station requirements

The PC station must have the necessary physical connection with corresponding hardware and software for the communication with the controller.

The application for display and acknowledgement of event messages is to use a standardized interface to the communications software to enable the integration of any event sources.

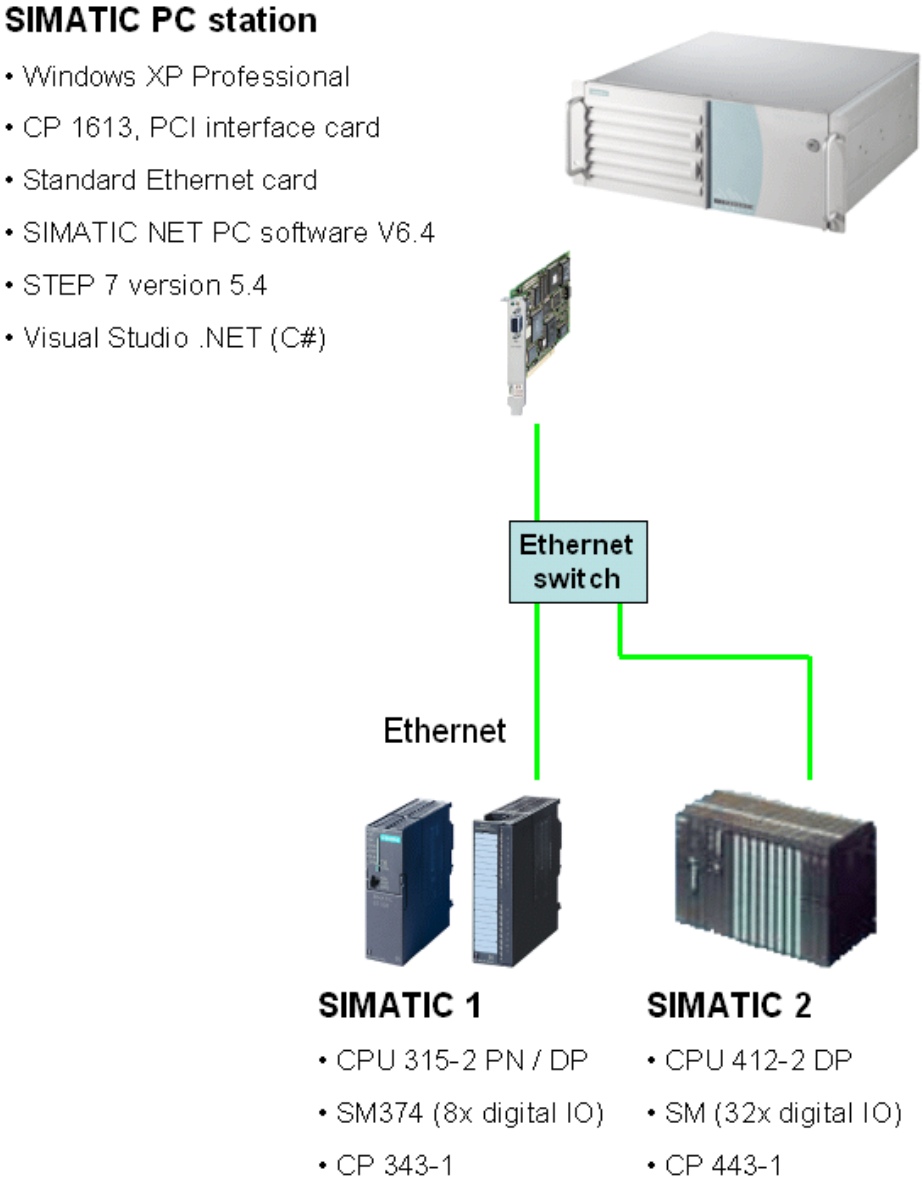
2 Automation Solution

2.1 Overview of the overall solution

Diagrammatic representation

The following figure schematically shows the most important components of the solution:

Figure 2-1



Configuration

A PC station is connected to a CPU 315-2 DP/PN and a CPU 412-2 via Ethernet. A CP 1613 is used in the PC. The onboard Ethernet card can also be used as an Ethernet card.

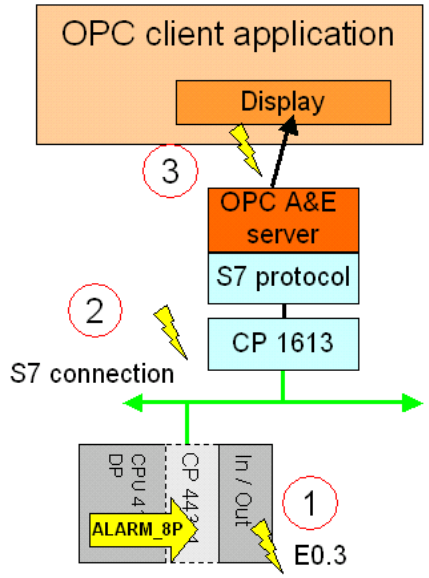
2.2 Description of the core functionality

The SIMATIC NET OPC Alarm and Events (A&E) server forms the core of the functionality of this example, which, unlike an OPC Data Access server, is capable of receiving messages directly from the controller without having to perform polling access to the controller.

The figure below shows the functional chain for such a message.

- When an event occurs, e.g. an E0.3 error input is set, an alarm block, for example ALARM_8P, can be called in the S7 program.
- Via the S7 protocol, the S7 station sends an event to the SIMATIC NET OPC server.
- The OPC A&E server converts all information to OPC parameters and sends an alarm to the OPC client application.

Figure 2-2



The OPC client application shown here also offers the option of displaying OPC alarms and, depending on the alarm type, also of acknowledging these alarms. For this purpose, the application provides the following functionality:

Display:

- Registering for the reception of events.
- Selecting and displaying attributes.
- Filtering alarms according to severity and type.
- Displaying message texts and associated values.

Acknowledgement:

- Highlighting messages requiring acknowledgement.

- Acknowledging indicating the user.

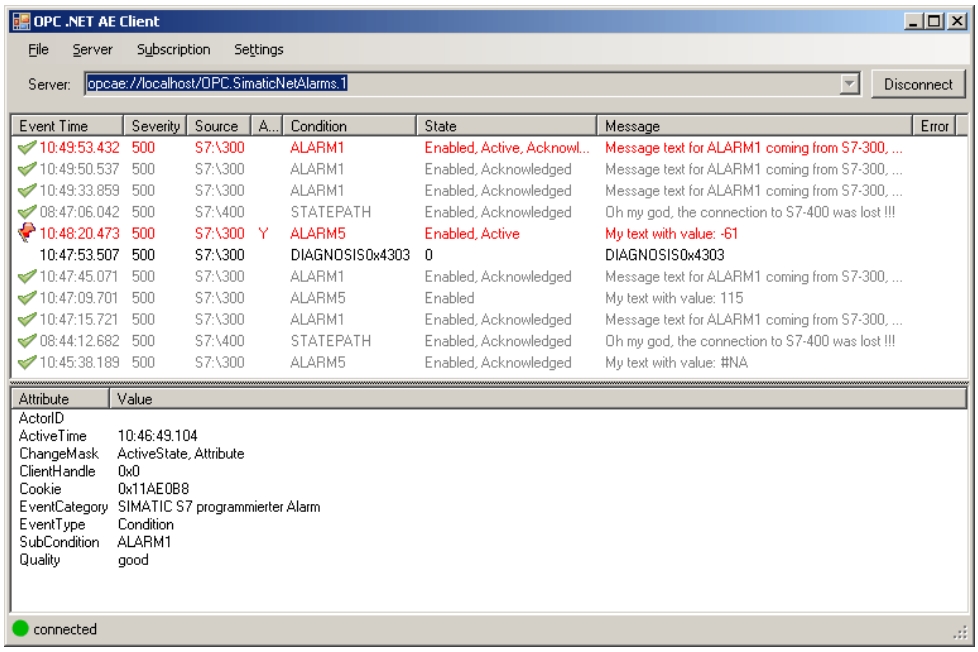
Overview and description of the user interface

The window of the user interface in Figure 2-3 displays the list of alarms and event messages. The **Server** and **Subscription** menus are used for the connection establishment to the server and the communication configuration. Alarms are acknowledged via a context menu on the alarm message or via the Server menu.

The alarms and event messages are inserted into the list from the top in the order of their occurrence. The message type, e.g. event message or alarm, is indicated by the color of the message. The status of an alarm with regard to acknowledgement and activity is indicated by the shading of the color and icons in the message line.

The user can select additional associated values for the display and make filter settings for the signaled events and alarms.

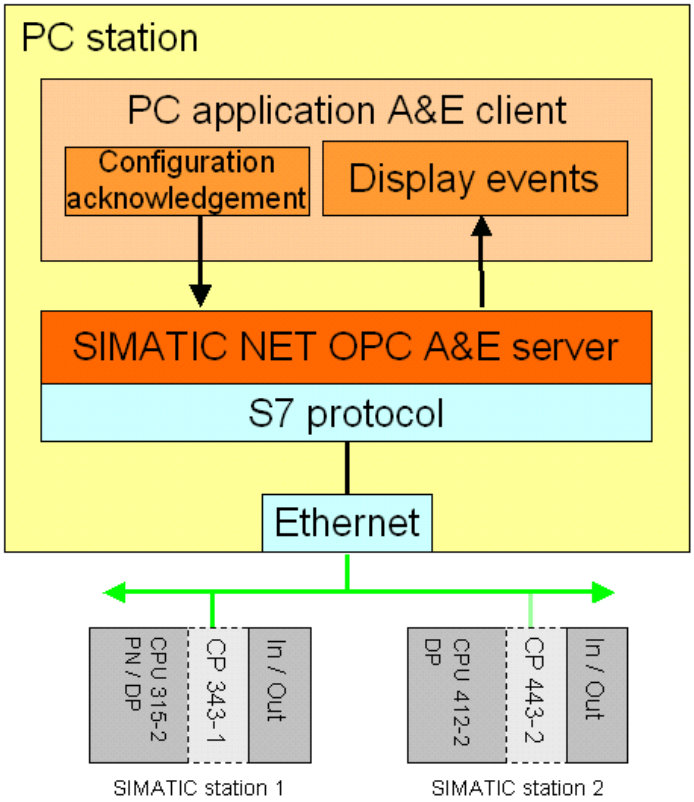
Figure 2-3



Sequence of the core functionality

Figure 2-4 schematically shows the most important components of this solution.

Figure 2-4



The following table describes which steps have to be performed to trigger different alarms and to operate the sample application.

Table 2-1

No.	Action	Note
1.	Selecting the OPC A&E server	The server must be an A&E server
2.	Registering for the reception of events	Buffer time and filter are indicated for the subscription
3.	Triggering alarms and events	Initiating programmed alarms with associated values in the PLC
4.	Triggering alarms of the connection	Simulating an interruption of the connection by removing the Ethernet cable or switching off the CP
5.	Triggering diagnostic alarms of the controllers	Simulation by start/stop or warm restart of the controller
6.	Changing the filter criteria	Severity, category and event type
7.	Acknowledging selected alarms	Different behavior of active and inactive alarms

Advantages of this solution

The advantages of the solution described here are based on using the manufacturer-independent and widely used OPC technology. This sample implementation shows the most important possibilities of the OPC Alarm and Events interface.

Advantages of the solution in detail:

- Use of the international OPC Alarm and Events standard.
- Protection of investment by expansion capability.
- Simple realization of a connection to different event and alarm sources.
- Configurable, thus flexible solution.
- Connection of non-Siemens systems with identical mechanism.
- High-performance data transmission by using the event mechanisms on all levels of the communication without the necessity for polling access.
- Acknowledgement of alarms via the OPC Alarm and Events standard.

2.3 Required hardware and software components

Hardware components

Table 2-2

Component	No.	MLFB / order number	Note
SIMATIC Rack PC IL 40 S industrial PC	1	6AG4 001-0AA21-0KX0	Configurator: See FAQ ID 17128155.
CP 1613 communications processor for ETHERNET, PCI card	1	6GK1 161-3AA00	Optionally, the onboard Ethernet card can also be used.
CPU 412-2 DP	1	6ES7-412-2XG04-0AB0	S7 CPU
CP 443-1 Advanced	1	6GK7 443-1EX40-0XE0	Ethernet communications processor.
SM474	1	DI/DO simulator	Simulation module, digital, 32xIO.
CPU315-2 DP	1	6ES7 315-2AG10-0AB0	S7 CPU, Ethernet onboard.
CP 343-1	1	6GK7 343-1EX21-0XE0	Ethernet communications processor.
SM374	1	DI/DO simulator	Simulation module, digital, 8xIO.

Standard software components

Table 2-3

Component	No.	MLFB / order number	Note
STEP 7 V5.4 SP1	1	6ES7 810-4CC08-0YA7	For configuring and programming S7 300 / 400 CPUs.
SIMATIC NET CD V7.0	1	PC Edition 2007	Delivered with the CP 1613.
Microsoft .NET Framework 2.0	1	Download from the Microsoft home page	(Optional) If older SIMATIC NET version is used.
Microsoft Visual Studio .NET 2005 Professional	1	To order this component, contact your administrator or visit the Microsoft home page	(Optional) Only if changes to the sample code are necessary.

Example files and projects

The following list includes all files and projects that are used in this example.

Table 2-4

Component	Note
26548467_OPC_AE_CODE_v10.zip	This file contains the following zip files: • **OPC_AE_STEP7_CODE.zip**: This zip file contains the STEP 7 project (incl. hardware configuration for the PC station and the SIMATIC stations). • **OPC_AE_Client_CODE.zip**: This zip file contains the executable files for the OPC client and the Visual C# project with the source code.
26548467_OPC_AE _DOKU_v10_e.pdf	This document.

2.4 Performance data

The Alarm & Event functionality is integrated in the SIMATIC NET product. The OPC A&E server is installed by default during the setup and does not have to be selected or configured separately. The OPC A&E server complies with the current OPC specification and the latest OPC Compliance Test Tool of the OPC Foundation was used to test its conformity with the OPC specification.

The S7 protocol is the only protocol that is capable of actively sending alarms from the programmable controller. For this reason, the OPC A&E functionality is only provided via the S7 protocol. It is irrelevant whether the S7 protocol is operated via Profibus or Ethernet.

Note

The SIMATIC NET product includes an additional OPC Alarm & Event server that can receive alarms of SNMP devices. This document refers exclusively to the A&E server of the S7 protocol.

Number and scope of the possible alarm blocks depend on the S7 CPU type. Alarm blocks are system functions (SFC) and thus a central component of the S7 CPU operating system.

Hardware

The SIMATIC S7 300 or 400 station and the PC must feature communications processors supporting the S7 protocol.

On the PC side, this example uses S7 connections (RFC1006) via a CP1613 (alternatively, another Ethernet card would also be possible). On the S7 side, the integrated Ethernet interface of an S7-300 and a CP443-1 in an S7-400 controller are used.

The example can be operated analogously on Profibus hardware. S7 connections are then configured via a corresponding Profibus card (CP5613/14 or CP5611). In the S7 300/400, the suitable Profibus interfaces are to be used.

S7 user program

When calling the SFC blocks in the S7 CPU, system resources are occupied for the duration of a signal cycle, thus while an alarm is present. These resources are released as soon as the alarm cycle is completed. The current status of the dynamically occupied system resources can be checked via the SFC 105 block.

Only a limited number of resources is available; this ensures that the maximum number of simultaneously active alarms is limited to approx. 50 (also less, depending on the CPU type).

For further information, please refer to the Step7 online help of the corresponding system functions and to the S7 CPU manual.

Principles of Operation and Program Structures

Content

This part describes the detailed functional sequences of the involved hardware and software components, the solution structures and – where useful – the specific implementation of this application.

It is only required to read this part if you are interested in the interaction of the solution components.

3 General Functional Mechanisms

3.1 OPC basics

Overview

In recent years, the OPC Foundation (an interest club of well-known manufacturers for the definition of standard interfaces) has defined a large number of software interfaces to standardize the information flow from the process level to the management level. According to the different requirements within an industrial application, four OPC specifications have been developed: Data Access (DA), Alarm & Events (A&E), Historical Data Access (HDA) and Data eXchange (DX). Access to process data is described in the DA specification, A&E describes an interface for event-based information, including acknowledgement, HDA describes functions for archived data and DX defines a server to server lateral communication.

This example deals with the OPC Alarm and Events 1.10 interface. A detailed documentation is available on the SIMATIC NET CD-Rom. Further information is available at www.opcfoundation.org.

What is OPC

OPC is a collection of software interfaces for data exchange between PC applications and process devices. These software interfaces have been defined according to the rules of Microsoft COM (Component Object Model) and can therefore easily be integrated into Microsoft operating systems. COM or DCOM (distributed COM) provides the functionality of the interprocess communication and organizes the exchange of information between applications even beyond computer boundaries (DCOM). Using mechanisms of the Microsoft operating system, an OPC client (COM client) can use it to exchange information with an OPC server (COM server).

The OPC server provides process information of a device on its interface. The OPC client starts the server and can access the offered data.

The OPC A&E server collects events from the process, a client registers for notification and can set filters to have only very specific alarms signaled.

OPC DX also enables direct data exchange between two OPC DA servers.

Note If the client is located on a computer other than the server and if it accesses the data “remotely”, this is referred to as a DCOM connection.

3.2 OPC Alarm & Events basics

This chapter explains the basics of the OPC Alarm and Events (A&E) interface necessary for the example.

3.2.1 OPC A&E basics and delimitation from OPC DA

OPC Data Access

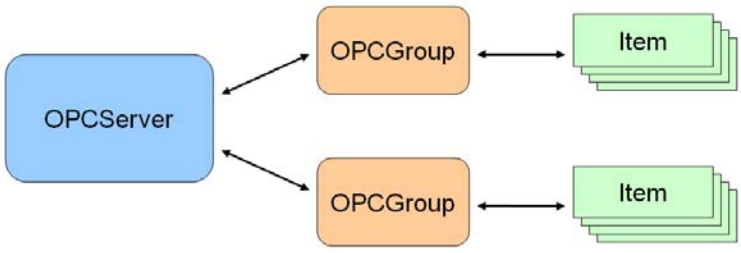
The OPC Data Access interface enables reading, writing and monitoring of variables.

The OPC DA client explicitly defines the variables (OPC items) it wants to read, write or monitor from the server. The OPC client makes this definition by establishing a connection to the server in the first step and by thus creating an OPCServer object in the OPC server, by creating an OPCGroup in which it can group items with identical settings such as update time in the second step and then by inserting items into the group in the third step. Figure 3-1 shows the different objects the OPC client creates in the server.

These items can then be read or written by the client. However, the preferred way for the data exchange from server to client is monitoring during which the server, after the set update time, signals the values of items that have changed since the last cycle to the client.

This means that events with changed data are also sent to the client when using OPC Data Access if the values of the selected items have changed. But the OPC client explicitly selects the items to be monitored.

Figure 3-1



OPC Alarm and Events

The OPC Alarm & Events interface enables the reception of event messages and alarm messages. Event messages are single messages informing the client on the occurrence of an event. Alarm messages are messages that inform the client on the change of a status in the process. Such a status can, for example, be the level of a tank. In this example, a status change can occur when a maximum level is exceeded or a minimum level is fallen below. Alarms mostly also include the requirement that the alarm has to be acknowledged. This acknowledgement is also possible via the OPC Alarm & Events interface.

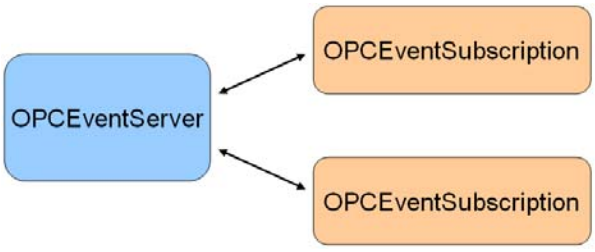
OPC Alarm & Events thus provides a flexible interface for transmitting process alarms and event from most diverse event sources.

To receive messages, the OPC A&E client logs on to the server and then receives all messages triggered in the server. To limit the number of messages, the OPC client can specify filter criteria.

The OPC client logs on by establishing a connection to the A&E server and by consequently creating an OPCEventServer object in the OPC server in the first step and by generating an OPCEventSubscription via which the messages are received in the second step. Filters for these messages can be configured on the subscription. Figure 3-2 shows the different objects the OPC client creates in the server.

In contrast to OPC DA, there is no explicit request of specific information, but all events are supplied in the process and the client can limit the quantity of the events by filter criteria when using OPC Alarm & Events.

Figure 3-2



3.2.2 OPC Alarm and Events model

Types of event messages

OPC Alarm & Events defines three types of event messages that are described in Table 3-1.

Table 3-1

Event type	Description
Simple Events	Single signaling of events without condition such as a system message providing information on the failure of components.
Tracking Events	Single signaling of changes to the process without condition such as the change of a setpoint.
Condition Events	Alarm messages informing on the change of a condition in the process. Each condition change triggers an event message. An alarm is typically a fault condition of a process requiring special handling. If the condition of the process is normal, the condition is inactive. If the defined condition changes to a fault condition, the condition is active. Such a condition can, for instance, be a limit value whose violation causes the activation of the condition and triggers a Condition Event.

Conditions of alarm messages

A status description (condition) for a part of a process has several properties that can trigger a status change. Table 3-2 describes the different properties. The properties are described using the example of the level of a tank.

Table 3-2

Property	Description
Enabled/Disabled	Indicates whether the condition monitoring is active. When the tank is in operation, the condition monitoring is enabled and violations of the selected level limit values are provided. When the process is not active, the limit values are not monitored. This can, for instance, be the case when the tank is emptied in order to clean it.
Active/Inactive	Indicates whether the condition is normal. When the tank level is within the limit values, the condition is inactive. When the limit values are violated, the condition is active.
SubCondition	Subconditions can be defined for a condition. For the level there are typically the HI conditions for a violation of the high limit, HI_HI for the violation of a critical high limit and accordingly LO and LO_LO for the underflow of the low limits.
Acknowledged	An alarm typically requires an acknowledgement by the operator. This property indicates whether the condition was acknowledged.

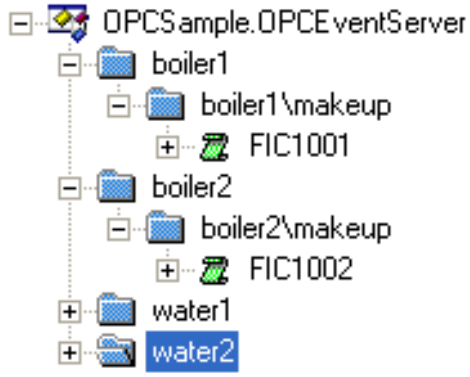
Sources for event messages

An OPC Alarm & Events server can manage several sources for event messages. The source is included in all event messages. The source is typically a device, for example a PLC.

When the server manages a large number of sources, OPC A&E offers the option of segmenting them into areas. These areas can be structured hierarchically in a tree.

The tree structure with areas and sources can be determined via the optional IOPCEventAreaBrowser interface. Figure 3-3 shows the address range of the OPC Foundation sample server with areas, for example boiler1, and sources, for instance FIC1001.

Figure 3-3



Note The SIMATIC NET A&E server does not support the optional interface for browsing.

Note SIMATIC NET uses the configured S7 connections as sources for events. The source name then corresponds to the S7 connection name.

Categories for event messages

The OPC A&E server divides the possible event messages into event categories. The event categories are defined by the server and enable a further subdivision of the event types defined by OPC Alarm and Events.

A list of event categories can be defined for each event type. Table 3-3 shows examples of event categories.

Table 3-3

Event type	Event category
Simple Event	System message
	Device error
Tracking Event	System configuration
	Change default values
Condition Event	Level alarm
	Deviation from setpoint
	System error

Filter options for event messages

Filtering enables an OPC A&E client to determine which events it wants to receive and to consequently adapt the quantity of the delivered messages to its requirements. Table 3-4 describes the possible filter criteria.

An event is forwarded to the client only if it corresponds to the filter values in all criteria. If a filter criterion is not set by the client, it will not be used.

The OPC A&E server does not have to support all filter criteria. The list of filters supported by the server can be determined by the client.

Table 3-4

Filter	Description
Event Type	Defines which of the possible event types are to be supplied. The event types given by OPC Alarm and Events are Simple, Tracking and Condition Events.
Event Category	Defines a list of possible event categories to be supplied. The event categories are defined by the OPC A&E server.
Severity	Defines the range of severity of events to be supplied. The possible range of values for the severity from 1 to 1000 is defined by OPC A&E.
Area	Defines a list of areas from which events are to be supplied. The possible areas are defined by the OPC A&E server.
Source	Defines a list of sources from which events are to be supplied. The possible sources are defined by the OPC A&E server.

3.3 Functional mechanisms of OPC Alarm & Events

This chapter explains the functional mechanisms of the OPC Alarm and Events interface necessary for the example.

3.3.1 Methods of the EventServer object

The OPCEventServer object is generated by the connection establishment of the client in the OPC A&E server. This object enables the client to determine the server configuration via the **IOPCEventServer** interface and to generate subscriptions. Table 3-5 explains the most important methods of this interface.

Table 3-5

Method	Description
QueryAvailableFilters	With this method the filters supported by the server can be determined.
QueryEventCategories	This method provides the list of event categories for the transferred event type. This list enables the client to determine which event categories the server can send for the different event types. The method supplies a numeric ID and a description for each category.
QueryEventAttributes	Using this method, the client can determine the available associated values for an event category. For a subscription these associated values can be requested in addition to the default associated values. The method supplies a numeric ID, a description and the data type of the attribute for each attribute.
CreateEventSubscription	With this method a subscription can be created in the server. When creating the subscription, properties such as active status, buffer time and maximum number of events per callback can be specified. The buffer time indicates how long events are collected until a callback with a list of events is sent to the client. This is to prevent that a separate callback is sent for each event. The maximum number of events ensures that a callback is sent in any case when the defined number of events per callback is reached. If filters are to be set for the subscription, it is advisable to generate the subscription as inactive, to set the filters on the subscription and to subsequently activate the subscription.
AckCondition	Using this method, a list of alarms requiring acknowledgement can be acknowledged. A text ID, a comment for the acknowledgement and information for the identification of the alarm have to be transferred for an alarm to be acknowledged. Source name, condition name, instant of status change to active and a cookie that was supplied with the event are transferred for the identification.

3.3.2 Methods of the EventSubscription object

The OPCEventSubscription object is generated by calling the CreateEventSubscription method on the OPCEventServer object.

This object enables the client to configure the properties and filters of the subscription via the **IOPCEventSubscriptionMgt** interface and, after generating the subscription, to determine the currently active alarms from the server. Table 3-6 explains the most important methods of this interface.

Table 3-6

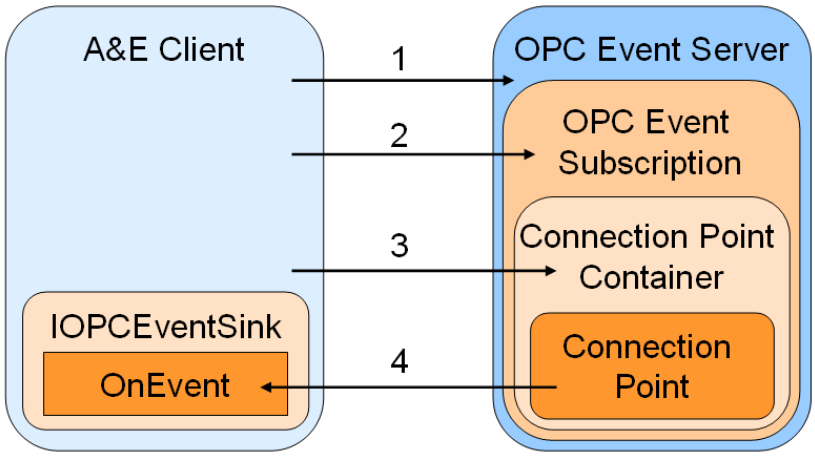
Method	Description
SetFilter	With this method the filter criteria for a subscription can be defined. Possible filters are event type, event categories, severity, areas or sources. The QueryAvailableFilters method enables the client to determine which filter criteria are supported by the server.
GetFilter	This method provides the filter criteria set for the subscription.
SelectReturnedAttributes	Using this method, additional associated values can be selected for an event category. These associated values are then supplied in addition to the associated values defined by A&E when an event of the category is signaled. The method has to be called for each event category for which additional associated values are to be called.
Refresh	The call of this method triggers event messages for all active conditions in the server. After activating a subscription, all event messages after this instant are sent to the client. Messages before the instant are not sent and can only be retrieved with Refresh for active conditions.
SetState	With this method the subscription properties such as active status, buffer time and maximum number of events per callback can be changed. The buffer time indicates how long events are collected until a callback with a list of events is sent to the client. This is to prevent that a separate callback is sent for each event. The maximum number of events ensures that a callback is sent in any case when the defined number of events per callback is reached. If filters are to be set for the subscription, it is advisable to generate the subscription as inactive, to set the filters on the subscription and to subsequently activate the subscription.
GetState	With this method the settings for the subscription such as active status, buffer time and maximum number of events per callback can be determined.

3.3.3 Receiving messages from A&E server

Register mechanism for events

A client is registered for events in four steps. Figure 3-4 shows the steps and the objects involved in an overview.

Figure 3-4



The four steps for registering are:

- **Step 1 – Establishing the connection to the OPC server**
  COM mechanisms are used to establish the connection from the OPC A&E client to the server. The OPC Event Server object is generated during this process.
- **Step 2 – Creating a subscription**
  Using the **CreateEventSubscription** method on the OPC Event Server object, the client can create a subscription. During this process the OPC Event Subscription object is generated in the server.
  The subscription should be created as inactive. The **SetFilter** method on the subscription is used to make the filter settings.
- **Step 3 – Establishing the callback connection**
  Receiving the event messages requires that the OPC A&E client implements the **IOPCEventSink** interface with the **OnEvent** method. This callback interface is registered via the connection point container and the associated connection point in the Subscription object. After registering the callback interface, the client can activate the subscription using the **SetState** method.
- **Step 4 – Sending the event messages**
  After the subscription has been activated, the server sends a corresponding list of event messages to the client via the **OnEvent** method. After activating, the client can request the messages for the active conditions using the Refresh method. The **OnEvent** method is also used for signaling.

Associated values of Simple Events

The OPC A&E specification defines a set of associated values that is supplied by the server for Simple Events. Table 3-7 describes these associated values. The other event types are also based on these associated values, but they supply additional associated values for the respective type.

Depending on the event categories, additional associated values can be additionally requested.

Table 3-7

Associated value	Description
Source	Name of the event message source.
Time	Instant at which the event has occurred.
Type	Event type of the event message. Value Meaning 1 Simple Event 2 Tracking Event 4 Condition Event
Event Category	Numeric ID of the event category defined by the server. The list of the event categories supported by the server with numeric ID and a description can be determined using the QueryEventCategories method on the Event Server object.
Severity	Severity of the event message. The range of values of the severity is from 1 to 1000. 1000 corresponds to the highest severity.
Message	Descriptive text for the event message.

Associated values of Tracking Events

The OPC A&E specification defines a set of associated values that is supplied by the server for Tracking Events. This set consists of the associated values defined for Simple Events and an additional associated value for Tracking Events. Table 3-8 describes this additional associated value.

Depending on the event categories, additional associated values can be additionally requested.

Table 3-8

Associated value	Description
Actor ID	Text ID identifying the application that has triggered the event, for example, by changing a setpoint. If this application is an OPC client, this is the OPC client name that can be set when establishing the connection.

Associated values of Condition Events

The OPC A&E specification defines a set of associated values that is supplied by the server for Condition Events. This set consists of the associated values defined for Simple Events and additional associated values for Condition Events. Table 3-9 describes these additional associated values.

Depending on the event categories, additional associated values can be additionally requested.

Table 3-9

Associated value	Description
Condition Name	Name of the condition at which a status change has occurred.
Sub Condition Name	Name of the active subcondition within the condition.
Change Mask	A bit mask indicating which condition properties have changed and thus triggered the event. Bit number — Meaning 0 — Active status 1 — Acknowledgement status 2 — Enabled / Disabled 3 — Quality of the value for the condition monitoring 4 — Severity 5 — Subcondition 6 — Message text 7 — Additional attributes
New State	A bit mask indicating the condition status at the time of the event. If the bit for the corresponding property is set, this property is active. Bit number — Property — Meaning when bit set 0 — Enabled / Disabled — TRUE = Enabled 1 — Active status — TRUE = Active 2 — Acknowledgement status — TRUE = Acknowledged
Condition Quality	Indicates the quality of the value on which the condition is based. This can, for instance, be the level of a tank.
Ack Required	A flag indicating whether the alarm has to be acknowledged.
Active Time	Instant at which the status of the condition has changed to active or at which the change to the current subcondition has been performed.
Cookie	A server-defined cookie that uniquely identifies the event together with other parameters from the event. When calling AckCondition on the Event Server object, this identification is necessary to acknowledge the alarm.
Actor ID	Text ID identifying the application that has acknowledged the alarm. If this application is an OPC client, this is the text ID that is transferred when acknowledging with AckCondition. This associated value is an empty string when the event is not triggered by an acknowledgement.

4 Functional Mechanisms of the Client Application

4.1 Alarm client functionality

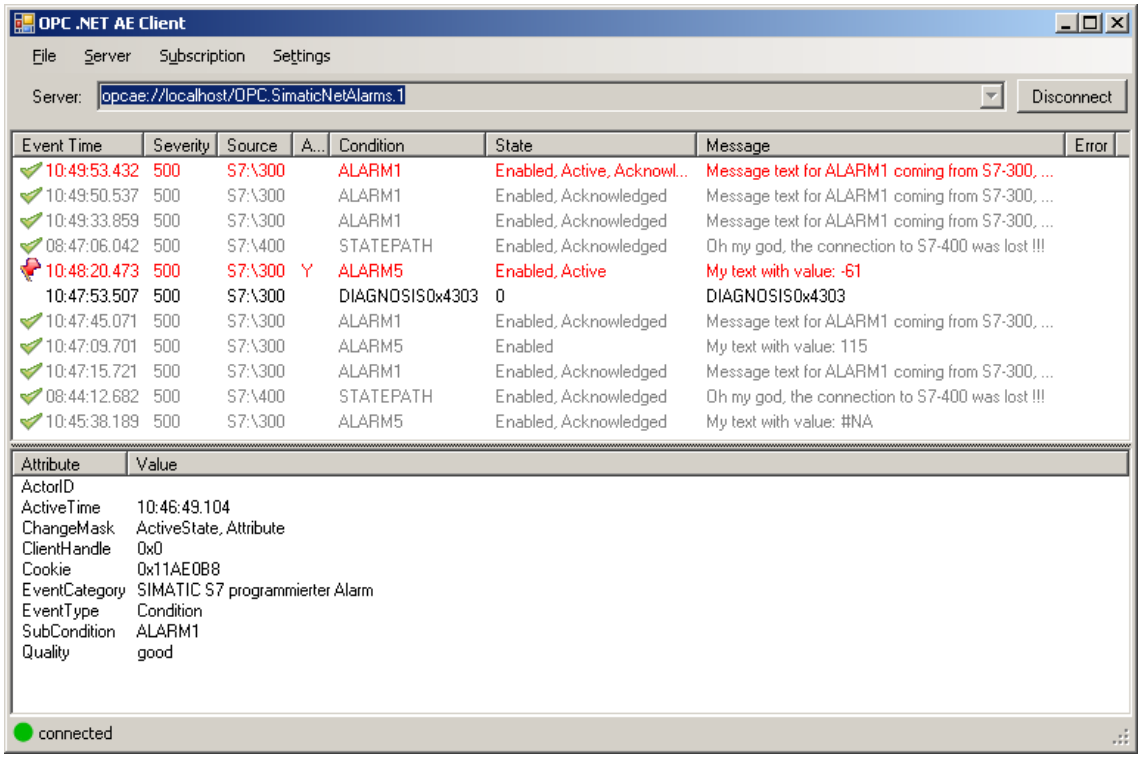
The OPC client application is used for the following tasks:

- Display of event messages and alarms of any OPC Alarm and Event servers.
- Selection and display of additional associated values, depending on the event categories provided by the server and their associated values.
- Acknowledgement of alarms.
- Configuration of the filter criteria to be able to restrict the signaled events.

User interface of the application

The following figure shows the user interface of the OPC client.

Figure 4-1



Configuration of the OPC A&E communication

The user interface is used to visualize the event and alarm messages and to change the settings for the quantity of the delivered messages and their associated values. Based on the instant they were received, the messages are arranged in the list from the bottom up. The latest message is thus always displayed on the top of the list. The list length is limited to 100 entries, older messages are deleted.

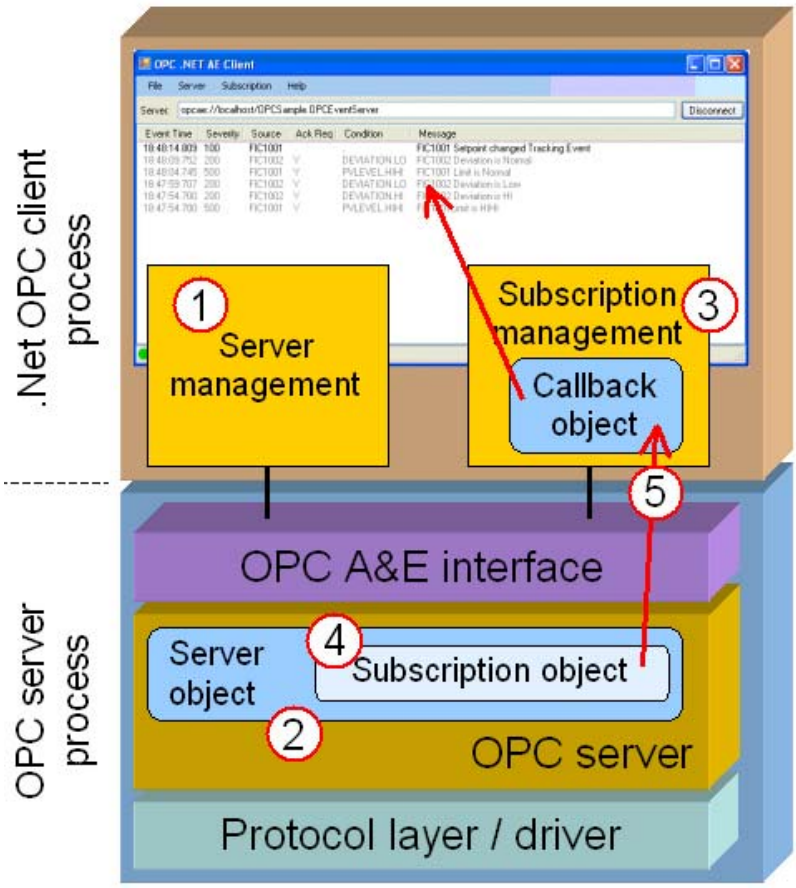
The following actions can be performed using the elements of the user interface (menu, context menu in the list or buttons):

- Connecting to the set OPC A&E server.
- Disconnecting from the OPC A&E server.
- Configuring the settings for the transfer of the messages.
- Configuring the settings for the filters that are applied to the messages.
- Selecting additional associated values.
- Acknowledging alarm messages.

4.2 Functional mechanisms of the alarm client

The figure below shows the function blocks in the OPC client and the interaction with the OPC server.

Figure 4-2



The following table explains the individual function blocks and steps.

Table 4-1

No.	Description
1	When establishing the connection between user interface and OPC server, a Server object is generated on the client side. This object encapsulates the .NET accesses to the EventServer object of the OPC Alarm and Events COM interface (2).
2	The OPCEventServer object is generated in the server via the OPC A&E interface.
3	When establishing the connection between user interface and OPC server, a Subscription object is generated on the client side. This object encapsulates the .NET accesses to the EventSubscription object of the OPC Alarm and Events COM interface (4).
4	The OPCEventSubscription object is generated in the server via the OPC A&E interface. On the subscription, the filters for the messages are set and additional associated values are requested.
5	To be able to receive event messages from the server, a callback connection is established. A Callback object is created in the client and connected to the subscription in the server. When messages are sent from the server to the client, the client enters the messages in the list.

4.3 Class diagrams of the alarm client

This chapter explains the static structure of the OPC client using a UML class diagram.

The following table provides an overview of the most important C# classes and classifies them according to their tasks.

The classes are divided into three modules

- OPC Alarm and Event Client API
  The module in the AEClientApi directory provides a .NET interface for access to OPC A&E servers. The module can be used independently of the sample application.
- Event View Control
  Based on a subscription from the OPC A&E Client API, this .NET Control displays event messages of OPC servers. The module can be used independently of the sample application.
- Sample application
  The sample application in the AEClient directory implements the OPC A&E client using the EventView and OPC A&E Client API modules.

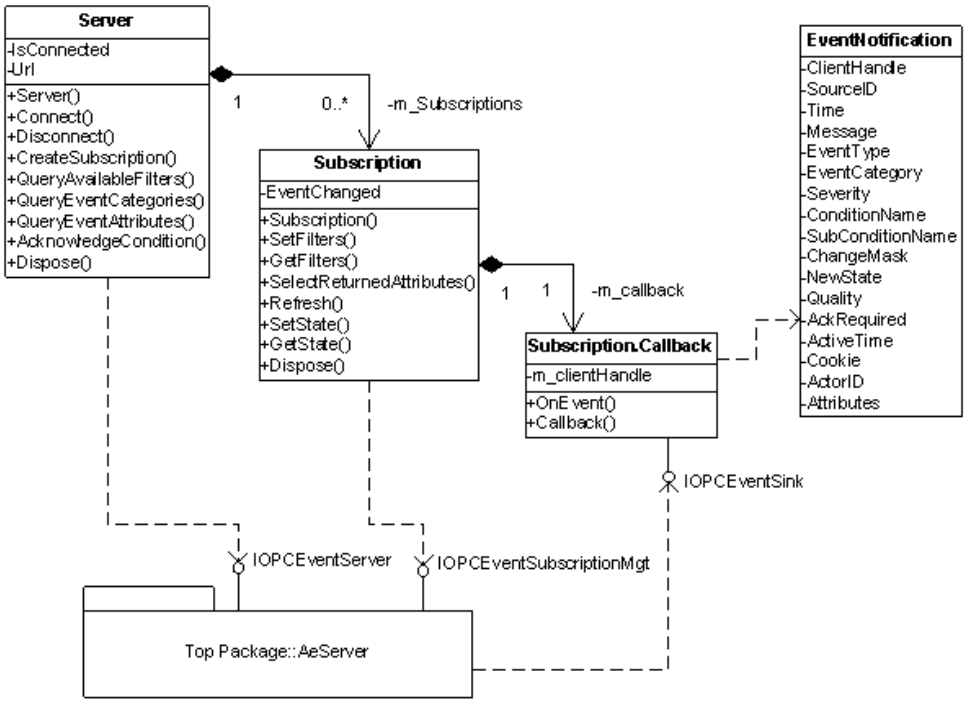
Table 4-2

Class	Task	Description
Server	OPC A&E Client API	A wrapper class that encapsulates the OPCEventServer functionality and thus enables simplified access to the COM object via .NET methods
Subscription	OPC A&E Client API	A wrapper class that encapsulates the OPCEventSubscription functionality and thus enables simplified access to the COM object via .NET methods.
EventNotification	OPC A&E Client API	This class encapsulates the data of an event.
SampleClient	Sample application	This class implements the functionality of the main dialog box of the sample application.
EventView	Display of events	This class implements the functionality to display event messages.
SubscriptionSettingsDlg	Sample application	This class implements the dialog box for configuring the general subscription settings.
FilterSettingsDlg	Sample application	This class implements the dialog box for configuring the filter settings for the subscription.
AttributeSelectionDlg	Sample application	This class implements the dialog box for selecting the additional associated values for the subscription.

4.3.1 OPC Alarm and Event Client API

The following class diagram shows the OPC A&E Client API classes. These classes encapsulate the accesses to the COM interface in a .NET API. The individual classes are explained in detail on the following pages.

Figure 4-3



Server class

The **Server** wrapper class described in the following table encapsulates the OPCEventServer functionality and thus enables simplified access to the COM object via C# methods.

The class is implemented in the OpcAeServer.cs file in the AEClientApi module.

Table 4-3

Method	Functionality
Server	Constructor for the Server class.
Connect	Establishes the connection to the server using COM mechanisms.
Disconnect	Enables all COM interfaces of the server and thus terminates the connection to the OPC server.
CreateSubscription	Generates a subscription with the transferred settings.
QueryAvailableFilters	Supplies the filter criteria accepted by the OPC server.
QueryEventCategories	Supplies the event categories provided by the OPC server. These event categories can be used for filtering.

Method	Functionality
QueryEventAttributes	Supplies the possible additional attributes that can be requested for an event category in a subscription.
AcknowledgeCondition	Acknowledges one or several alarm messages.
Dispose	Enables all COM interfaces of the server and thus terminates the connection to the OPC server.

Subscription class

The **Subscription** class described in the following table is a wrapper class that encapsulates the OPCEventSubscription functionality and thus enables simplified access to the COM object via C# methods.

The class is implemented in the OpcAeSubscription.cs file in the AEClientApi module.

Table 4-4

Method	Functionality
Subscription	Constructor for the Subscription class.
SetFilters	Sets the filter criteria to be used for the event messages of this subscription.
GetFilters	Supplies the currently set filter criteria
SelectReturnedAttributes	Enables the selection of additional attributes for an event category and has to be called for each event category.
Refresh	Using this method, all messages for currently active conditions can be requested.
SetState	Sets the subscription settings, for example active status.
GetState	Supplies the current subscription settings.
Dispose	The call of this method is necessary if the subscription is no longer required and if all COM interfaces of the subscription are to be enabled. Enables defined disconnecting from the subscription.
EventChanged	Property of the class via which the delegate for event callbacks to the application can be registered. When registering, the callback connection to the OPC server is also established. The mechanism for registering the callback to the OPC server is described in chapter 3.3.3.

EventNotification class

The **EventNotification** class encapsulates the data of an event. The properties of the class correspond to the associated values of the event messages described in chapter 3.3.3.

The class is implemented in the OpcAeSubscriptionHelper.cs file in the AEClientApi module.

Subscription.Callback class

The **Subscription.Callback** class described in the table below is an internal utility class to implement the callback interface from the OPC server to the OPC client for signaling the events. The conversion of the OnEvent callback to the EventChanged delegate of the subscription is realized in this class.

The class is implemented in the OpcAeSubscription.cs file in the AEClientApi module.

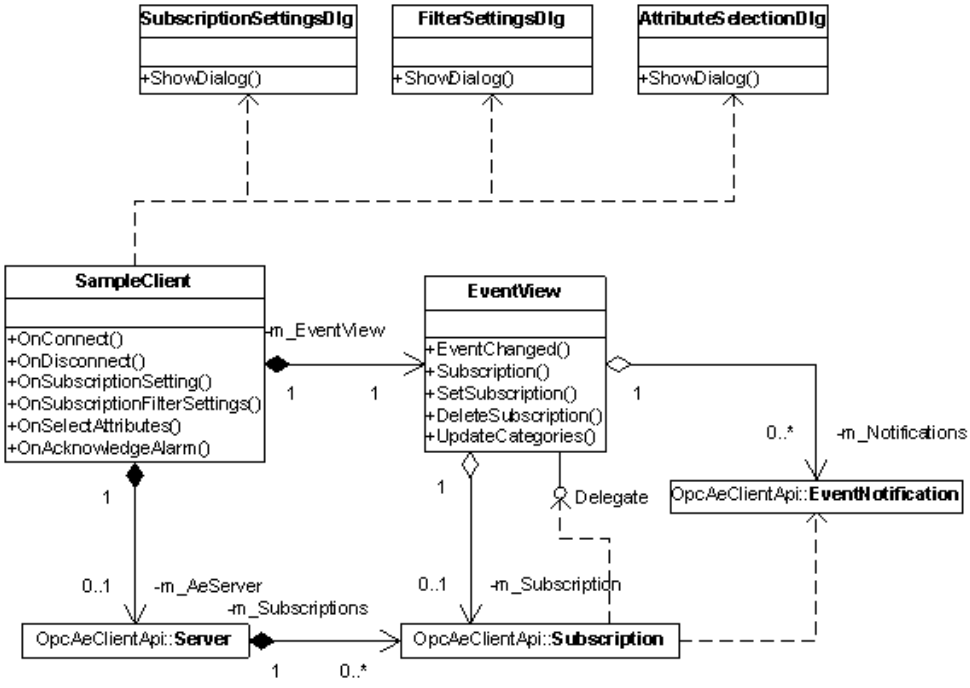
Table 4-5

Method	Functionality
OnEvent	This method implements the callback interface for the signaling of events from the server to the client. When this method is called by the server, the transferred event information for each event is packed in the EventNotification class and forwarded to the application via the EventChanged delegate.

4.3.2 OPC A&E client sample application

The following class diagram shows the classes of the OPC A&E sample client. These classes realize the functionality of the user interface and use the A&E Client API classes. The individual classes are explained in detail on the following pages.

Figure 4-4



SampleClient class

The **SampleClient** class described in the following table implements the main dialog box functionality of the client application.

The class name corresponds to the file name in the AEClient module.

Table 4-6

Method	Functionality
OnConnect	Implements the functionality to connect to the server, to poll the settings for the subscription and to create the subscription.
OnDisconnect	Deletes the subscription and disconnects from the server.
OnSubscriptionSettings	Opens the dialog box for the subscription settings and updates the settings of the subscription.
OnSubscriptionFilterSettings	Opens the dialog box for the filter criteria settings and updates the filter criteria for the subscription.
OnSelectAttributes	Opens the dialog box for selecting the attributes.
OnAcknowledgeAlarm	Acknowledges an alarm.

EventView class

The **EventView** class described in the table below implements the functionality to display event messages.

The class name corresponds to the file name in the EventView module.

Table 4-7

Method	Functionality
SetSubscription	Sets the subscription for the EventView and registers the callback for the EventView.
DeleteSubscription	Deletes the subscription.
OnEventChanged	Receives new events and enters them in the list.

SubscriptionSettingsDlg class

The **SubscriptionSettingsDlg** class described in the following table implements the dialog box for configuring the general subscription settings.

The class name corresponds to the file name in the AEClient module.

Table 4-8

Method	Functionality
ShowDialog	Transfers the current status of the subscription to the dialog box and displays the dialog box. If the dialog box is closed with OK, the new settings for the subscription are read out of the dialog box and returned.

FilterSettingsDlg class

The **FilterSettingsDlg** class described in the table below implements the dialog box for configuring the filter settings for the subscription.

The class name corresponds to the file name in the AEClient module.

Table 4-9

Method	Functionality
ShowDialog	Transfers the current filter settings of the subscription to the dialog box and displays the dialog box. If the dialog box is closed with OK, the new filter settings for the subscription are read out of the dialog box and returned.

AttributesDlg class

The **AttributesDlg** class described in the following table implements the dialog box for selecting the additional associated values for the subscription.

The class name corresponds to the file name in the AEClient module.

Table 4-10

Method	Functionality
ShowDialog	Transfers the saved filter settings to the dialog box and displays the dialog box. If the dialog box is closed with OK, the selected attributes are read out of the dialog box and returned.

4.4 Sequence diagrams of the alarm client

Connecting to the OPC server

The following sequence diagram shows the sequences when the user of the application executes the “Connect Server” action to establish the connection to the OPC server. This action can be started using the **Connect button** in the user interface or via the **Server menu**.

Figure 4-5

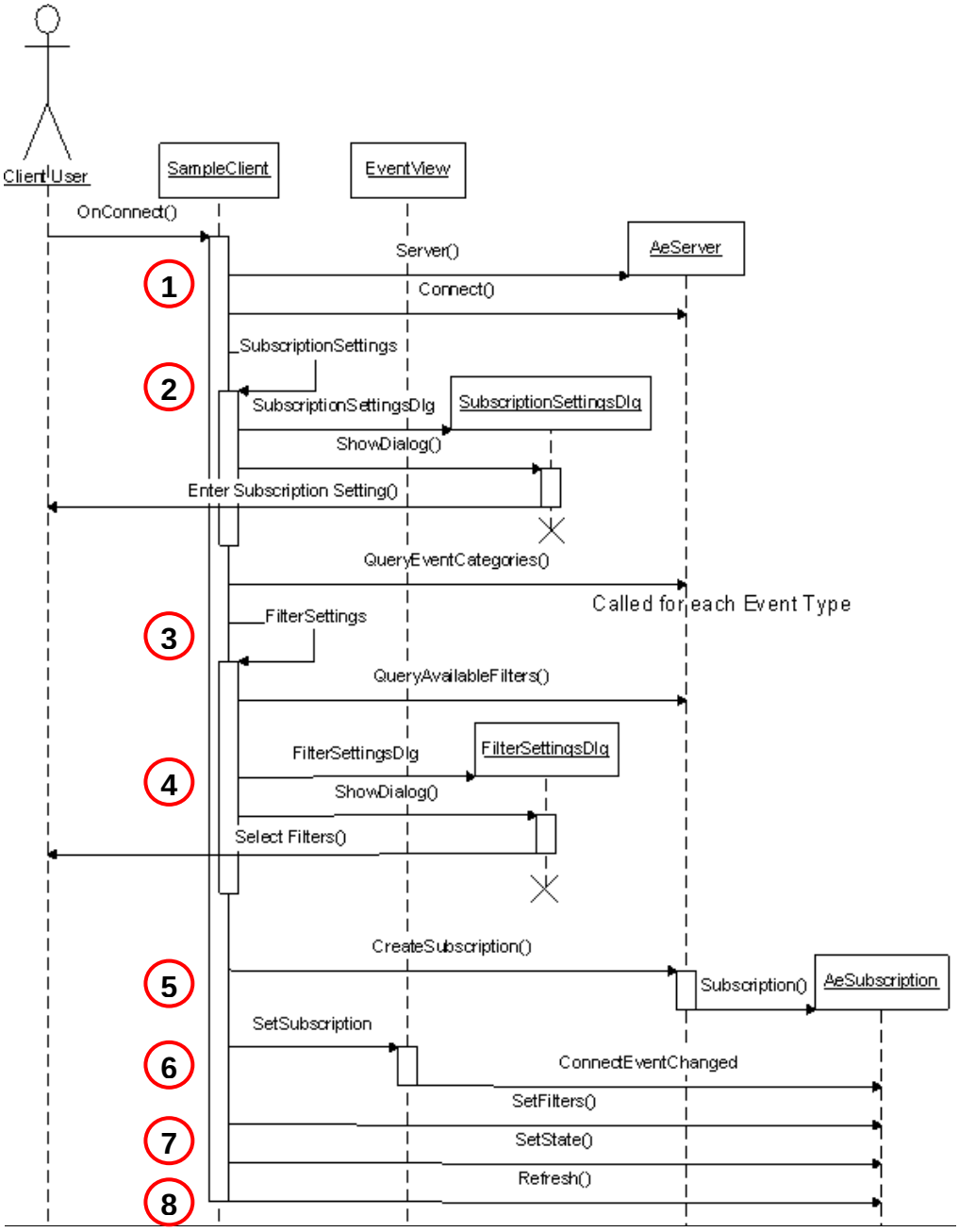


Table 4-11

No.	Description
1	The **OnConnect** method on the SampleClient object is called by the “Connect Server” user action. The actions for connecting to the server and for creating the subscription are performed in this method. In the first step, the Server object from the Client API is created and the **Connect** method on the Server object is called. This ensures that the connection to the OPC server defined by the URL is established.
2	The dialog box for the subscription settings is created and initialized using the **SubscriptionSettings** auxiliary function of the SampleClient class. Default values are transferred as status. Subsequently, the dialog box for the settings is opened to enable the user to make the desired settings. When the user closes the dialog box, the settings are determined from the dialog box.
3	The **QueryAvailableFilters** and **QueryEventCategories** methods on the Server object are used to determine the necessary configuration parameters for the filter dialog box by the OPC server. **QueryEventCategories** is called for each event type.
4	The dialog box for the filter settings is generated and initialized with the information from the server. The configuration parameters determined in step 3 and blank filter settings are transferred. Subsequently, the dialog box for the filter settings is opened to enable the user to select the desired filters. When the user closes the dialog box, the settings are determined from the dialog box.
5	Using the **CreateSubscription** method on the Server object, a subscription is generated as inactive.
6	The subscription created in step 5 is transferred to the EventView object with **SetSubscription**. In this method, EventView connects the **Callback** method in the EventView object to the delegate of the subscription. The subscription activates the callback connection to the OPC server.
7	After generating the subscription and after establishing the callback connection, the filter settings for the subscription are made using the **SetFilter** method on the Subscription object. By calling the **SetState** method on the Subscription object, the desired status of the subscription is set. If the user has configured the subscription as active, the subscription is activated with this call and events can be sent from the server to the client.
8	By calling the **Refresh** method on the subscription, the messages for active conditions are requested from the server. Like all other event messages, these messages are delivered to the client via the callback mechanism.

Disconnecting from the OPC server

The following sequence diagram shows the sequences when the user of the application executes the “Disconnect Server” action. This action can be started using the **Disconnect button** in the user interface or via the **Server menu**.

Figure 4-6

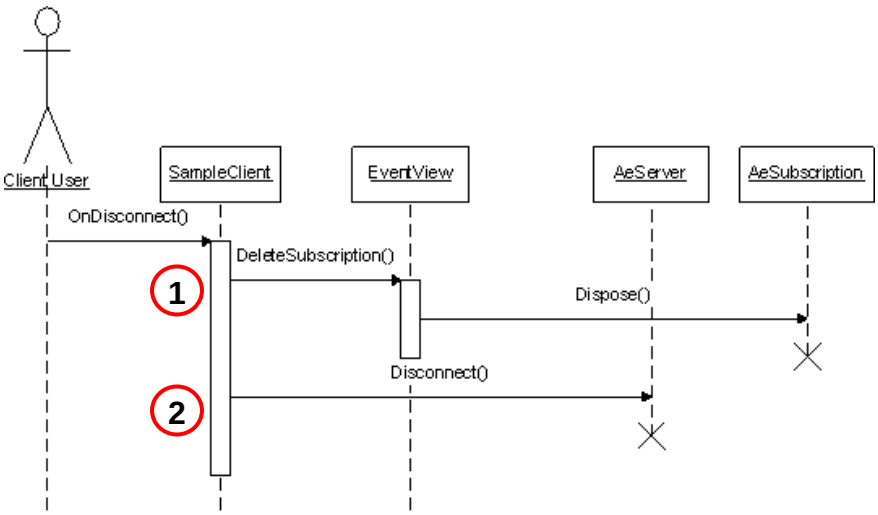


Table 4-12

No.	Description
1	The **OnDisconnect** method on the SampleClient object is called by the “Disconnect Server” user action. The actions for deleting the subscription and for disconnecting from the server are performed in this method. In the first step, the subscription is deleted in the server. This is done by calling the **DeleteSubscription** method on the EventView object that calls the **Dispose** method on the AeSubscription object. In this method, the COM interfaces of the Subscription object are enabled in the OPC A&E server; this deletes the subscription.
2	In the second step, the **Disconnect** method on the AeServer object is called. In this method, the COM interfaces of the Server object are enabled in the OPC A&E server; this clears the connection.

Changing the subscription settings

The sequence diagram below shows the sequences when the user of the application executes the “Change Subscription Settings” action. This action can be started via the Subscription menu.

Figure 4-7

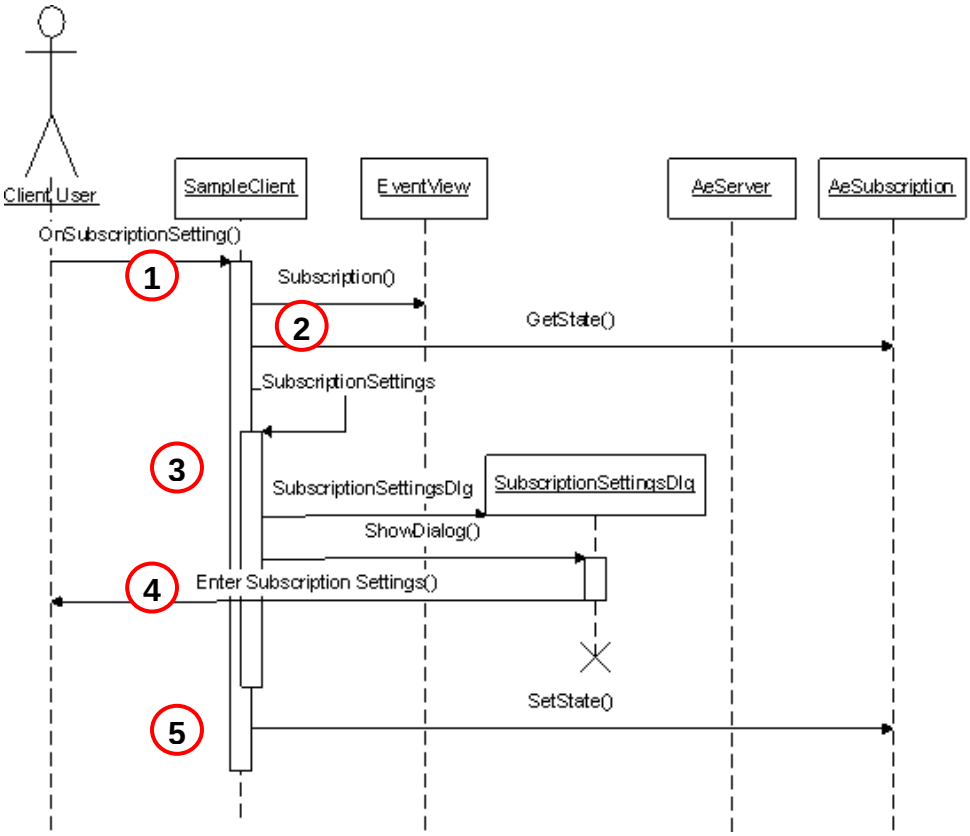


Table 4-13

No.	Description
1	The **OnSubscriptionSetting** method on the SampleClient object is called by the “Change Subscription Setting” user action.
2	The current subscription status is read out using the **GetState** method.
3	The dialog box for the subscription settings is created and initialized using the **SubscriptionSettings** auxiliary function of the SampleClient class. The current subscription settings are transferred as status.
4	Subsequently, the dialog box for the settings is opened to enable the user to make the desired settings. When the user closes the dialog box, the settings are determined from the dialog box.
5	The **SetState** method is used to change the subscription status to the new settings.

Changing the subscription filters

The sequence diagram below shows the sequences when the user of the application executes the “Change Subscription Filter” action. This action can be started via the Subscription menu.

Figure 4-8

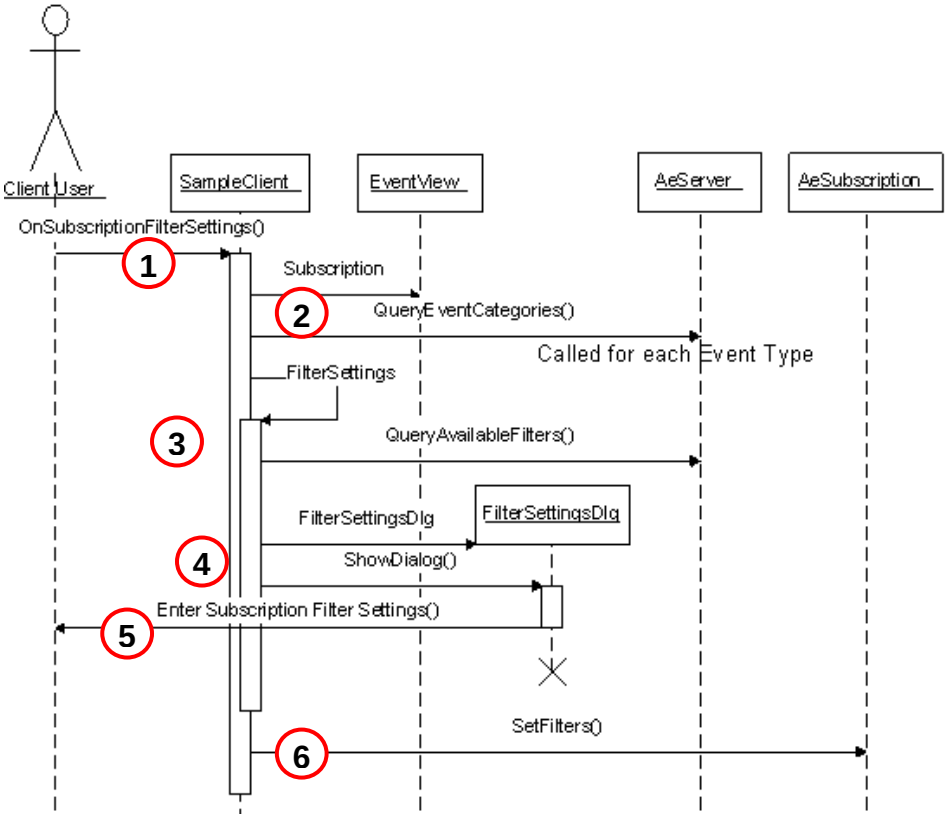


Table 4-14

No.	Description
1	The **OnSubscriptionFilterSetting** method on the SampleClient object is called by the “Change Subscription Filter” user action.
2	Using the **QueryEventCategories** method, the event categories are determined by the server. The method is called for each event type.
3	The **FilterSettings** auxiliary function is used to determine the new settings. The possible filter settings are polled by the server via **QueryAvailableFilters**.
4	The dialog box for the filter settings is generated and initialized with the information from the server. The information determined in step 2 and step 3 is transferred.
5	Subsequently, the dialog box for the filter settings is opened to enable the user to select the desired filters. When the user closes the dialog box, the settings are determined from the dialog box.
6	The **SetFilter** method is used to change the subscription filter settings to the new settings.

5 Explanations of the S7 Sample Program

5.1 Alarms of the SIMATIC S7 station

This chapter shows the possible event messages of a SIMATIC S7 station and describes their display in OPC Events in greater detail.

Categories

Event messages that can be signaled by an S7 station are divided into four Categories:

- Simple alarms (Category 2)
- Diagnostic alarms (Category 12)
- Programmed alarms (Category 13)
- Connection alarms (Category 14)

Event types

The OPC Alarm & Events specification defines three event types; the SIMATIC NET OPC server supports two of these types. The events are displayed according to the following table:

Table 5-1

OPC event type	OPC.SimaticNETAlarms.1
Simple Events	Simple alarms (SIMATIC S7 simple alarm) Diagnostic alarms (SIMATIC S7 diagnosis event)
Conditional Events	Programmed alarms (SIMATIC S7 programmed alarm) Connection alarms (SIMATIC S7 statepath alarm)
Tracking Events	Not supported

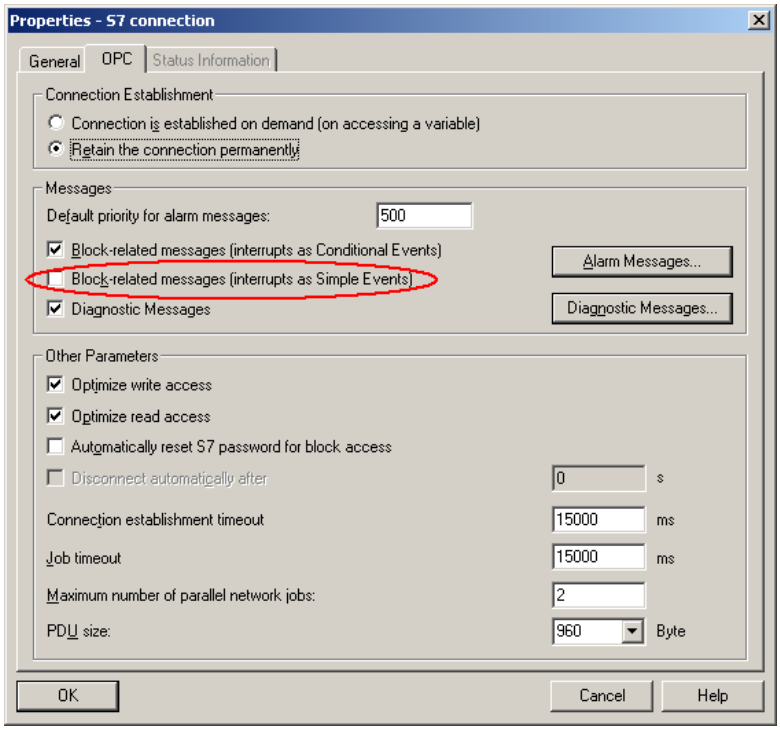
Simple alarms

The SIMATIC NET A&E OPC server offers simple alarms of the “SIMATIC S7 simple alarm” type with clearly reduced information content only for compatibility reasons with earlier SIMATIC NET software versions. The use of these alarms is not recommended and should be disabled in the S7 connection configuration (NetPro). Nowadays, corresponding alarms are sent as conditional events and it is absolutely necessary to avoid “double” sending.

Note Disable simple alarms in the connection configuration (NetPro) to avoid “double” signaling of the same event. If possible, use ALARM_D and the acknowledgeable ALARM_DQ variant instead of the older variants.

The figure below shows the disabling of the alarms of the “SIMATIC S7 simple alarm” type in the Properties dialog box of the connection configuration.

Figure 5-1



Diagnostic alarms

Events of the diagnostic alarm class are independently triggered by the S7 CPU or CP and filled into the diagnostic buffer of the respective component. The cause can be determined by means of the hexadecimal ID number of the diagnostic event using the module description (for example, cold restart/warm restart request, restart, I/O access error, etc.)

Programmed alarms

As part of the operating system, the SIMATIC S7 controller family offers system function blocks (SFB/SFC) capable of sending events via the S7 protocol. Scope and type of these SFC depend on the respective S7 CPU type. These blocks have to be called in the S7 control program (e.g., STL code) to trigger a corresponding alarm. This is the reason for the designation “SIMATIC S7 programmed alarm”.

The following table contains all block-related S7 events. The names of the blocks and their core functions are listed in the table. The listed blocks differ in the number of channels (number of monitored signals) and in the number of possible associated values that can be included in the transfer. Furthermore, some alarms can be acknowledged, others can’t.

Table 5-2

Event	Description
ALARM_8 (SFB34)	8 channels, acknowledgeable, no associated values
ALARM_8P (SFB35)	8 channels, acknowledgeable, up to 10 associated values
NOTIFY (SFB36)	1 channel, non-acknowledgeable, up to 10 associated values
ALARM (SFB33)	1 channel, acknowledgeable, up to 10 associated values
ALARM_S (SFC18)	1 channel, non-acknowledgeable, 1 associated value
ALARM_SQ (SFC17)	1 channel, acknowledgeable, 1 associated value
AR_SEND (SFB37)	For sending archives
NOTIFY_8P (SFB 31)	8 channels, non-acknowledgeable, up to 10 associated values
ALARM_DQ (SFC 107)	1 channel, acknowledgeable, 1 associated value
ALARM_D (SFC 108)	1 channel, non-acknowledgeable, 1 associated value

The S7-300 does not have the full scope of alarm functions.

Note SIMATIC S7-300 supports only ALARM_S and ALARM_D and the acknowledgeable ALARM_SQ and ALARM_DQ variants.

Each time the status of one of the monitored channels changes, an alarm is triggered and sent (rising and falling edge of a channel input form an incoming and outgoing event). The duration of a pending alarm is referred to as an alarm cycle, thus the time between rising and falling edge of the signal input while the signal input (SIG) of the block has the “high” (true) status. During this time the alarm occupies system resources, its status and time stamp are kept in the memory and can, for example, be polled by a refresh. When the state machine has been completely processed, thus an acknowledged alarm event has “gone” and accordingly the SIG input has fallen to “low” (false), the S7 CPU “forgets” this alarm and releases the resource. No history is kept in the controller.

Connection alarms

The connection or statepath alarm class is not initiated in the S7 station but in the actual OPC A&E server. A failure of an S7 connection or an interruption of the connection (for example, CP goes to stop or a cable is removed) is detected by the OPC server, a corresponding alarm generated and sent to all accordingly registered clients.

5.2 Display of the attributes in OPCEventNotification

The OPC Alarm & Events specification defines attributes that must be contained in an OPC event notification and attributes that have to be additionally included depending on the alarm type (event type). Furthermore, there are attributes that can be optionally (manufacturer-specifically) included.

This chapter describes the OPC attributes and their contents according to the S7 alarms.

Standard attributes of OPCEventNotification

The following table shows all attributes that are supplied with each event:

Table 5-3

OPC attribute	Content (description)
Source	<S7:\S7_Connection_1>
Time	<S7 time when the alarm was called> (local PC time is only displayed for statepath)
Type	<SIMPLE> or <CONDITION> (no Tracking Events)
Category	<Number> 2=Alarm, 12=Diagnosis, 13=Programmed, 14=Statepath
Severity	<1..1000> (default=500, can be set via NetPro, S7 connection configuration or at the block)
Message	<Text#> The text corresponds to the Category and EventID. The text can be changed when using version 6.4 + HF2 or higher. The configuration is described in chapter 7.3.

Note

For some blocks, e.g. ALARM, ALARM_8P or NOTIFY, the severity (0-127) is specified directly at the block in the S7 program; this severity wins through against the default priority for alarm messages that can be configured in NetPro and is automatically converted to OPC severity (1000-1) according to linear conversion. The highest block severity “0” corresponds to the highest OPC severity “1000”.

The alarm severities for specific alarm numbers configured in NetPro win through against the general default severity for alarm messages and also against programmed message weights.

ALARM_S/D and ALARM_SQ/DQ have no severity so that the configured message weights are always used.

Additional attributes of OPCEventNotification for conditional events

The following table shows attributes that are supplied for “conditional events” in addition to the ones shown table 5-3.

Table 5-4

OPC attribute	Content (description)
ConditionName	<PermanentText#> (Specified text, consisting of Category & EventID)
SubConditionName	<ConditionName>
ActiveTime	<S7 time when the condition has occurred> (Local PC time is displayed for statepath alarms)

Attributes for Category 2 (Alarm)

Alarms of Category 2 are not to be used; for this reason, their attributes are not described in detail.

Attributes for Category 12 (Diagnosis)

In manufacturer-specific attributes, diagnostic alarms of Category 12 provide additional information as shown in the following table:

Table 5-5

Attribute ID	Content = example (description)
0xFF FF FF FB	<Event Type> (VT_I4) =“16” (simple)
0xFF FF FF FA	<Severity> (VT_I4) =“500”
0xFF FF FF F7	<Category Identifier> (VT_I4) =“12”
0xFF FF FF F6	<Category Description> (VT_BSTR)=“SIMATIC S7 diagnosis event”
0x00 00 17 70	<PC Time> (VT_Date)
0x00 00 17 71	<S7 Time> (VT_Date)
0x00 00 17 A0	<Diagnostic ID> (VT_I4) =“12345” (decimal)
0x00 00 17 A1	<Diagnostic Data> (Array of VT_UI1)=“255|70|199|114|67|4|8|20|23|20”

Attributes for Category 13 (Programmed)

In manufacturer-specific attributes, programmed alarms of Category 13 provide additional information as shown in the following table. Up to 10 associated values (each with data type, length and the actual data) can be included in EventNotification as attributes.

Table 5-6

Attribute ID	Content = example (description)
0xFF FF FF FB	<Event Type> (VT_I4) =“32” (conditional)
0xFF FF FF FA	<Severity> (VT_I4) =“500”

Attribute ID	Content = example (description)
0xFF FF FF F7	<Category Identifier> (VT_I4) ="13"
0xFF FF FF F6	<Category Description> (VT_BSTR)="SIMATIC S7 programmed alarm"
0x00 00 17 70	<PC Time> (VT_Date)
0x00 00 17 71	<S7 Time> (VT_Date)
0x00 00 17 72	<Status> (VT_UI2) ="0"
0x00 00 17 73	<Acknowledgement status> (VT_UI2) ="256"(unack);"257"(ack)
0x00 00 17 74	<Event State> (VT_UI2) ="0"
0x00 00 17 75	<# Additional Data> (VT_UI2) ="1"
0x00 00 17 76	<Data Type> (VT_UI2)
0x00 00 17 77	<Length> (VT_UI2)
0x00 00 17 78	<Data> (array of VT_UI1)
to 0x00 00 17 9C	<Data Type><Length><Data> (a triplet per associated value, as above)
0x00 00 17 9E	<Event ID> (VT_UI4) ="5"
0x00 00 17 9F	<Subevent ID> (VT_UI4) ="3"

Attributes for Category 14 (Statepath)

In manufacturer-specific attributes, connection alarms of Category 13 provide additional information as shown in the following table.

Table 5-7

Attribute ID	Content = example (description)
0xFF FF FF FB	<Event Type> (VT_I4) ="32" (conditional)
0xFF FF FF FA	<Severity> (VT_I4) ="500"
0xFF FF FF F7	<Category Identifier> (VT_I4) ="14"
0xFF FF FF F6	<Category Description> (VT_BSTR)="SIMATIC S7 statepath alarm"

5.3 User program of this example

Introduction

A block call that is preferably executable in all S7 CPUs and that illustrates the basic functions of the “programmed alarms” was selected for this application example. For reasons of clarity, the status word was not evaluated.

Note To demonstrate the functionality, the block is called cyclically in OB1 as an example. Depending on the desired application, a call in the time-controlled OB35 or in other OBs (e.g., OB40) is advisable.

The required STL code fragment is executable in the S7-300 and in the S7-400 and is thus used as a general example.

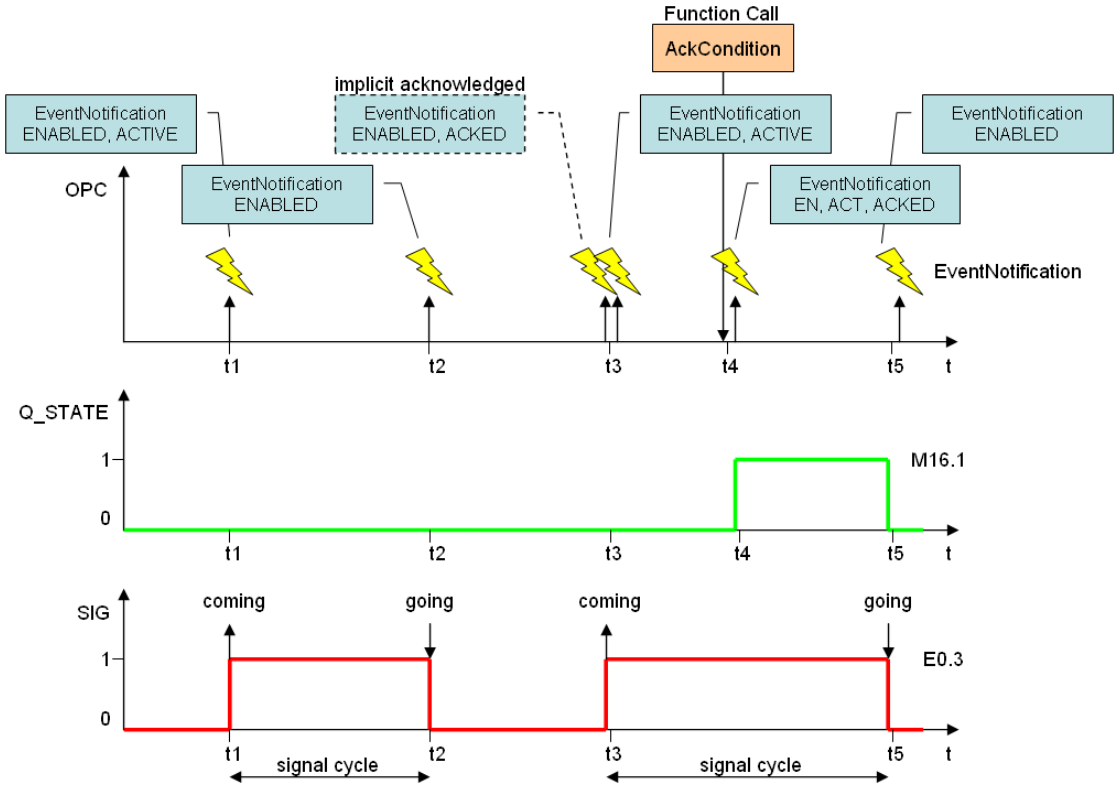
User program in OB1

```
// Connection of the channel input
A       I0.3        // Physical input
O       M10.0       // Simulation via memory bit
=       M10.1

// Cyclic call of the alarm block: SFC107
CALL  "ALARM_SQ"
SIG    :=M10.1                  // Triggers the alarm
ID     :=W#16#EEEE              // System selection
EV_ID  :=DW#16#5                // Unique number in this CPU
SD     :=DB1.DBB0               // Pointer to the associated value
RET_VAL:=MW12                   // Error code

// Cyclic call of the acknowledgement evaluation: SFC19
CALL "ALARM_SC"
EV_ID:      DW#16#5     // ID of the alarm to be scanned
RET_VAL:    MW14        // Error code
STATE:      M16.0       // Signal status (here as for M10.1)
Q_STATE:    M16.1       // Acknowledgement status
                        (“1”=Acked)
```

Signal status overview

The graphic representation below shows the most important signals of the block calls in the user program according to the above example and the associated reaction of the SimaticNET OPC A&E server.

Figure 5-2



Each status change triggers a notification. The status of an OPC alarm can only change within the state machine specified by OPC. When an alarm comes (t1: SIG = 1) and goes without being acknowledged (t2: SIG = 0), this alarm is implicitly signaled as acknowledged the next time it occurs. Simultaneously the alarm enters a new signal cycle (here t3: SIG = 1) and is signaled as “came in” (SIG = 1). When the alarm is acknowledged while the signal is present (t4), the OPC server supplies an acknowledgement notification (ACKED). The current status of each notification can be read out via a bit mask.

The A&E client of this application example uses this information to color the alarms in the view (e.g., an active, pending alarm is “red”) or to display specific symbols (for example, green checkmark if the alarm has been acknowledged).

5.4 Call of an ALARM_8P as an example

Introduction

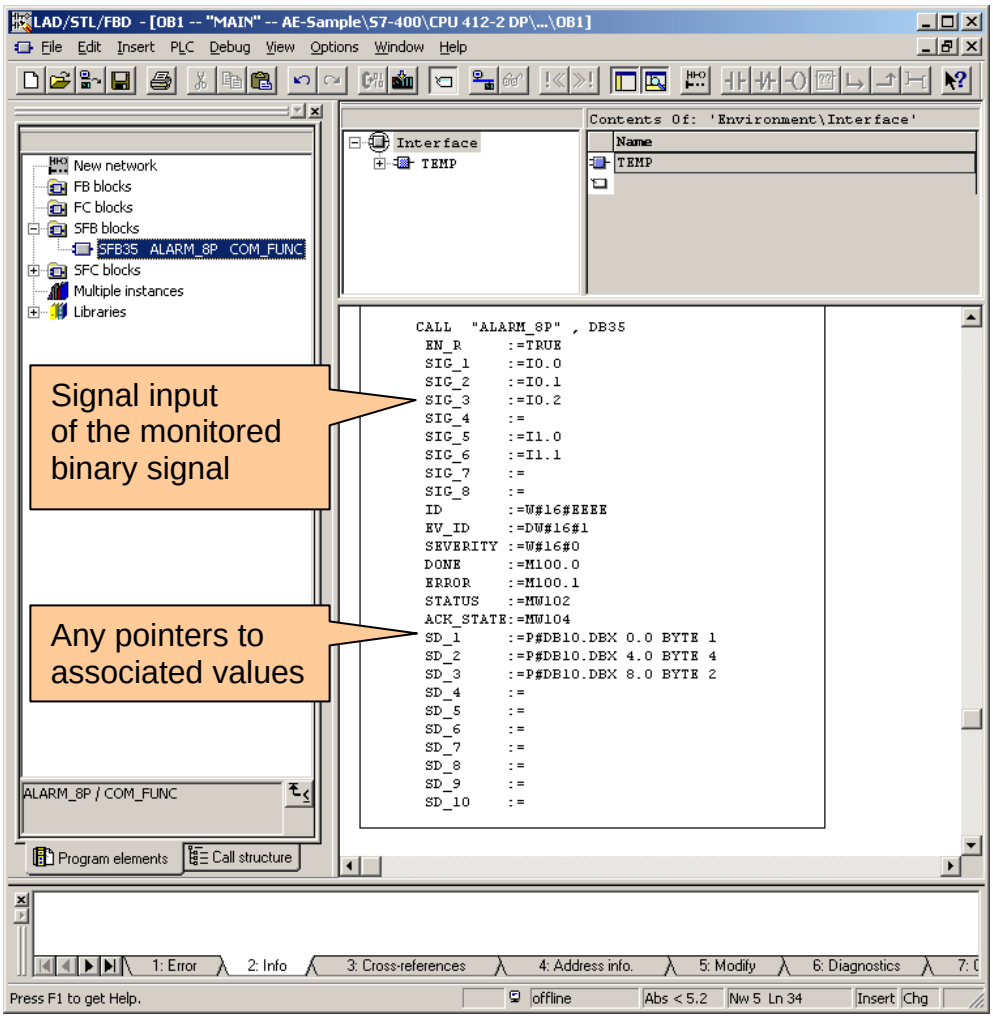
The most frequently used alarm block in S7-400 stations is SFB35 (ALARM_8P). This block provides the maximum functionality possible with regard to Alarm and Events. The call parameters are explained as examples and their meaning for OPC Events is explained.

Note The block is only available in S7 400 CPUs. In CPU Configuration (HW Config), the “Acknowledgement Triggered Reporting” setting has to be deactivated; this is the only way to ensure that ALARM_8P actually sends alarms.

Note To demonstrate the functionality, the block is called cyclically in OB1 as an example. Depending on the desired application, a call in the time-controlled OB35 or in other OBs (e.g., OB40) is advisable.

The figure below shows the block call in STL code of an S7-400. An instance data block (here DB35) is generated that contains the local data of the call. For reasons of clarity, error bit and status word were not evaluated. For a detailed description of the parameters, please refer to the STEP7 online help.

Figure 5-3



Channel parameters

For example, the inputs of an I/O module are connected to the signal inputs (also channels) SIG_1 to SIG_8 of the block. As soon as one of the signal inputs changes its status, an alarm is triggered. A positive change is assessed as an “incoming” event and a negative change as an “outgoing” event.

Management parameters

The event number (EV_ID) uniquely identifies the block for the entire controller and is assigned by Step7 to ensure consistency (“0” is not permitted)

The severity (also weighting) of the alarm is set at the SEVERITY parameter with a range of values from “0” to “127”, a low number representing a high severity.

The current acknowledgement status of the individual channels is represented in the ACK_STATE parameter. The bit array shows a “1” for

acknowledged and a “0” for not acknowledged. Bits 0 to 7 are required for “incoming” events and bits 8 to 15 for “outgoing” events of the 8 channels.

Note Unlike the S7 CPU, the OPC A&E state machine does not allow the separated acknowledgement of signal edges that “came in” and “went out”. Via OPC, the signal status can be acknowledged only once.

Associated value parameters

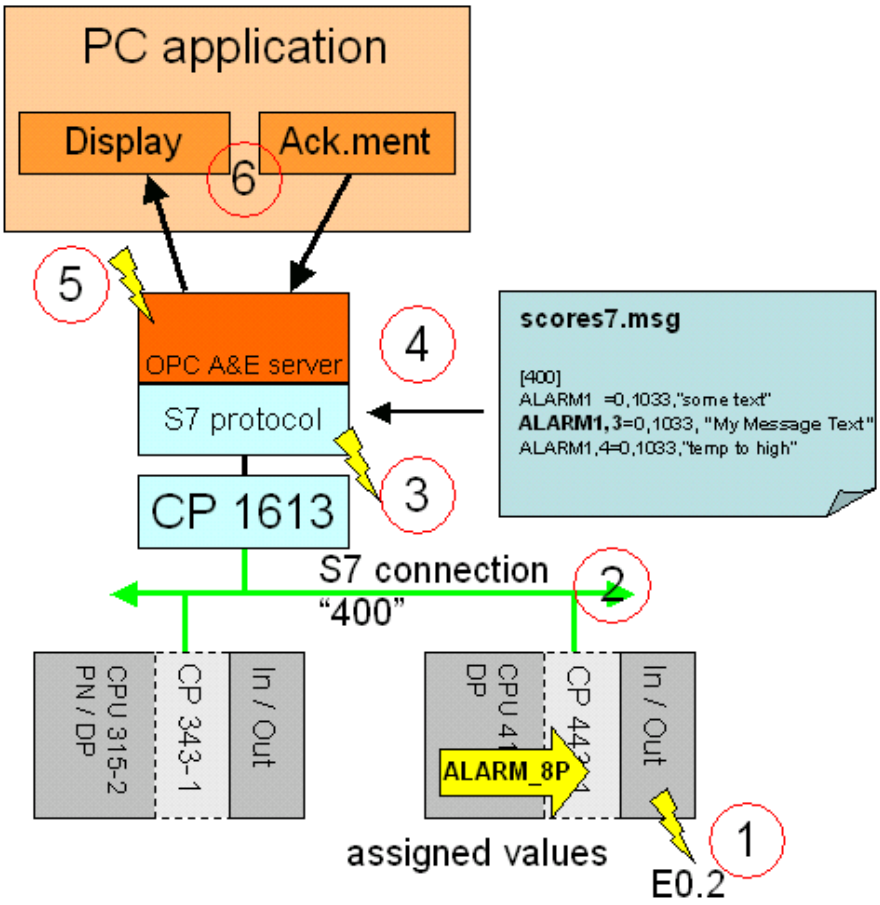
Up to 10 associated values can be parameterized. These values are ANY pointers that point, for example, to the DB10 data block as shown here. The value included there is supplied as an associated value when triggering the alarm.

Note All currently pending associated values are always included in the sending, irrespective of the specific channel (signal input) that triggered the alarm.

Response

The figure below shows the response of the components when an alarm is triggered:

Figure 5-4



The following table describes the sequences between the components when an alarm is triggered.

Table 5-8

No.	Description
1	The status of the E0.2 input changes from “0” to “1”. A call of the **ALARM_8P** block is triggered.
2	Via the S7 protocol, the S7 station sends an event to the connection partner (the PC station). The SIMATIC NET OPC server identifies the alarm by its origin (S7 connection) and its event ID (here “1”). Furthermore, the triggering channel (here SIG_3) is known (see parameters of the block call).
3	The OPC server now checks the received parameters and maps them to an **OPC Event Notification**. The “ALARM1,3” identifier (event ID=1 and signal =3) is assigned to the event and the OPC attributes are filled. The S7 block severity is converted to the OPC event severity (here: “0” becomes default = “500”). The S7 connection name (here “S7:\400”) is entered as **OPC Event Source**, the Time and ActiveTime parameters and the OPC Category are filled.

4	Before the alarm is now signaled to the OPC clients, the **scores7.msg** file is searched to determine whether a message text is stored for “ALARM1,3”; if this is the case, this text is copied into the **OPC Message** parameter. If required, associated values are extracted, formatted and inserted into the text.
5	If not prevented by a filter criterion, the notification is sent to the OPC client.
6	The event is displayed in the OPC client. The status is ACTIVE and ACK_REQUIRED, this corresponds to “came in” and “requiring acknowledgement”. When the alarm is acknowledged in the client, the OPC A&E server sends a message to the S7-400 CPU. The status can be checked in the flag word.

Note A separate acknowledgement for the “incoming” and “outgoing” edge of the signal is only possible within the S7 CPU. Via OPC Alarm and Events, merely the signal status can be acknowledged. When the alarm has “come in” (signal input to “true”) and changes to the “went out” status without being acknowledged (signal input to “false”), the event is implicitly acknowledged the next time it occurs.

Note The designation of the alarm with “ALARM<EV_ID>,<SIG#>” does not exist for the first channel (SIG_1); this channel is supplied without signal number. In the above example, 5 of 8 channels are connected and events with the following identifiers can occur: “ALARM1”, “ALARM1,2”, “ALARM1,3”, “ALARM1,5” and “ALARM1,6”. Message texts for exactly these identifiers should then be stored in the scores7.msg file.

Note For further information on the parameters of the alarm blocks, please refer to the STEP7 online help.

Structure, Configuration and Operation of the Application

6 Installation and Startup

6.1 Hardware and software installation

This chapter describes which hardware and software components have to be installed. The descriptions and manuals as well as delivery information included in the delivery of the respective products should be observed in any case.

Hardware installation

For the hardware components, please refer to chapter 2. For the hardware configuration, follow the instructions listed in the table below:

Table 6-1

No.	Action	Remark
1.	PC station: Install the CP1613 PCI plug-in card in the PC station according to the installation instructions included in the delivery.	Instead of the CP 1613, the onboard Ethernet card can also be used.
2.	SIMATIC stations: Install the S7 controllers as shown in the figure in chapter 2.	Deviations in the hardware configuration have to be considered when configuring.
3.	Connect the PC station to the two SIMATIC stations via Ethernet as shown in the figure in chapter 2.	Please note the setting of subnet masks.

Note The installation guidelines for Industrial Ethernet networks always have to be observed.

Standard software installation

For the software components, please refer to chapter 2.

Starting up the example requires the following components from the SIMATIC NET CD V6.4:

- SIMATIC NET PC products
- SIMATIC NCM PC/S7

Note You have to install SIMATIC NCM PC/S7 only if STEP 7 is not installed on the PC.

The SIMATIC STEP 7 V5.4 configuration tool is only required on the PC station if the S7 controllers are to be changed or loaded. Alternatively, this software package can also be installed on a separate computer.

Note If neither STEP 7 nor SIMATIC NET have been installed on the PC station, install STEP 7 first.

Microsoft Visual Studio .Net Professional is only required on the SIMATIC PC station if the sample code is to be changed. Alternatively, the development environment can be installed on a separate PC (e.g., engineering station).

Note During the Visual Studio .Net installation the security settings of the Windows operating system are relaxed. After installing the development environment, please check the security of the SIMATIC PC station and, if necessary, install Windows updates.

Table 6-2

No.	Action	Remark
1.	Install the SIMATIC STEP 7 software as described in the installation instructions.	Restart required.
2.	Install the SIMATIC NET PC products driver software (without NCM PC)	Restart required.
3.	Install Microsoft Visual Studio .Net	Optional

6.2 Application software installation

The user interface and the source code of the application are supplied as a ZIP file.

To start the user interface on the SIMATIC PC station, follow the steps listed in the table below.

Table 6-3

No.	Action	Remark
1.	Extract the following file: OPC_AE _Client_CODE_v10.zip	Contains the executable file and the source code.
2.	Double-click the AEClient.exe file	Starts the sample application.

6.3 Configuring the SIMATIC S7 stations

It is required that all hardware and software components have been successfully installed and cabled.

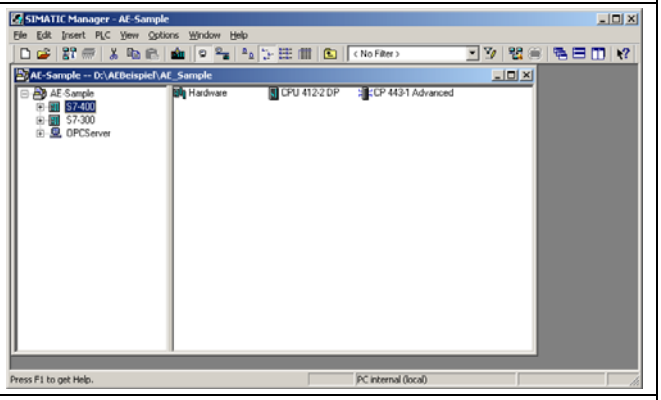
Note

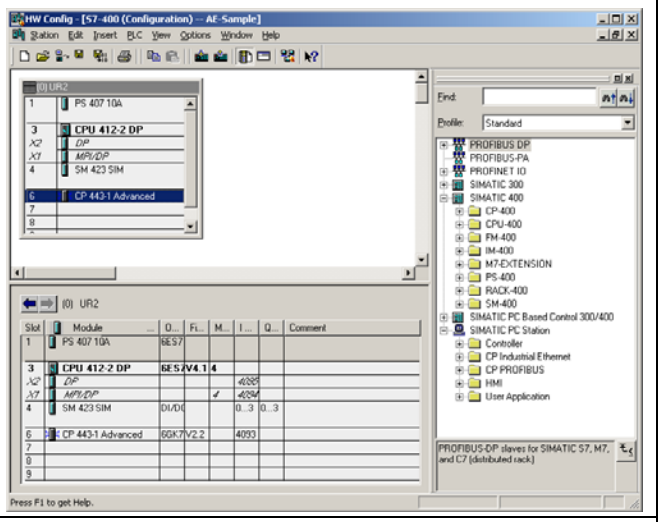
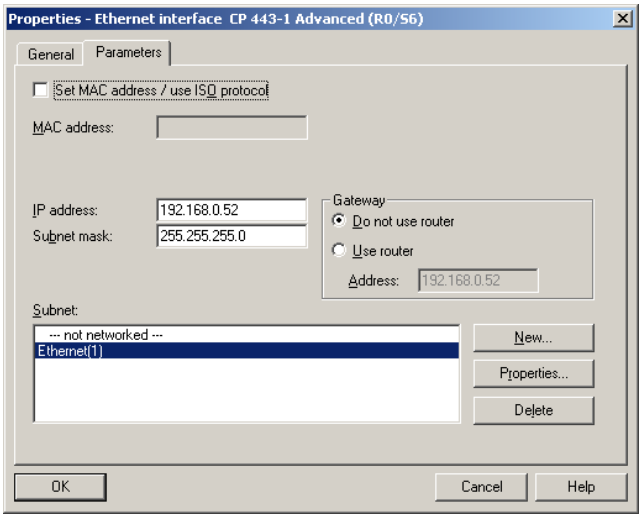
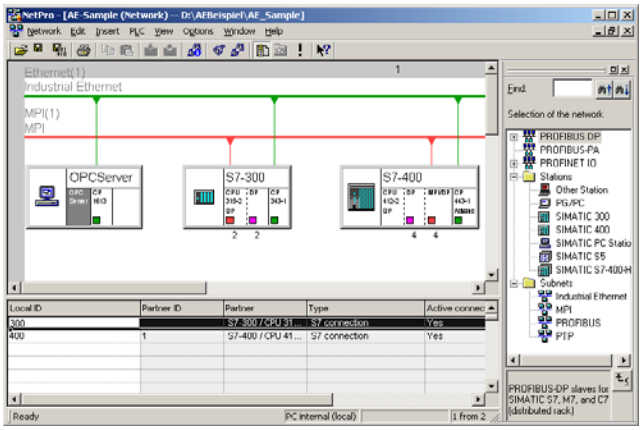
The project file delivered with this example contains the completely configured SIMATIC S7 stations according to the description in chapter 2.3. This STEP7 project can only be used without adaptation if the hardware is identical to the configuration.

If different hardware is used, the configuration of the SIMATIC stations has to be adjusted. This is particularly necessary when the hardware releases differ or when the Ethernet addresses are not identical. The STEP 7 project included in the delivery then has to be opened and adapted accordingly. After saving and compiling, all configuration information is overwritten.

The following table shows the configuration of the SIMATIC S7 station by means of download from the SIMATIC Manager.

Table 6-4

No.	Action	Remark
1.	Retrieve the project	Extract the following file: OPC_AE_STEP7_CODE_v10.zip
2.	Open the `SIMATIC Manager` by double-clicking the  icon on the desktop.	
3.	Set the `S7ONLINE` access point for STEP7. Select `Options --> Set PG/PC Interface ...`	Set the access point to the card with which you are connected to the controllers.

No.	Action	Remark
4.	Open the `HWConfig` tool to check the IP addresses and other hardware settings.	Perform this step for the S7-300 and S7-400 station and for the PC station. 
5.	Open the `Properties` of the communications processor. Adjust the IP address.	
6.	Open the `NetPro` tool to set/check the connection configuration	Perform this step for the S7-300 and S7-400 station and for the PC station. 

No.	Action	Remark
7.	Select the station to download the configuration. Click the  icon in the `SIMATIC Manager`	Perform this step for the S7-300 and S7-400 station. Select the S7 station or the OPC server and start the download. Confirm the dialog boxes with “Yes” to overwrite the station completely.
8.	Restart the modules.	The stations are restarted (warm restart). Confirm corresponding dialog boxes with “Yes”.
9.	Configure and download the PC station with the `SIMATIC Manager`	This step is not required when you import the XDB file. Alternative: Remote configuration of the PC station with the `SIMATIC Manager` is also possible performing the two steps `Configure` and `Download`. Right-click the PC station and select `PLC --> Configure...` After successful configuration select `PLC --> Download`.

6.4 Commissioning the PC station

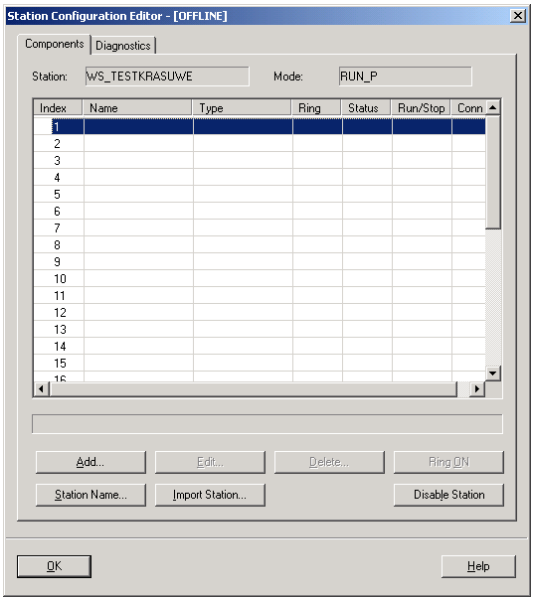
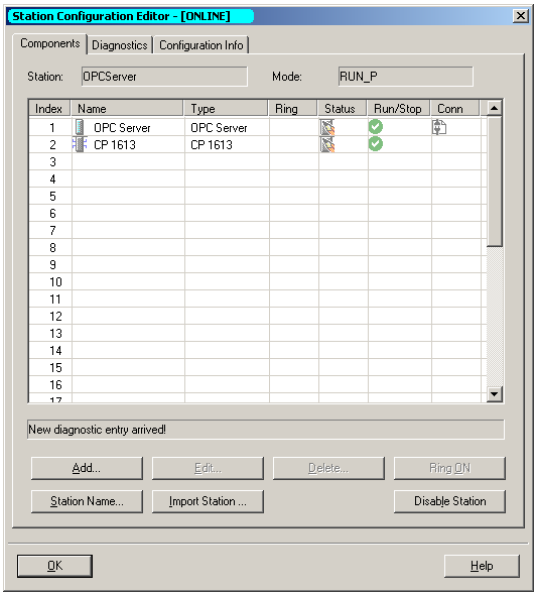
It is required that all hardware and software components have been successfully installed and cabled.

Note The project file (XDB) delivered with this example contains the completely configured PC station. This file can only be used without adjustment if the hardware is identical to the configuration.

If different hardware is used, the configuration of the PC station has to be adjusted. This is particularly necessary when the hardware releases differ or when the Ethernet addresses are not identical. The STEP 7 project included in the delivery then has to be opened and adapted accordingly (see chapter 7). After saving and compiling, all configuration information in the XDB file is overwritten.

The following table shows the configuration of the PC station by means of importing an XDB file.

Table 6-5

No.	Action	Remark
1.	Retrieve the project	Unzip the following file: 26548467_OPC_AE_STEP7_v10.zip
2.	Open the `Station Configurator` by double-clicking the  icon in the task bar.	
3.	Click `Import Station`	Confirm the following query with Yes
4.	Select the XDB file	The XDB file is located in the XDBs subdirectory of the extracted ZIP file in the directory tree of the STEP7 project.
5.	• After importing the XDB file, the PC station has been configured.	• 

7 Configuration

7.1 Configuring the SIMATIC S7 stations

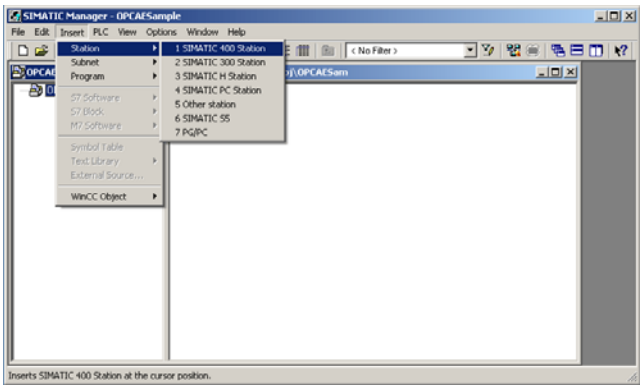
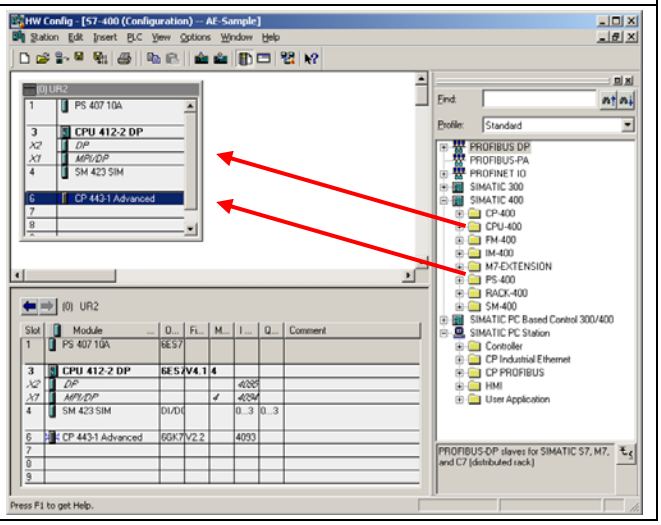
It is required that all hardware and software components have been successfully installed and cabled.

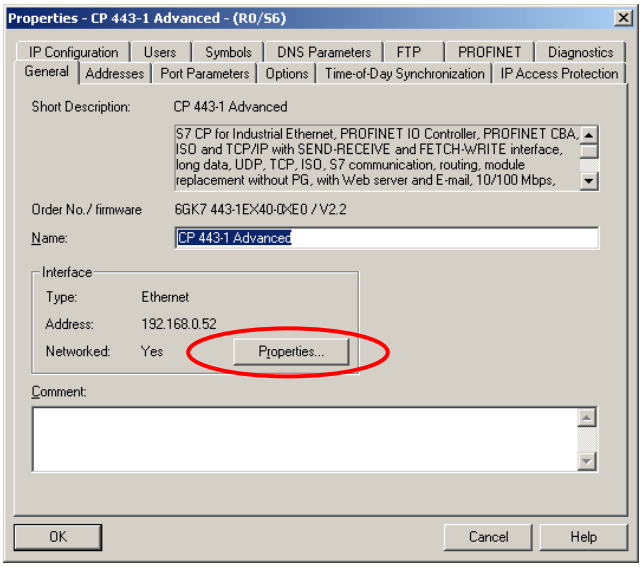
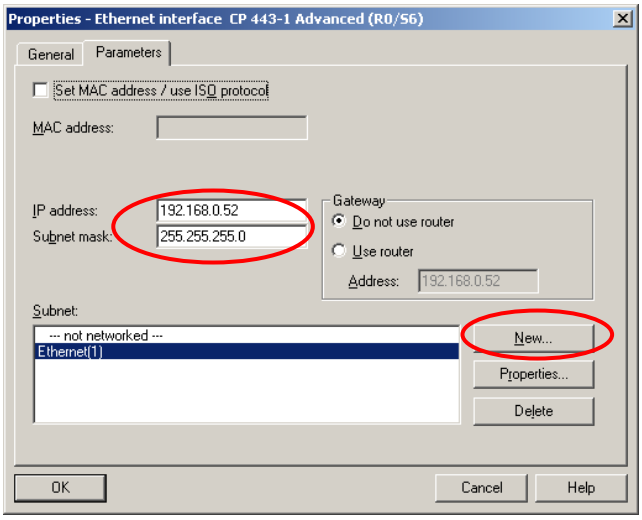
The following configuration steps of the SIMATIC S7 stations exemplify the procedure. Adjust the configuration as required for your hardware.

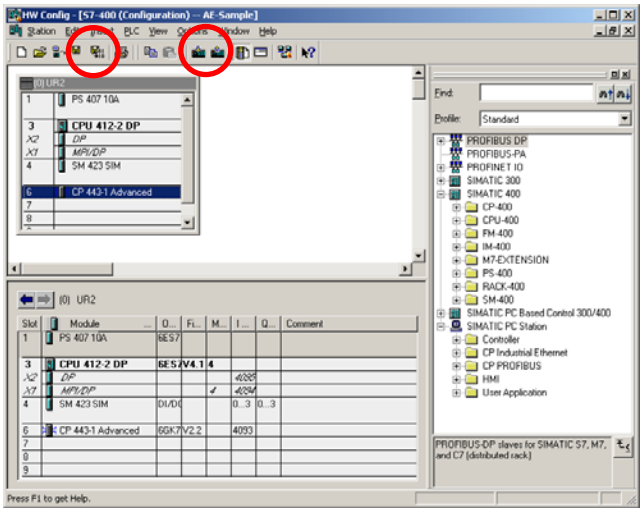
Note After saving and compiling, all configuration information is overwritten.

The following table shows the configuration of the SIMATIC S7 station.

Table 7-1

No.	Action	Remark
1.	Start STEP 7: Open the `SIMATIC Manager` and create a new project.	The name “AE-Sample” was used here.
2.	Insert SIMATIC 400 Station and assign name (here “S7-400”). Insert SIMATIC 300 Station and assign name (here “S7-300”).	
3.	Open SIMATIC stations with `HW Config` and insert CPU, CPs and other components.	

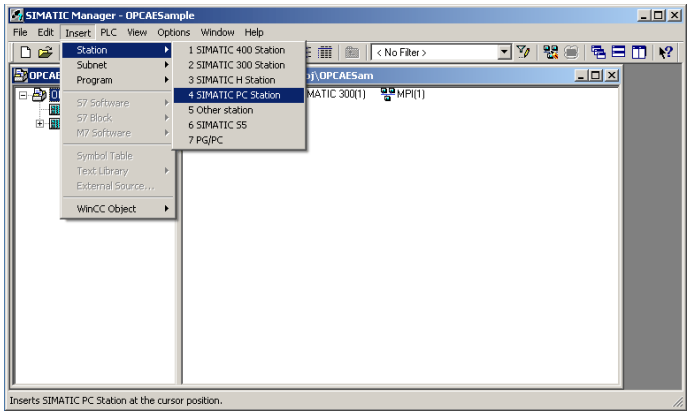
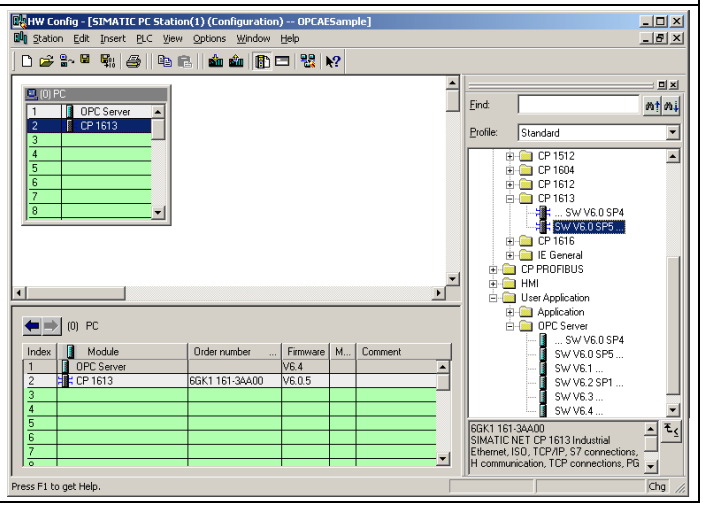
No.	Action	Remark
4.	To set the IP address, the Properties dialog box is opened.	
5.	Set the IP address (here 192.168.0.52) and the associated subnet mask. Create an Ethernet network. A mac address is entered only if the station is to communicate via ISO transport layer 4.	

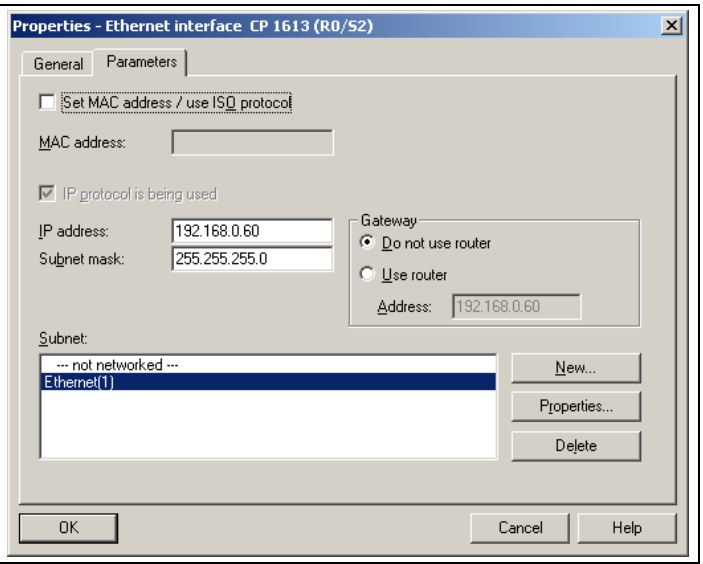
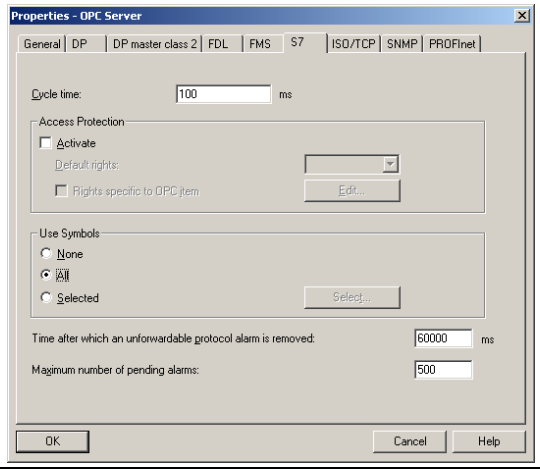
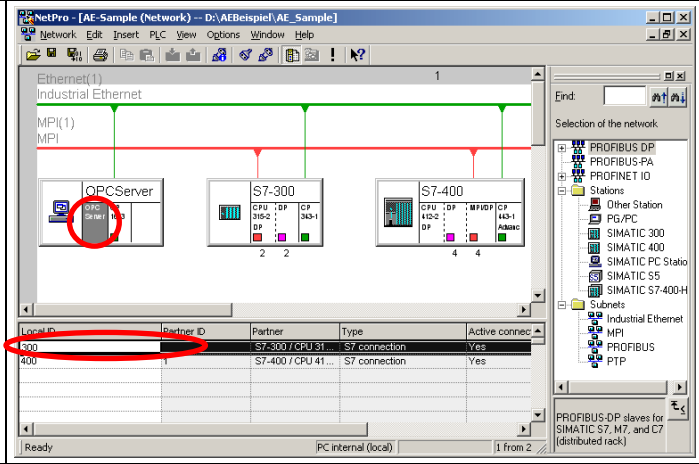
No.	Action	Remark
6.	Repeat the steps for both SIMATIC stations and then download both stations.	Perform this step for the S7-300 and S7-400 station. 
7.	Restart the modules.	The stations are restarted (warm restart). Confirm corresponding dialog boxes with “Yes”.
8.	Create S7 connections	The connection configuration is described together with the configuration of the PC station (in chapter 7.2).

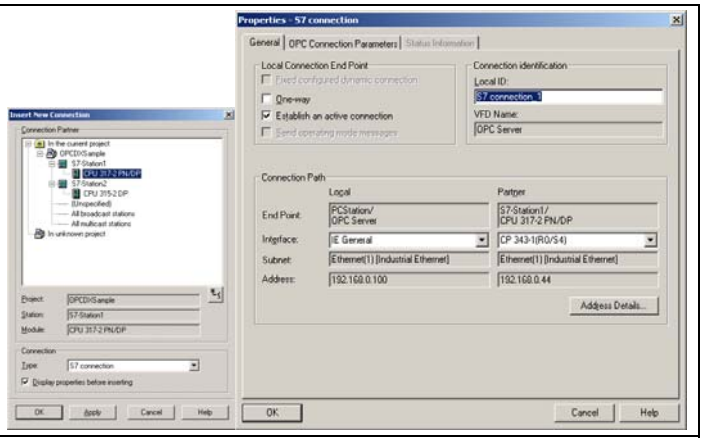
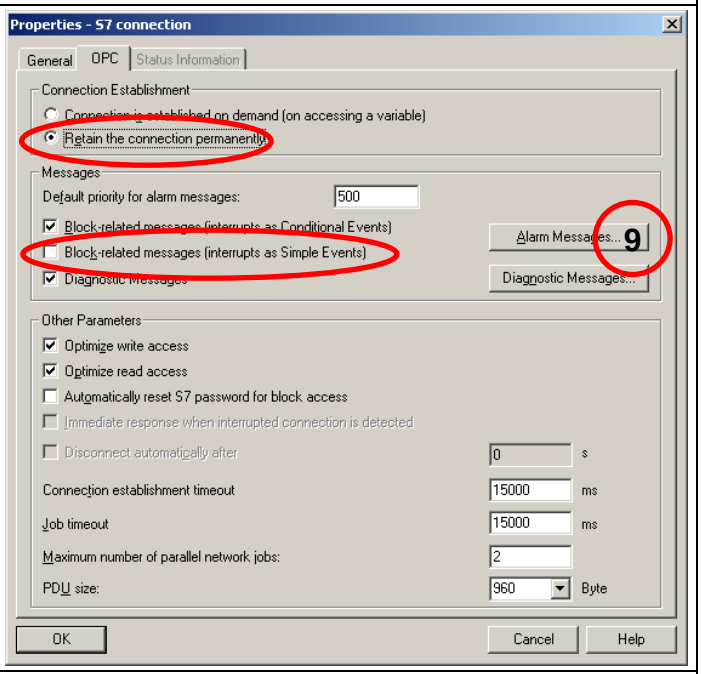
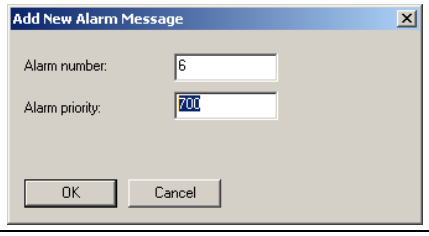
7.2 Configuring the PC station

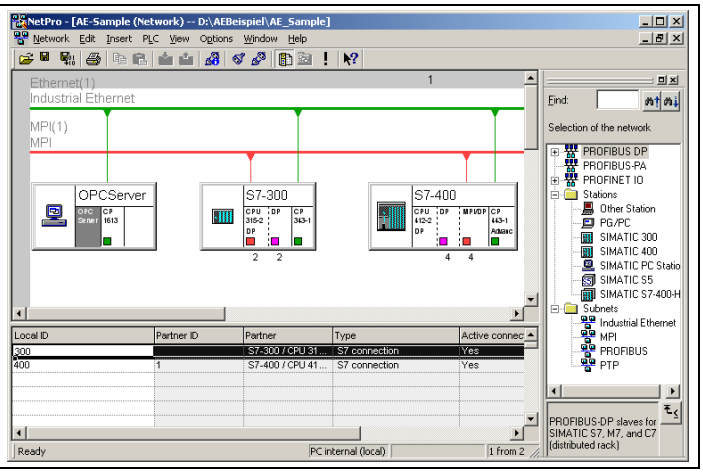
The SIMATIC PC station is configured with STEP 7 and described step by step. Alternatively, the NCM PC software package can also be used for the configuration. The procedure is identical, but unilaterally configured connections are used.

Table 7-2

No.	Action	Remark
1.	Start STEP 7: Open `SIMATIC Manager` and project.	Open the previously created “AE-Sample” project.
2.	Insert SIMATIC PC Station and specify name. The name of the PC station must be identical to the “Windows name” of the PC (see `My Computer` → `Properties` → `Computer Name`).	
3.	Open the PC Station with `HW Config` and insert the CP. The slot must be identical to the index assigned in the configuration console, here index “2” for Ethernet card. The “OPC Server” application was plugged into slot “1”.	

4.	An IP address (here “192.168.0.60”) is assigned for the Ethernet card and the card is connected to the Ethernet network.	
5.	On the OPC Server Property pages (double-click OPC Server), the use of the symbolic addressing is activated in the “S7” tab.	
6.	`NetPro` is used to create two S7 connections from the OPC server to the two controllers. Both connections are created via Ethernet. After the OPC server has been selected, the Properties dialog box is displayed by double-clicking in the list of connections.	

7.	S7 connection via Ethernet: After the connection path has been selected, the connection name can be changed (here “300” for the connection to the S7-300 and “400” for the connection to the S7-400). The connection partners and connection parameters are displayed.	
8.	In the Properties dialog box of the S7 connection, select the second tab (OPC Connection Parameter). Connection-specific settings are made here. “Retain the connection permanently” is set for the Connection Establishment to ensure that the connection is maintained also if there is no communication. The connection is configured for the transfer of “Conditional Events” and “Diagnostic Events”. Simple Events are not required since they contain redundant information and are only provided for downward compatibility with older SIMATIC NET versions.	
9.	It is usually not required to set the alarm priority. All alarms are signaled to the OPC clients with default priority “500”. The list of the configured OPC severity is displayed after selecting the “Alarm Messages...” button (see step 8)	According to a conversion table, all alarms of the SIMATIC station are mapped to the corresponding “OPC severity”. If no priority can be specified at the function block, “500” is used by default. A list of exceptions enables the user to provide individual alarms with a changed severity, this “configured severity” always prevails. 

10.	After the connections have been created, the project has to be compiled with Save and Compile. Now the stations can be downloaded. As described in chapter 6, the PC station can also be downloaded with the XDB file.	

7.3 Configuring the OPC A&E server

A configuration of the SIMATIC NET OPC A&E server is usually not required. As soon as an S7 connection has been configured for transferring events, the SIMATIC NET OPC A&E server can receive these alarms and forward them to connected clients.

If useful texts are to be sent to the clients in addition to the alarm ID numbers, another configuration step is required. Corresponding “message texts” for each alarm number are stored in a configuration file. This is possible in several languages and configured with numbers (lcids) identifying the respective languages. For each alarm sent from the S7, the server checks whether a text was configured and inserts this text into the OPC event in the language selected on the server. If no text is stored, the alarm is only forwarded with its ID number (default behavior).

Note Configuration of message texts is only available for SIMATIC NET version V6.4 + HF2 and higher.

Table 7-3

No.	Action	Remark
1.	Open the scores7.msg file with a text editor, e.g. notepad.exe	In Windows XP, the file is located in: c:\Program Files\Siemens\Simatic.Net\opc2\binS7\
2.	For each S7 connection, insert a section into the file; the section name must correspond to the S7 connection name (here “[400]” and “[300]”)	In each section, corresponding texts are then assigned to the event IDs; these texts can be stored in different languages. Syntax: `[<S7 Connection Name>]` `ALARM<event_id>=<lcids>,<lcid>,”<message text>”` Example: `[400]`

3.	Associated values of the S7 blocks can also be output within a message text. A wildcard marked with “@” is inserted into the text and the number of the associated value is entered at the block and a formatting qualifier is entered.	Blocks with associated values are, for example, ALARM, ALARM_8P, NOTIFY, NOTIFY_8P; only one associated value is possible for each block for S7-300 controllers, e.g. ALARM_S, ALARM_SQ, ALARM_D, ALARM_DQ Syntax: `@2X%f@` (second associated value as a floating point) Example: `ALARM5,3=0,"Boiler Temperatur of @1X%d@ degrees was exceeded, switch of heater @2X%u@"`

4.	Diagnostic alarms are signaled by S7 controllers and usually appear in the diagnostic buffer of the corresponding CPU.	Syntax:
		`DIAGNOSIS<hex>=<lcids>,<lcid>,”<messagetext>”`
		
		Example:
		`DIAGNOSIS0x4302=0,1033,"My Text for Diagnosis"`

5.	Further information	Further information is available in the actual scores7.msg file.

Note The scores7.msg file is read out by the SIMATIC NET OPC A&E server only during startup. Please make sure that the server restarts after changes have been made to this file.

8 Operation of the Application

You are provided with information on...

how you can operate all functions of this application. The operation described here focuses on the triggering of alarms and the default selection of associated values.

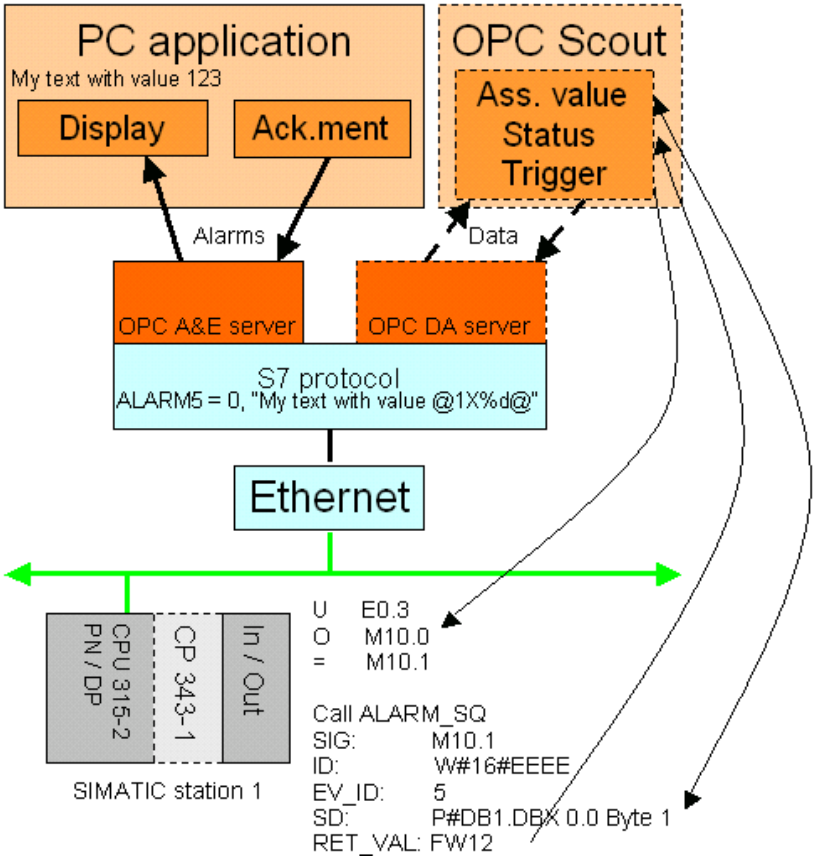
Note

To provide a simplified representation of the principle of operation, an ALARM_SQ is called cyclically in OB1. The required STL code fragment is executable in the S7-300 and in the S7-400 and is thus used as a general example.

Overview

To simulate an alarm, a memory bit was interconnected in addition to the input. An ALARM_SQ is triggered when either the 0.3 input or the 10.0 memory bit changes to “1” status.

Figure 8-1



The alarm block has received the fixed ID “5” and was interconnected with an associated value (here an Any pointer that points to the first byte in data block 1). The return value of the alarm block is written into flag word 12.

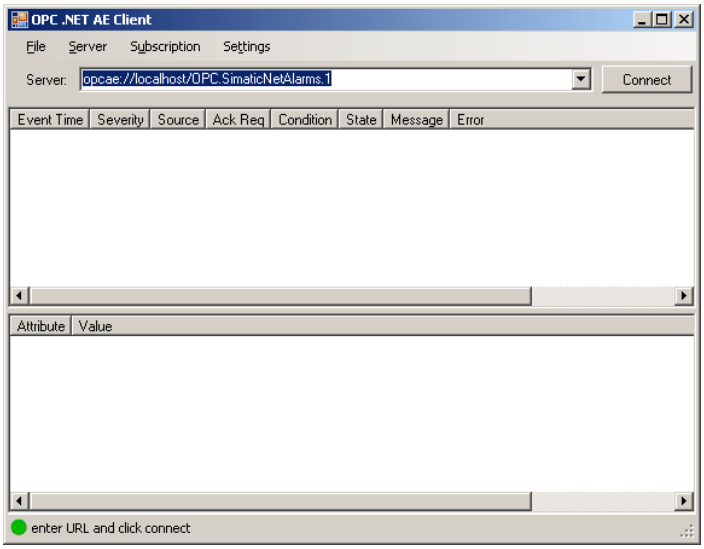
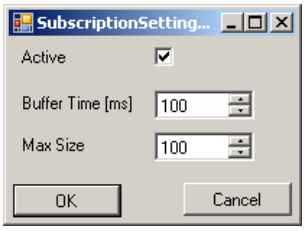
Three items are monitored in OPC Scout (OPC Data Access Client):

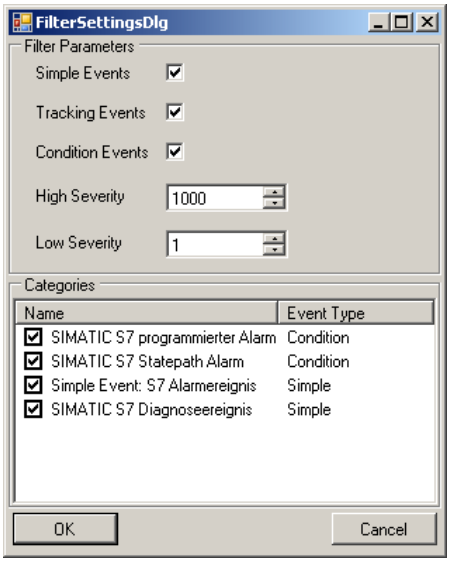
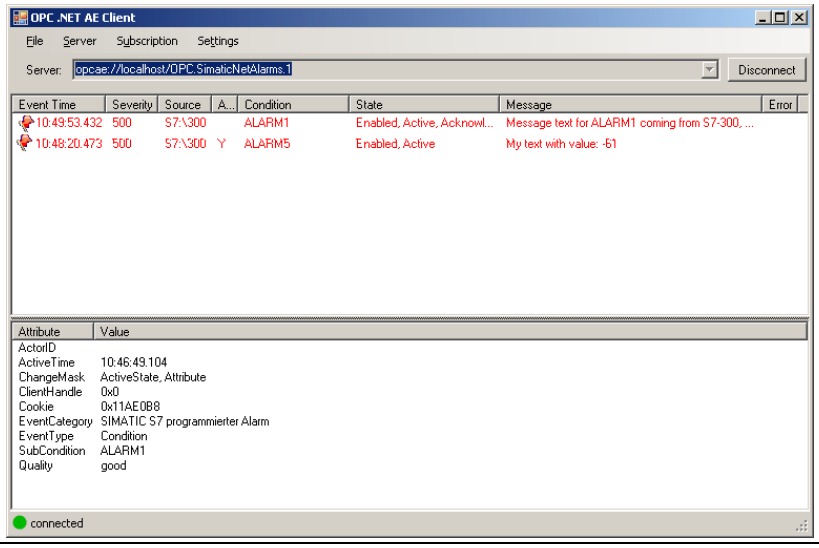
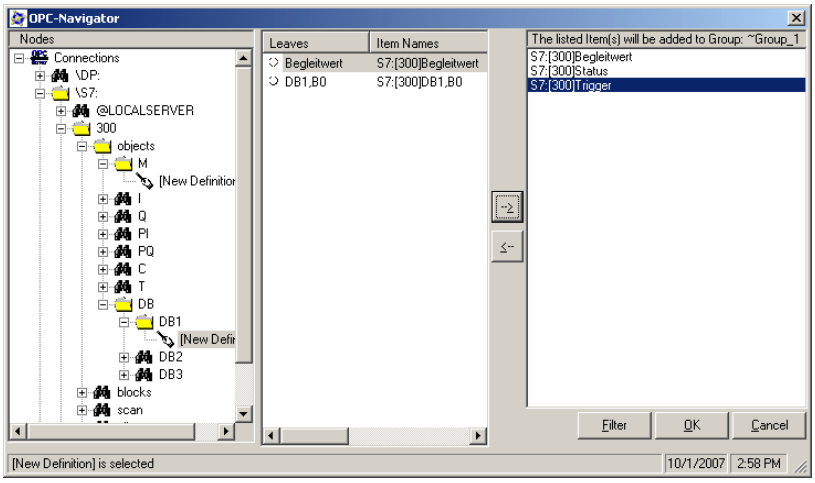
- S7:\[300]DB1,B0,1 (Begleitwert [associated value])
- S7:\[300]MW12,1 (Status)
- S7:\[300]MX10.0,1 (Trigger)

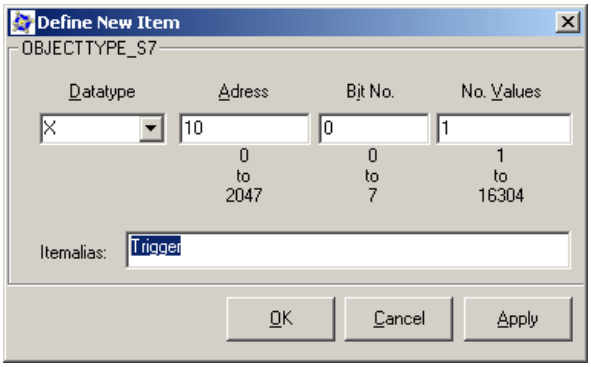
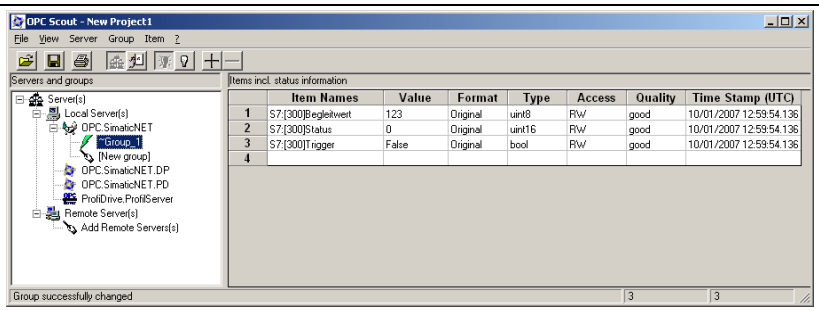
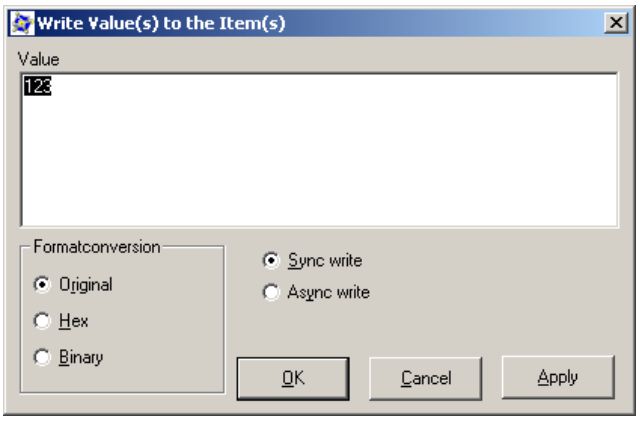
Triggering alarms

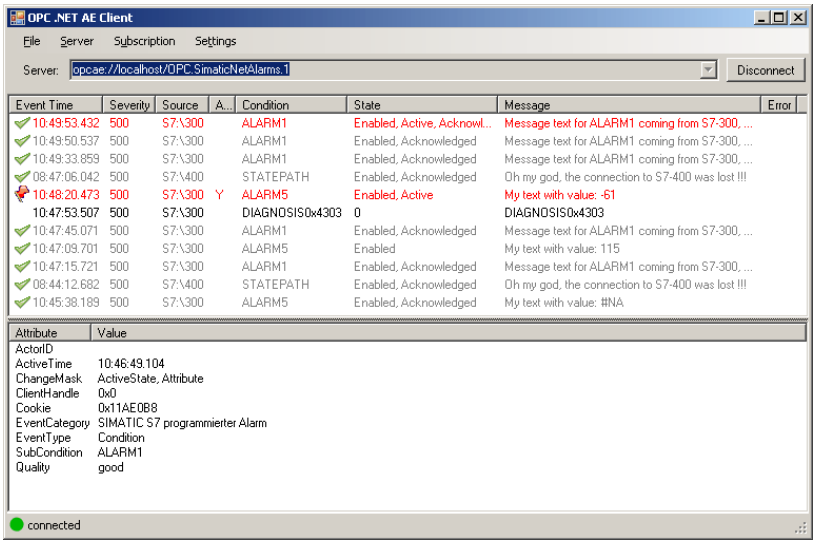
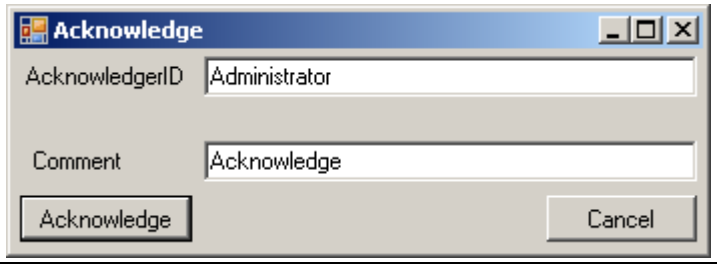
To simulate an alarm, the 10.0 memory bit can now be used in addition to the physical E0.3 input. The memory bit is written with OPC Scout (trigger). An alarm with ID number 5 is triggered when the status changes from “0” → “1”, an additional alarm is triggered in the event of a status change from “1” → “0”. This corresponds to the “came in” and “went out” principle of status-controlled alarms.

Table 8-1

No.	Action	Remark
1.	Start the OPC A&E client (this example)	
2.	Select the server	In the drop-down box of the user interface, the OPC A&E server can be selected.
3.	Connect	The Connect button is used to establish the connection to the server. After the connection has been established, two configuration dialog boxes for the subscription are displayed.
4.	Configure the subscription settings	

No.	Action	Remark
5.	Configure the subscription filter settings	
6.	Display of the active alarms	
7.	Start the OPC Data Access client (OPC Scout).	

No.	Action	Remark
8.	Create the OPC items Begleitwert [associated value], Status and Trigger	
9.	Right-click in the “Value” field to display the dialog box for writing values	
10.	Default selection of an associated value (for example, “123”) Subsequently, trigger the alarm by writing a “1” (true) to the “Trigger” item	

No.	Action	Remark
11.	Display of the alarm in the sample application	
12.	To acknowledge the alarm, select the line by right-clicking it and select the drop-down menu	

Appendix and Literature

9 Literature

9.1 Bibliographic references

This list is by no means complete and only provides a selection of appropriate sources.

Table 9-1

	Topic	Title
\1\	STEP7	Automating with STEP7 in STL and SCL Hans Berger Publicis MCD Verlag ISBN 3-89578-113-4

9.2 Internet links

This list is by no means complete and only provides a selection of appropriate sources.

	Topic	Title
\1\	Reference to the entry	http://support.automation.siemens.com/WW/view/en/26548467
\2\	Siemens A&D Customer Support	http://www.ad.siemens.de/support
\3\	OPC Data Access Custom Interface Version 3.0	Specification on the OPC Foundation website for download for OPC members www.opcfoundation.org
\4\	OPC Alarm and Events Custom Interface Version 3.0	Specification on the OPC Foundation website for download for OPC members www.opcfoundation.org
\5\	SIMATIC NET Commissioning PC Stations – Manual and Quick Start for SIMATIC NCM PC / STEP 7 version V5.2 or higher	Description of or information on: • General information on the PC tools. • Functions of NCM PC. Installed by SIMATIC NET, see: Start → SIMATIC → Documentation → English. Available in Product Support; Entry ID: 13542666 http://www4.ad.siemens.de/WW/view/EN/13542666
\6\	SIMATIC NET – Industrial Communication with PG/PC	Manual for industrial communication on PG/PC with SIMATIC NET. Installed by SIMATIC NET, see: Start → SIMATIC → Documentation → English. Available in Product Support; Entry ID: 2044387 http://www4.ad.siemens.de/WW/view/EN/2044387

10 History

Table 10-1 History

Version	Date	Modification
V1.0	10/30/07	First edition