Programming a Message System for OPC UA Alarms & Conditions with .NET C# for the SIMATIC NET OPC UA Server

SIMATIC NET OPC UA Server

Application Description • December 2011

Applications & Tools

Answers for industry.

SIEMENS

Siemens Industry Online Support

This article is taken from the Siemens Industry Online Support. The following link takes you directly to the download page of this document:

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

Caution
The functions and solutions described in this article confine themselves to the realization of the automation task predominantly. Please take into account furthermore that corresponding protective measures have to be taken up in the context of Industrial Security when connecting your equipment to other parts of the plant, the enterprise network or the Internet. Further information can be found under the Item-ID 50203404.

http://support.automation.siemens.com/WW/view/en/50203404.

If you have any questions concerning this document please e-mail us to the following address:

mailto:online-support.industry@siemens.com

You can also actively use our Technical Forum from the Service & Support Portal regarding this subject. Add your questions, suggestions and problems and discuss them together in our strong forum community:

http://www.siemens.com/forum-applications

SIMATIC
.NET OPC-UA A&C Client

OPC UA Alarms & Condition Example for
SIMATIC NET OPC UA Server

Warranty and Liability

Note

The application examples are not binding and do not claim to be complete regarding configuration, equipment 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 used correctly. These application examples do not relieve you of the responsibility to use sound practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we 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 this application example and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We accept no liability for 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.

Preface

Objective of this application

The objective of this application is to provide the programmer of an individual message system with a comprehensive overview regarding the use of the OPC UA communication interface which supplies the data, alarms and diagnostic information from the SIMATIC S7 controller. You will learn about the components used, standard hardware and software components and the specially created user software.

Document structure

The following table shows the individual chapters of this document, the discussed contents and the addressed reader.

No	Chapter	Content	Reader
1	Automation Task	Overview of the task of this example	All
2	Automation Solution	Overall solution for this example, core functionality	User, commissioner, programmer
3	Basic Information	OPC and OPC UA, specifically OPC UA Alarms&Conditions	Technician, programmer
4	Functional Mechanisms of the Client Application	Software description of the user interface, class diagrams	Programmer
5	Functional Mechanisms of the S7 Application	Alarms in the S7-300 and 400 and how they are used	PLC programmer, project planner, engineering
6	Configuration and Settings	Configuring and downloading the S7 stations and the PC stations with OPC Server	Commissioner
7	Installation and Commissioning	Installation instruction, quick start-up	User, commissioner
8	Operation of the Application	Functions of the user interface of the UA A&C client	User
9	Further Notes, Tips & Tricks, etc.		All
10	Links & Literature	Literature, internet links, further information	All
11	History	Versions oft this document	All

Table of Contents

1 Task

Reason

The OPC Unified Architecture (UA) in SIMATIC NET OPC Server provides an additional, convenient and performant option for process interfacing. This connection of PC systems to SIMATIC S7, will successively replace the existing OPC Data Access (DA) and Alarms & Events (A&E) functions.

The main advantages of OPC UA over conventional OPC interfaces are:

- Communication via the internet and across firewalls.
- Optimized, robust and fault-tolerant protocol with integrated security mechanisms.
- OPC UA can be directly integrated in applications on different operating systems with different programming languages.
- All OPC information such as data or alarms are integrated in a namespace.
- Information can be described with object-oriented means.

Target group

This application is designed for end users who need a comprehensive introduction into this technology and want to acquire experience with professional generating of OPC UA clients in C# under .NET

Content

This is where you get an overview of the use of the OPC UA communication interface which offers the data, alarms and diagnostic information from the SIMATIC S7 controllers. You will learn about the components used, standard hardware and software components and the specially created user software.

The user software offers examples for the creation of OPC UA Alarm clients with C# under .NET. Included are a simplified, reusable API and an example application for a message system with a convenient user interface. The example also provides notes on the optimization and expansion of the application.

1.1 Overview

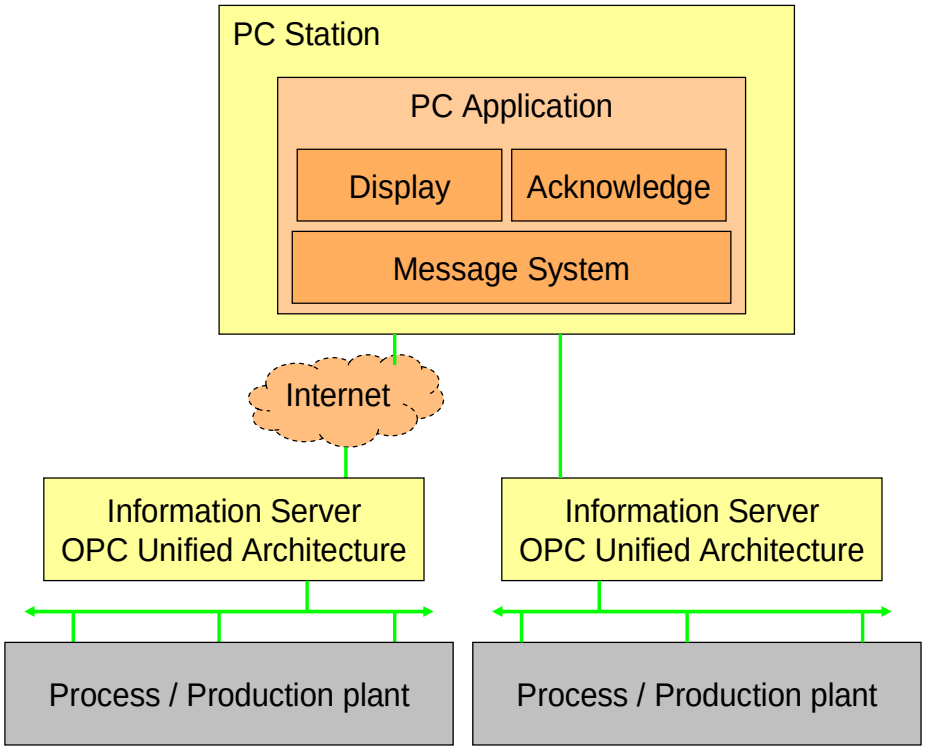
Introduction

To realize a data link, it is nowadays preferred to use standardized mechanisms in order to ensure that such a data exchange remains independent of the used bus system or protocol or even manufacturer. For the exchange of event and alarm messages, a standardized mechanism for connecting different subsystems will also be used. Apart from local communication this mechanism shall also enable information exchange via firewalls and the internet, which also requires user authentication and encoded data transmission.

Overview of the automation task

The following figure shoes an overview of the automation task.

Figure 1-1



Description of the automation task

In the automation system the OPC UA server shall be considered the information server, which can display and describe individual components but also the entire system. Through the encrypted access, which is checked and secured with certificates, a link to other locations is also possible.

The automation problem consists of acquiring event messages and alarms from the automation systems in a central message 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 shall communicate via Ethernet.

The application should contain the following functionalities:

- Server selection including security settings.
- Display of messages and alarms including accompanying values of the various stations in a list.
- Display of standard and user-defined diagnostic messages.
- Display of system-related messages
- Selecting associated values.
- If required, acknowledgement of individual messages.

Further data processing (e.g. saving in database or similar) is not described in this application.

1.2 Requirements

Requirements of the automation task

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.

Requirement for the controller

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 message system, preferably via Ethernet.

PC station requirements

The PC station must have the necessary physical connection with corresponding hardware and software to be able to communicate with the controller.

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

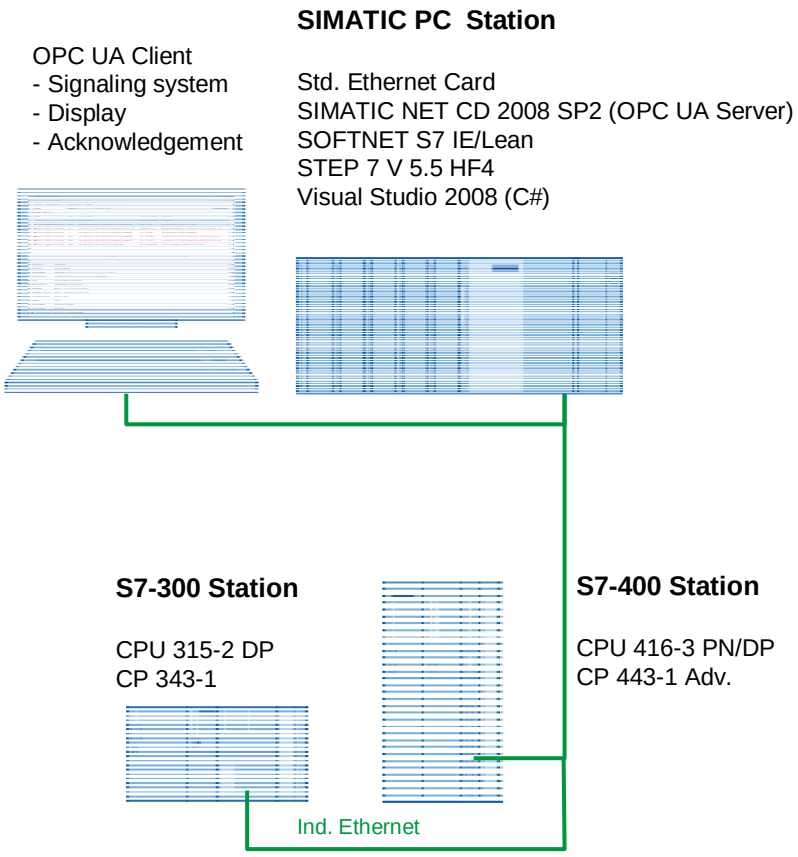
2 Solution

2.1 Overview of the overall solution

Overview

The following figure displays the most important components of the solution:

Figure 2-1



Setup

A PC station is connected to a CPU 315-2 PN and a CPU 414-2 via Ethernet. A standard Ethernet card is used in the PC.

OPC-UA client software

The OPC-UA client in the PC station realizes a message system based on OPC UA with a display of the messages and an acknowledgement option. The client with comfortable user interface demonstrates to you the professional handling of reusable C# classes for the implementation of a .NET-based OPC UA client.

The functionality of these sample clients will be explained in the next section.

2.2 Description of the core functionality

Overview

The core of the functionality of this example is the SIMATIC NET OPC UA server. It simplifies the functions and information of the classic OPC server for Data Access and Alarm & Events in one single namespace and permits access to information via a service-oriented architecture. Communication via the Internet and across firewalls is secure and performant. In terms of alarm processing it should be capable to receive messages directly from the controller without a necessary polling access to the controller.

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

Figure 2-2

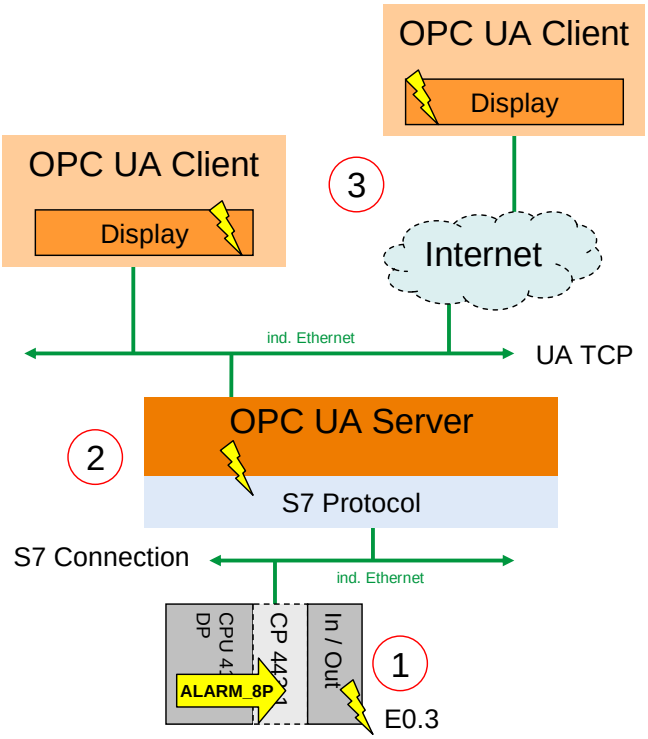


Table 2-1

No	Component	Description
1.	S7 station	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.
2.	OPC UA server	Via the S7 protocol, the S7 station sends an event to the SIMATIC NET OPC UA server. The OPC UA server sets the events to OPC UA events and alarm objects and provides the OPC services such as Browse, Read, Event Monitoring and Method calls for acknowledgement.
3.	OPC UA client	The OPC UA client can establish a secure connection with the server and receive and display events.

Software components of the application (OPC UA .NET client)

The figure below shows the software components used for the more complex application (OPC UA .NET client). The OPC UA server and the basic libraries for the OPC UA communication on the client side are from the SIMATIC NET CD.

The software components created in C# for the application can be divided in reusable modules and sample code.

Figure 2-3

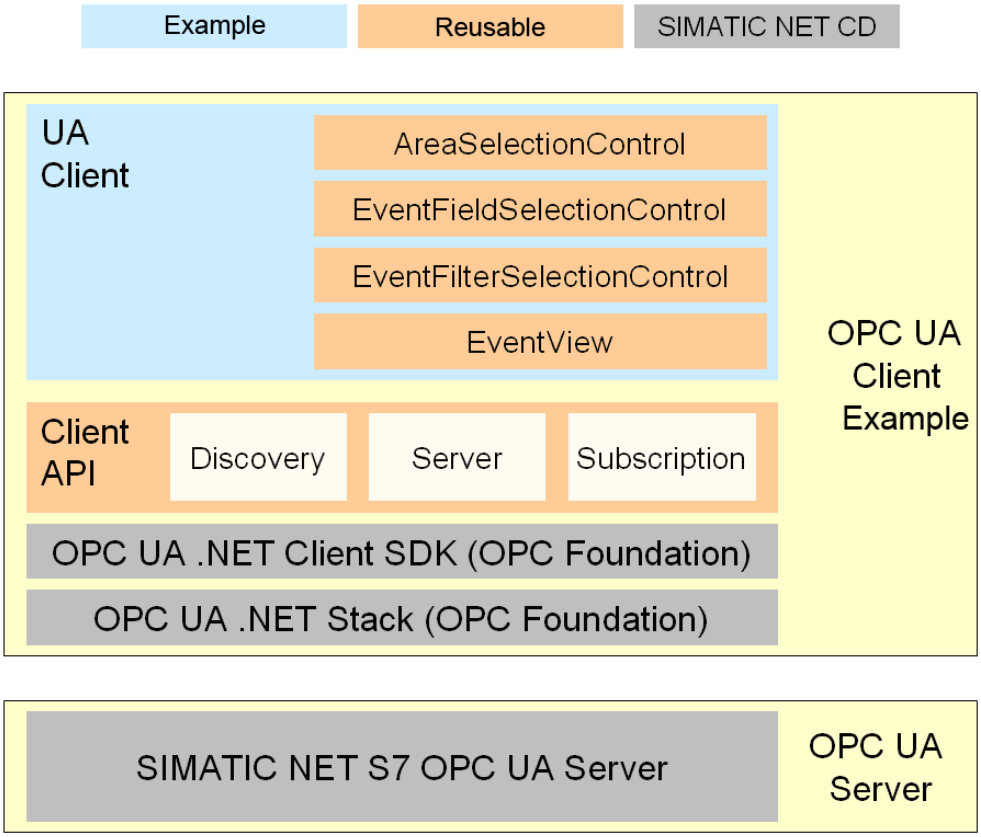


Table 2-2

Module	Description
OPC UA .NET Stack	The .NET OPC UA stack from the OPC Foundation for the realization of the network communication.
.NET Client SDK	The .NET OPC UA client SDK of the OPC foundation. The two DLLs of the OPC foundation are part of the delivery of the SIMATIC NET CD.
Client API	Reusable, simplified and tailored to the .NET Client API task. It offers reusable C# classes for discovery, session and subscription handling.
UA Client	Comfortable OPC Client with the functions discovery, connect, disconnect, monitoring of events, filter settings for events, selection of delivered event fields and acknowledgement of alarms. General functions such as browsing of areas, display of available filters and display of event fields are encapsulated in reusable controls.
S7 OPC UA Server	The SIMATIC NET OPC UA server implements the necessary server logic for sessions and subscriptions and the data connection to the S7 stations.

Overview and description of the user interface (OPC UA .NET Client)

The following figure and table describes the user interface of the OPC UA AC Client example, which can be used for receiving alarms and events from an OPC UA Server.

Figure 2-4

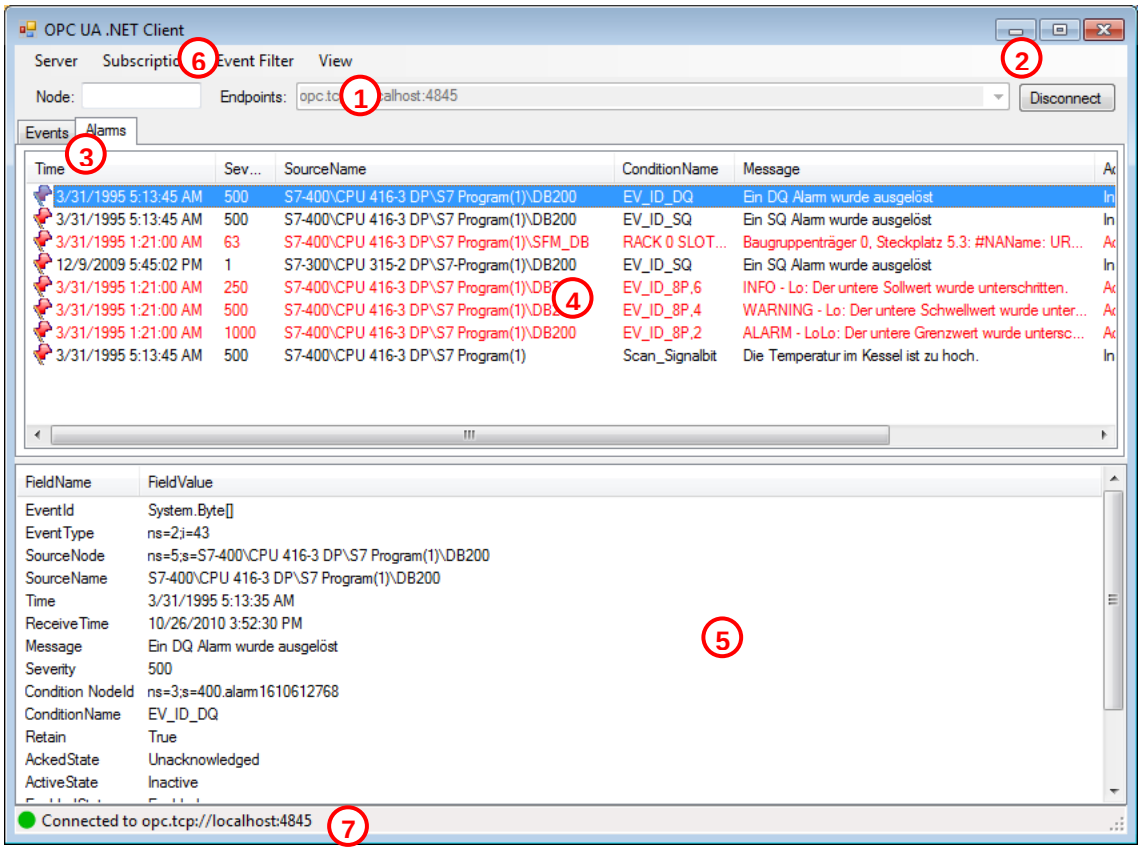


Table 2-3

No	Description
1.	The server can be selected via the **Endpoints** selection list. For this purpose the list of the available OPC UA servers from the corresponding network node is determined. The computer, from which the list is to be prompted, can be entered in the **Node** text field. If the field is empty, the list will be determined on the local computer. The URL of the OPC UA server can also be entered manually. The URL for the SIMATIC NET OPC UA server it is made up of **opc.tcp://<name of computer>:4845**.
2.	The connection to the server can be established or closed via the **Connect** button.
3.	In the main window 2 views can be toggled: Events: displays all received events. Alarms: only displays condition events (visible in the picture).
4.	List of the received condition events.
5.	Event fields for the currently selected event. The displayed event fields can be selected in the configuration dialog.
6.	The properties of the subscriptions and the configuration of the filters, areas and event fields can be changed via the application menu. This is how e.g. the publishing interval of the subscription can be changed.
7.	Current status and URL of the connection with the server.

Advantages of this solution

The solution presented here offers you the following advantages:

- Using the international OPC Unified Architecture standard
- Simple realization of a connection with different event sources
- High-performance data transmission by using the event mechanisms on all levels of the communication without a polling access
- Protection of investment by means of a simple expansion capability
- Re-usable components
- Access possible via internet and across firewalls
- Access rights can be assigned individually for users
- Security regarding certificates, encryption and authentication

Delimitation

This application does not contain a description for processing or saving data in the OPC UA client e.g. in databases.

Required knowledge

Basic knowledge of the handling of the SIMATIC configuration and programming tool STEP7 as well as of the Microsoft Visual Studio 2008 development environment and the programming language C# and object-orientated programming is assumed.

2.3 Used hardware and software components

The application was created with the following components:

Hardware components

Table 2-4

Component	No.	MLFB/order number	Note
S7-400 CPU 416-3 PN/DP	1	6ES7416-3ER05-0AB0	Any other S7-400 CPU can also be used
CP 443-1 Advanced	1	6GK7443-1GX20-0XE0	Alternatively any other S7 capable Ethernet CP can be used.
S7-300 CPU 315-2DP	1	6ES7315-2AG10-0AB0	Alternatively any other S7-300 with an Ethernet CP can be used.
CP 343-1	1	6GK7343-1EX30-0XE0	Alternatively any other S7 capable Ethernet CP can be used.
SIMATIC PC station as OPC UA server	1	6AG4104-1AA22-0BB0	Standard PC (e.g. PGs) under Windows Vista or Windows XP.
Standard PC as OPC UA client	1	6AG4104-1AA22-0BB0	Alternatively the client can also be operated locally on the SIMATIC PC station.

Software components

Table 2-5

Component	No.	MLFB/order number	Note
SIMATIC NET CD 2010 (V8.1) SOFTNET S7 IE	1	6GK1704-1LW71-3AA0 6GK1704-1CW71-3AA0	LW=8 S7 connections (Lean), CW=64 S7-connections
STEP 7 V5.5 SP1	1	6ES7810-4CC08-0YA5	For the configuration of bilateral S7 connections on S7 300 / 400 CPUs
Microsoft Visual Studio 2008 SP1	1	Standard Edition Professional Edition	Available in Microsoft Store (http://emea.microsoftstore.com/DE/Microsoft/Design-+-Entwicklung/Visual-Studio)

Example files and projects

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

Table 2-6

Component	Note
OPC_UA_CODE_v10.zip	This zip file contains the OPC UA client and the sources.
OPC_UA_DOKU_v10_e.pdf	This document.
OPC_UA_STEP7_v10.zip	This zip file contains the STEP 7 project

2.4 Alternative solutions

OPC Data Access on the basis of COM

Today, this automation task is typically solved with the COM based classic OPC Alarm & Events interface.

Advantages of the solution with COM OPC Alarm & Events:

- Extensive use of the interface.
- Many applications for different tasks support the interface.
- Easy access for local applications.

Disadvantages of the solution with COM OPC Alarm & Events:

- Complicated DCOM configuration for remote access.
- No communication possible across firewall or internet boundaries.
- OPC clients can only be operated on Windows PC systems.
- Restricted security mechanisms and user authentication only within the framework of the DCOM configuration.
- No user-defined access rights possible.
- No shared name space with Data Access

3 Basic Information

The following chapter is addressed to technicians and programmers who wish to gain an overview of the basics of OPC Unified Architecture.

The chapter starts with an overview of the OPC foundation and the previously available standards. The following section explains the basics of the setup and contents of an OPC UA Server address space and the interface for the access to the OPC UA Server and the used protocols and security mechanisms. The last part of the chapter contains a description of the basics of the alarm model and the alarm processing in OPC UA.

3.1 OPC basics

Overview

In recent years, the OPC Foundation (an interest grouping 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 different 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 lateral server to server communication.

Based on the experience with this classic OPC interface, the OPC Foundation defined a new platform, called OPC Unified Architecture (UA). Aim of this new standard is the generic description and uniform access to all information which is to be exchanged between systems or applications. This includes the functionality of all previous OPC interfaces. Furthermore, it is to generate the possibility to natively integrate the interface in the respective system, irrespective of on which operating system the system is operated and irrespective of the programming language in which the system was created.

This example deals with the OPC Unified Architecture interface. A detailed documentation is available on the SIMATIC NET CD. Further information is available at www.opcfoundation.org.

What is OPC

In the past, OPC was 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 inter-process communication and organizes the exchange of information between applications even beyond computer boundaries (DCOM). With the help of the mechanisms of the Microsoft operating system, OPC clients (COM client) can exchange information with an OPC server (COM server).

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

Using COM or DCOM has the effect that OPC servers and clients can only be operated on a Windows PC or in the local network and that they mainly have to realize communication to the respective automation system via proprietary

protocols. Additional tunneling tools have to be used for the network communication between client and server in order to get through firewalls or to avoid the complicated DCOM configuration. The interface can furthermore only be accessed natively with C++ applications, .NET or JAVA applications can only access via a wrapper layer. In practice, these restrictions lead to additional communication and software layers which increase the configuration effort and the complexity.

Due to the widespread use OPC, the standard is increasingly used for the general connection of automation systems and no longer only for the original application as driver interface in HMI and SCADA systems to access process information.

To solve the mentioned restrictions in real-life situations and to fulfill the additional requirements, the OPC Foundation has defined a new platform in the last five years, called OPC Unified Architecture, which offers a uniform basis for the exchange of information between components and systems. OPC UA will also be available as IEC 62541 standard and therefore forms the basis for other international standards.

OPC UA offers the following features:

- Summary of all previous OPC features and information such as DA, A&E and HDA in a generic interface.
- Use of open and platform-independent protocols for inter-process or network communication.
- Internet access and communication across firewalls.
- Integrated access control and security mechanisms on protocol and application level.
- Extensive possibility to illustrate object-oriented models, objects can have variables and methods, and can trigger events.
- Extendable type system for objects and complex data types.
- Transport mechanisms and modeling rules form the basis for other standards.
- Scalability of small embedded systems up to business applications and from simple DA address spaces up to complex, object-oriented models.

3.2 Basics of the OPC Unified Architecture

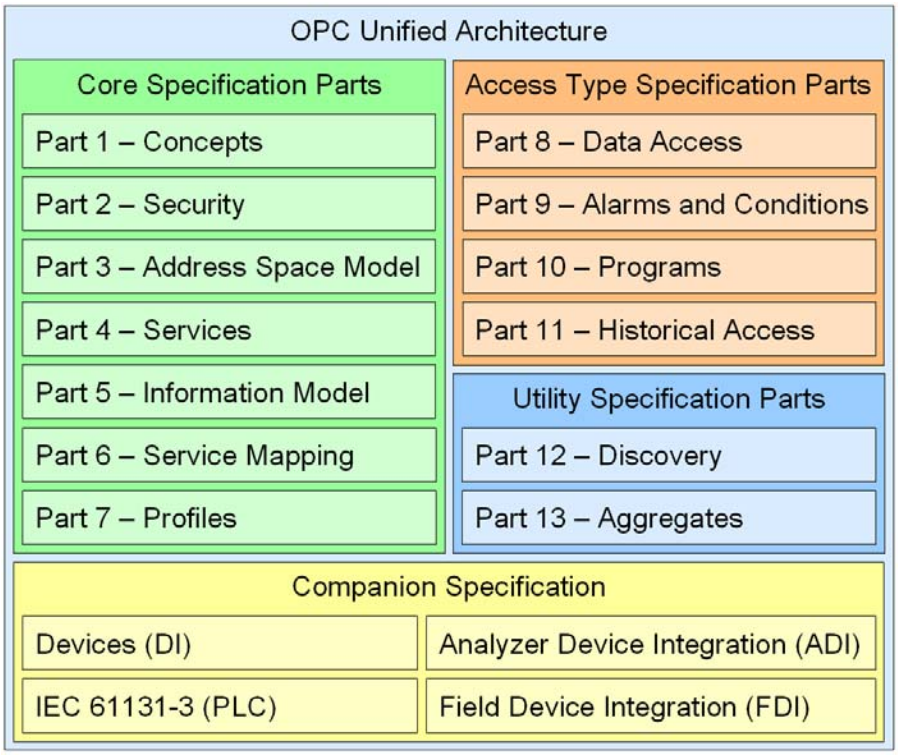
This chapter explains the basis of the OPC Unified Architecture necessary for the example.

3.2.1 OPC UA specifications

Overview

The OPC UA specifications are divided in different parts due to the IEC 62541 standardization. Figure 3-1 gives an overview of the different parts.

Figure 3-1



Part 1 to 7 form the basis of the technology and the realization of OPC UA applications. It is mainly parts 3 to 5 which form the core of the standard.

Parts 8 to 11 define OPC specific information models for the provision of classic OPC information such as current process data or alarms.

Additional tools are defined in part 12 and 13.

Moreover, so called companion specifications are generated which define additional information models, together with other standardization organizations, based on OPC UA. The models and information in other standards form the basis and the companion specification defines how this information is described and transported with OPC UA.

Note

For this application the parts three to five and part eight are relevant. The description of the other parts is included to provide a comprehensive overview of the OPC Unified Architecture.

List of specifications

Table 3-1 explains the list of specification and its content. The currently relevant specifications for the SIMATIC NET server are highlighted here

Table 3-1

Specification	Description
Part 1 – Concepts	This non-normative part gives an overview of the standard.
Part 2 – Security	The requirements to security and an introduction to the basics are described in the second part which is also non-normative.
Part 3 – Address Space Model	This part defines the basic rules and elements for the set up of the address space of an OPC UA server. These rules form the basis for the information models in part 5, 8 to 11 and the companion specifications.
Part 4 – Services	This document is the only part which defines the interface for the access to all OPC UA information. It specifies a list of methods, the so called services. These services are generic and form the basis for all information models.
Part 5 – Information Model	The basis information model defines the access points in the address space and basic types such as, e.g. data types or object types. This part, together with part 3 and 4 forms the core of OPC UA.
Part 6 – Service Mapping	The services in part 4 are independent of the defined transport mechanism used. This part specifies the realization of the services in different ways of serialization, security and transport protocols for messages between OPC UA client and server. This part forms the basis for the implementation of communication stacks and is not relevant for the users of the technology.
Part 7 – Profiles	A profile specifies subset of OPC functionalities for different applications which are offered by an OPC UA server or which can be used by an OPC UA client. This part defines the list of profiles for OPC UA.
Part 8 – Data Access	This part defines the variable types, properties and quality status codes for process data. All other necessary concepts are already contained in the parts 3 to 5.
Part 9 – Alarms and Conditions	This part defines the model for the description of condition monitoring and process alarms and the signaling of status changes via events. All other necessary concepts for events are already contained in the parts 3 to 5.
Part 10 – Programs	This part defines how actions which are running over a longer period of time can be started and monitored. This is performed on the basis of state machines whose handling is defined in part 5 in OPC UA.
Part 11 – Historical Access	Here, the access to historical data and events is defined.
Part 12 – Discovery	Defines how the OPC UA server can be found in the network.
Part 13 – Aggregates	This part defines aggregate functions for data compression such as average or maximum value over a time range. The aggregates can be used for current or historical data.
Devices (DI)	This companion specification defines a generic model for the configuration and diagnostics of devices.
IEC 61131-3 (PLC)	This companion specification defines a mapping of the IEC 61131-3 software model and of the standardized control programming languages on an OPC UA server address space.
Analyzer Device Integration (ADI)	This companion specification defines a model for the configuration and data linking for complex devices for process analysis based on DI
Field Device Integration (FDI)	This companion specification defines a model for the complete engineering of field devices on the basis of Electronic Device Description Language (EDDL) and Field Device Tool (FDT).

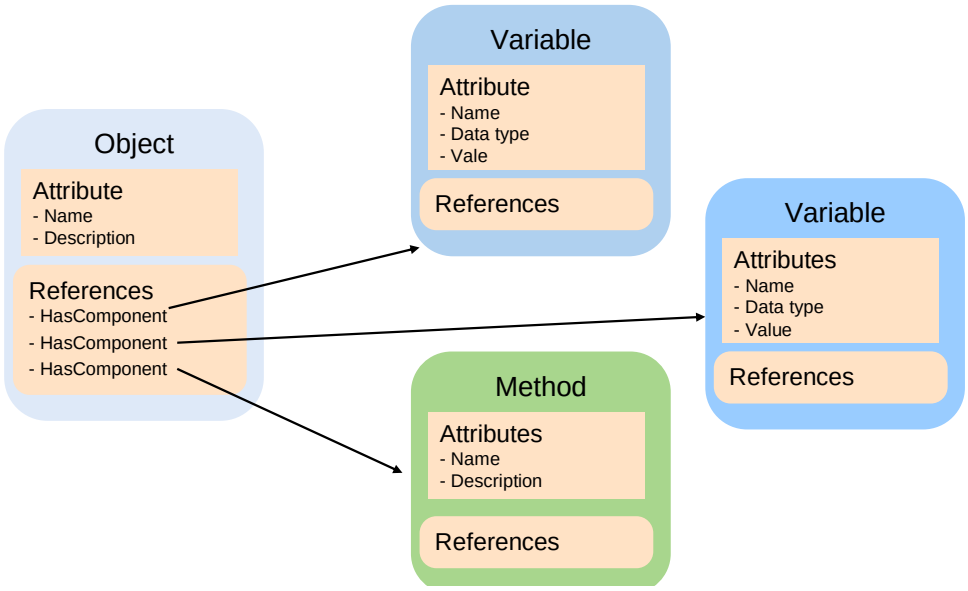
3.2.2 Structure of the OPC UA server address space

Nodes in the address space

A node in the OPC UA address space is of a certain type such as e.g. object, variable or method and is described by a list of attributes. All nodes have joint attributes such as name or description and specific attributes such as, e.g. the value of a variable. The list of attributes cannot be extended. Additional information on the node can be added as property. Properties are a special type of variable.

The nodes are interconnected with references. The references are typified. There are two main groups, hierarchical references such as, e.g. HasComponent for the components of an object or non-hierarchical references such as, e.g. HasTypeDefinition for a connection of an object instance to an object type. Figure 3-2 offers an example for a node and the connection references.

Figure 3-2



Available types of nodes in the address space

The defined node types are listed in Table 3-2. The list of types cannot be extended.

Table 3-2

Node type	Description	Example
Object	An object is used as typified container for variables, methods and events.	The objects which represent a S7 connection always have the same structure.
Variable	Variables represent the data of objects or as property, the properties of a node.	S7 variable in a data block.
Method	Methods are components of objects and can have a list of input or output parameters. The parameters are described via defined properties.	BlockRead( ) method on a S7 connection object with which a block can be read out from the S7.
View	Views represent a part of the address space. The node is used as access point and as filter when browsing.	Views are not available in the SIMATIC NET server.

Node type	Description	Example
Object type	Object types supply information on the structure or the components of an object.	S7ConnectionType describes the components which are present in a S7 connection object.
Variable type	Variable types typically describe which properties or data types can be found in an instance of the type (variable).	The AnalogItemType defines that a variable of this types provides the EngineeringUnits properties and the EURange.
Reference type	Reference types define the possible types of references between nodes.	A method is referenced by an object with HasComponent.
Data type	Data types describe the content of the value in a variable.	The value of a variable can have the Double data type.

Structuring of the address space

The basic structure of the OPC UA address space is defined in part 5. Figure 3-3 shows one part of this structure and SIMATIC NET shows specific parts. The different areas are described in Table 3-3.

Figure 3-3

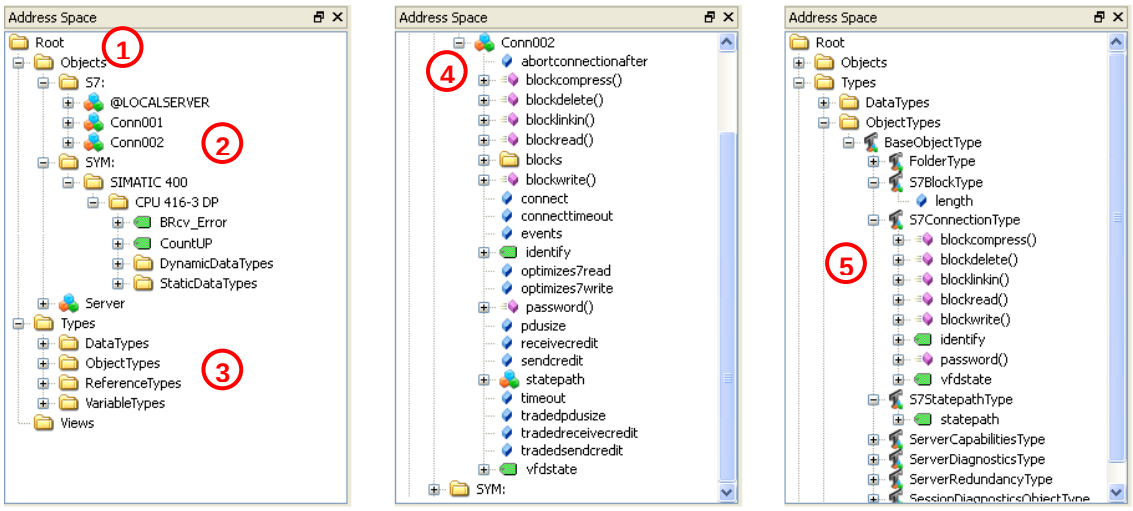


Table 3-3

No	Description
1.	In the Objects directory, instances such as objects and variables can be found. In this directory a data access client can find the variables for data access Apart from the specific SIMATIC NET directories you can also find the server object here which was defined by OPC UA. It contains information on the range of function and the status of the server.
2.	The two directories S7: and SYM: under Objects are specific to the SIMATIC NET OPC UA server. Under S7: the configured S7 connections are listed as objects. Under SYM: the symbols from the STEP 7 project can be found.
3.	In the Types directory are the different type nodes for DataTypes, ObjectTypes, ReferenceTypes and VariableTypes.
4.	An S7 connection object provides various status information and methods. You can, e.g. process or read out blocks in the S7 via methods. Apart from the methods, the properties supply information on the configuration of the S7 connection.
5.	The S7ConnectionType belonging to the S7 connection object, can be found in the ObjectTypes directory. It describes the minimum of methods and variables, present at the instance. The rules for the type system are described in detail in /2/.

Namespaces and NodeId

Each node in the OPC UA address space is uniquely identified by a nodeId. This nodeId is made up of a namespace to distinguish codes from different subsystems and a code which can either be a numerical value, a string or a GUID.

Strings are typically used for the code. This is analog to OPC Data Access, where the itemId as code is also a string. Numerical values are used for statistical namespaces such as, e.g. type system.

OPC UA defines a namespace for the nodes defined by OPC. The OPC UA servers additionally define one or several namespaces. Table 3-4 lists the relevant namespaces for the SIMATIC NET OPC UA server.

Table 3-4

Namespace	Description
http://opcfoundation.org/UA/	Used for nodes which are defined in the OPC UA part 5. These are nodes which form the basic structure of the address space and nodes which represent types defined by OPC UA.
S7:	Namespace for direct addressing of S7 variables with an optimized syntax.
S7COM:	Namenspace for direct addressing of S7 variables with syntax compatible to the OPC Data Access Server.
SYM:	Namenspace for symbolic addressing of S7 variables. The symbol information is exported from the STEP 7 project.
S7AREAS:	Namenspace for the alarm area objects.
S7SOURCES:	Namenspace for event source objects.

Attributes of the nodes

The most important attributes of nodes are listed as an example in the table below. The main emphasis is on the variable node type.

Table 3-5

Attributes	Node type	Description
NodeId	All	Unique node address.
DisplayName	All	Localized display name for the node. The language depends on the language requested by the client for the connection and on the languages supported by the server.
BrowseName	All	Non-localized name for the node. The name contains a namespace and is mainly relevant for the use of types.
NodeClass	All	Type of node such as, e.g. object, variable or method.
Description	All (optional)	Optional localized description of the node.
Value	Variable	Value of the variable. Just like for all other attributes, time stamp and status of the value are delivered together with the value of the attribute when reading them.
DataType	Variable	Data type of the variable or of the value attribute. Data types are, e.g. OPC UA defined data types such as Int32, Double or String or also structured data types.
ValueRank	Variable	Indicates whether the value (value attribute) is a scalable value, an array or a multi-dimensional array.
AccessLevel	Variable	Indicates whether the variable can be read or be written.

3.2.3 Interface for the access to the OPC UA server address space

Communication channel and application objects

Figure 3-4 shows the different objects which can be created during data exchange between OPC client and server. The objects are described in Table 3-6.

Figure 3-4

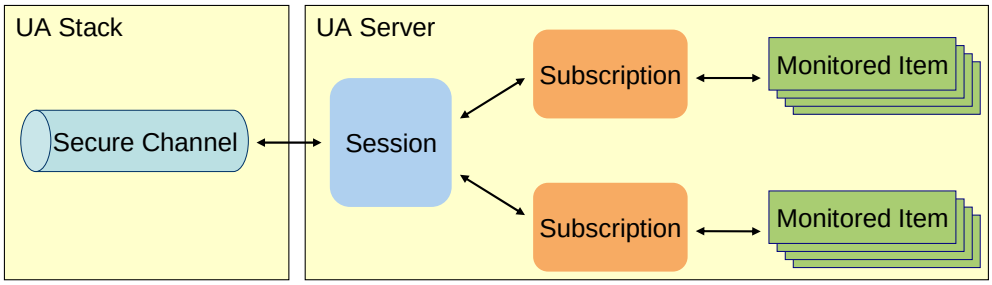


Table 3-6

Object	Description
Secure Channel	The secure communication channel is realized in the OPC UA stack. The objects on application level are independently viable. However, they can only be created, used or changed within the context of a secure channel. If a new secure channel is established after an interrupted connection, it has to be assigned to the session on application level.
Session	The session in the server is the logic connection between OPC UA client and server. It contains user information and language settings for the connection. The session is deleted from the server if no calls are received by the client within the timeout. The timeout is specified by the client. The session is linked to a secure channel but can be assigned a new secure channel if the communication was interrupted.
Subscription	A subscription object can be created by the client to group monitored items. Monitored items are used to monitor value changes or to receive event messages. The subscription is deleted by the server if no data or KeepAlive messages could be sent to the client within the timeout. The timeout is specified by the client.

Methods for establishing a connection

Table 3-7 explains the most important methods of the OPC UA interface for establishing a connection.

Table 3-7

Method	Description
OpenSecureChannel	Opens a secure communication channel between client and server. To open the connection, the server URL, the application certificates and the security settings are necessary.
CreateSession	Creating an application session within the context of a secure channel.
ActivateSession	Activating the session by transferring the user authentication and language settings. This method is also used to assign an existing session to a new secure channel or to change the user.
CloseSession	Closes the application session.

Methods of the session object

Table 3-8 explains the most important methods of the OPC UA interface regarding the session.

Table 3-8

Method	Description
Browse	Supplies the list of nodes which can be obtained from a start node via a reference. The quantity of nodes can be restricted by filters. For each node, information is delivered which is, e.g. necessary for the display in a tree view.
Read	Reads a list of node attributes. With this method, values of variables (value attribute) and also meta data such as, e.g. the data type of a variable (DataType attribute) can be read.
Write	Writes a list of node attributes. This is a typical method for writing values of variables. If the server permits it, other attributes can also be written.
Call	Calls a list of methods. For a method, the nodeId of the object and the nodeId of the method to be called is transferred. If parameters are defined for the method, the input parameters are transferred at the call and the output parameters are supplied in the result. Methods are also used for actions at alarm objects such as the acknowledgement of alarms.
CreateSubscription	Creating a subscription for the receipt of data changes or event messages. The subscription is used for the grouping of information which is to be monitored. All new data or events are delivered as a package in adjustable time intervals for a subscription.
DeleteSubscription	Deleting a subscription.

Methods of the subscription object

Table 3-9 explains the most important methods of the OPC UA interface regarding the subscription.

Table 3-9

Method	Description
ModifySubscription	Changes the settings of a subscription, such as e.g. the publish interval in which new data for the client is collected and jointly sent.
CreateMonitoredItems	Creating a list of monitored items in a subscription. A monitored item is either used to monitor a value of a variable or to monitor event messages. Both types of monitored items can be combined to this method in one call. Only event messages are monitored in this application.
ModifyMonitoredItems	Changes the settings of a list of MonitoredItems, such as e.g. the event filter for the monitoring of event messages.
DeleteMonitoredItems	Deletes a list of monitored items in a subscription.
Publish	Method for transferring data packages for a subscription with value changes and event messages in the publish interval. This method is not visible in the Client API. The functionality there is realized as callback to the client application.

3.2.4 Protocols and security mechanisms

OPC UA communication architecture

The services for the access to the information in an OPC UA server address space such as browse, read and write are abstract and specified independent from the transport protocol in part 4.

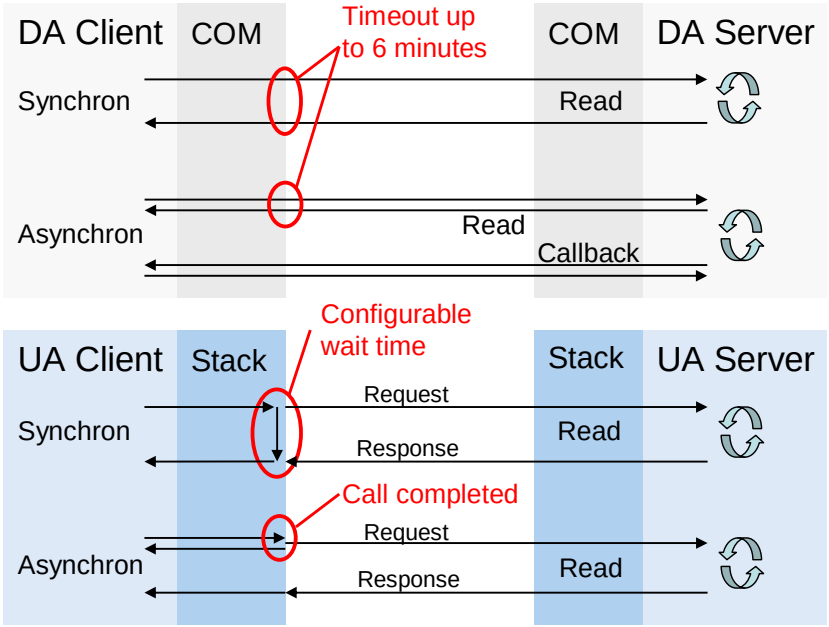
The different bindings for the transmission of service messages between OPC UA client and server are defined in part 6. A binding is made up of protocol, security mechanisms and serialization type for the data.

The bindings are implemented in communication stacks. At the moment there are three implementations from the OPC Foundation, namely in ANSI C, C# / .NET and JAVA. In this application the C# / .NET stack is used.

The methods on the API of the stacks for the application correspond to the services in part 4 with concrete data types from the respective programming language. This is how in application development a native API can be accessed in the respective programming language. The application can also be implemented independent from the binding used. New bindings can be expanded by exchanging the OPC UA stacks.

Synchronous and asynchronous calls

Figure 3-5



For COM all calls to the server are synchronous. This is why additional asynchronous functions were defined for few actions such as read and write. A synchronous call starts the action in the server. After completing the action, the server sends a synchronous callback to the client. Due to the synchronous call to start the action, asynchronous calls may also block when the network connection is interrupted.

In the case of OPC UA all calls to the server are asynchronous. There is no differentiation between synchronous and asynchronous methods in the specification. Once the request message was written on the network, the asynchronous call is returned to the client application. This is why an asynchronous

call cannot be blocked. Since an asynchronous call can always be made synchronous, the stacks offer all OPC UA methods also as synchronous calls. For this purpose, the call is held in the stack until the response message has arrived from the server or until the timeout has expired. The timeouts can be adjusted individually per call. There is no difference between synchronous and asynchronous calls for the server.

Security layers

The different security layers of OPC UA are described in Figure 3-6 and Table 3-10.

Figure 3-6

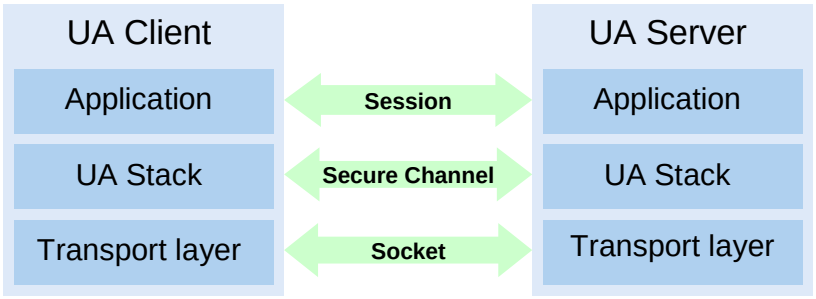


Table 3-10

Layer	Description
Socket	On the socket level, a connection-oriented security of the socket connection via Secure Socket Layer (SSL) or via Virtual Private Network (VPN) can be used in addition or as an alternative to the secure channel.
SecureChannel	On the SecureChannel level, mutual authentication of the applications and a message-based security of the communication are performed. Each message is signed and encrypted to ensure the integrity and secrecy of the messages. Basis of these mechanisms are certificates which uniquely identify the applications based on a Public Key Infrastructure (PKI) system. A detailed description of these mechanisms can be found in /2/. Exchanging these certificates as an important step in the security configuration is described in the next section.
Session	On the session level a user authentication is performed.

Configuration options for security

Table 3-11 describes the different configuration options for the security mechanisms.

Table 3-11

Option	Description
Security Policy	**None** – In the secure channel no security is used. **Basic128Rsa15** – Set of algorithms for the security. **Basic256** – Set of algorithms for the security with longer keys.
Message Security Mode	**None** – The messages are not secured. **Sign** – The messages are signed. **Sign&Encrypt** – The messages are signed and encrypted.
User Authentication	**Anonymous** – No user authentication is necessary. **User Password** – The user authentication is performed using user names and password. **Certificate** – The user authentication is performed using a certificate.

Exchange of certificates

The exchange of certificates between client and server and the accepting of the certificates is explained in Figure 3-7 and Table 3-12.

When all applications involved implement the guidelines of the OPC UA regarding the security configuration, then only one manual step at the server is necessary for the exchange of certificates, since the certificates are automatically exchanged between the applications and the certificates only have to be accepted by an administrator.

The manual exchange of certificates is explained in chapter 6.3.2 since not all applications implement the automatic steps yet.

Figure 3-7

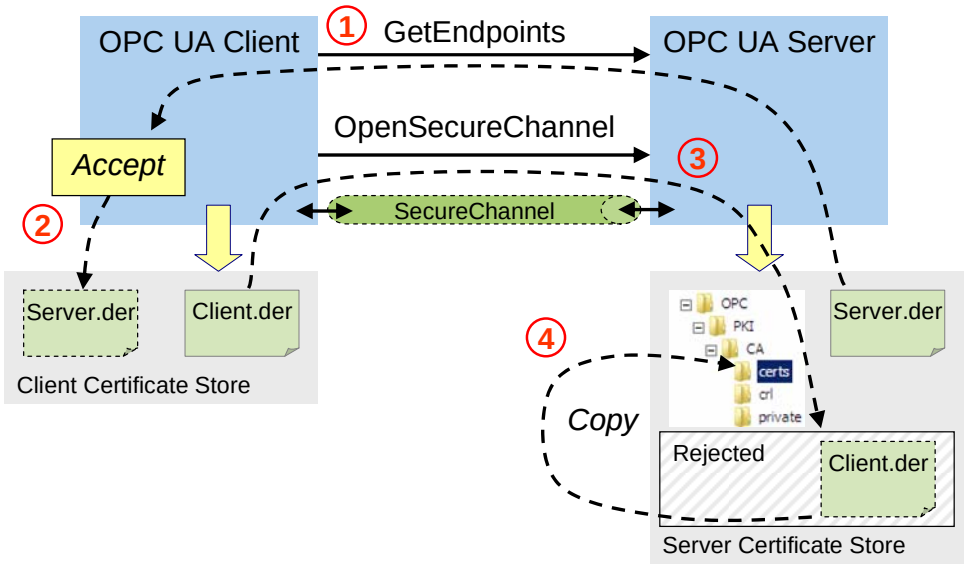


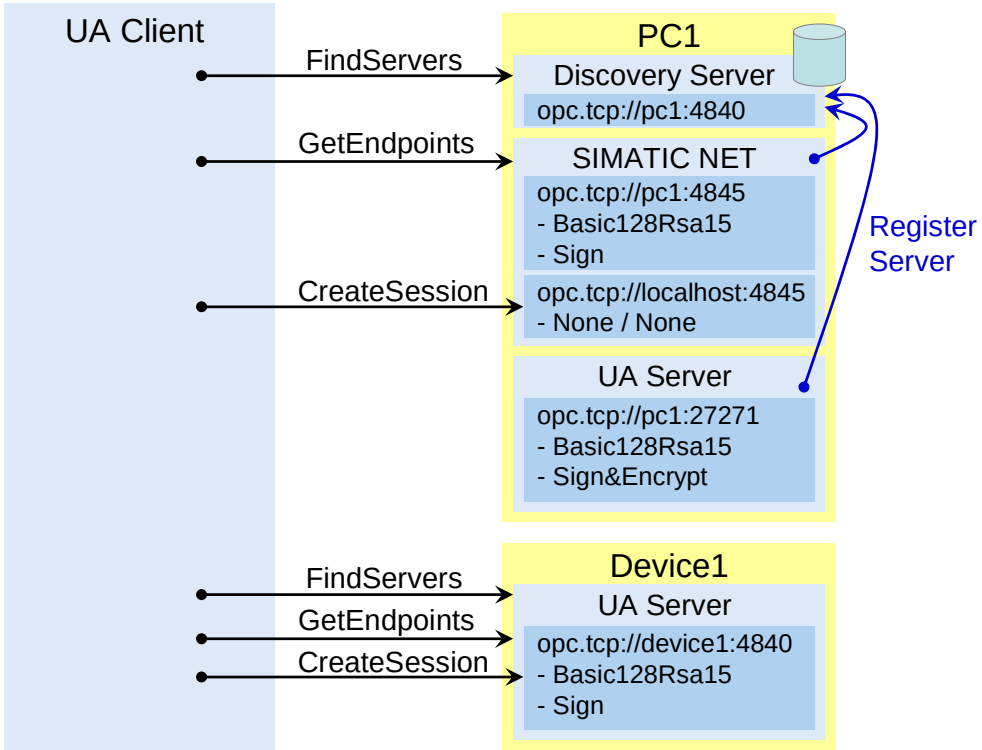
Table 3-12

Step	Description
1.	Before the client can connect itself with the server, it needs the necessary information such as the security mechanisms, protocols and the address for connection demanded by the server. This information indicates a so called endpoint. The available endpoints of a server are delivered with the GetEndpoints call. With the description of the endpoints the server also delivers its certificate.
2.	Once the endpoints have been selected with the security settings, the user is asked whether he/she wants to accept the certificate. If yes, this is stored in the certificate storage of the client.
3.	When calling the OpenSecureChannel the client certificate is transferred to the server. If the certificate is not known in the server, it will be stored in a Rejected directory.
4.	With a configuration tool of the server, certificates from the Rejected directory can be accepted. They are moved to the certificate storage of the server.

Server discovery

So far, a Local Discovery Service (LDS) has been defined for OPC UA Discovery which from its basics functionality is comparable with the OPC Enum with the classic OPC.

Figure 3-8



A LDS supplies a list of the local network nodes available on OPC UA servers. A LDS supplies a list of the local network nodes available on OPC UA servers. The servers on a PC are registered with the LDS.

A client can select a server and establish a connection with the following steps:

- Establishing a connection without security with port 4840 and call of **FindServers**. This call supplies a list of available servers and their discovery URL.
- Establishing a connection without security to discovery URL of the desired server and call of the **GetEndpoints**. This call supplies the list of endpoints with the endpoint URLs and the security settings of the endpoints.
- Establishing a connection with the endpoint URL and the demanded security settings. Subsequently an application session can be opened with **CreateSession**.

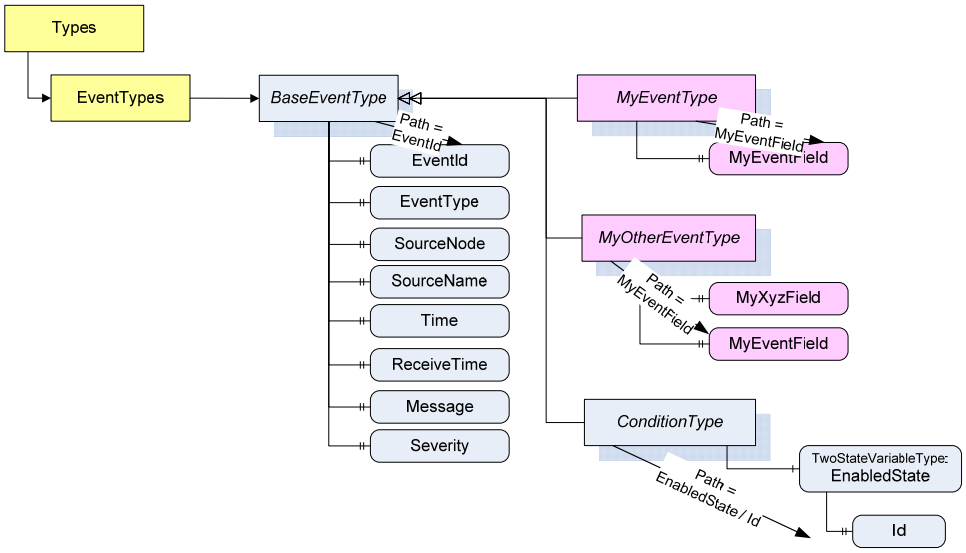
If only one OPC UA server is available on a system, it can run on the standard port 4840, since all servers also have to implement FindServers and as a result only supply themselves. In this case, the endpoints also use port 4840.

3.3 Basics for OPC UA event messages and alarms

Event types

Like any other type information, the types of events the server can provide are also visible in the address space of the OPC UA server. Figure 3-9 shows an example for the type hierarchy of the event types. The root of the derivation hierarchy is the BaseEventType. The types for Alarms & Conditions are available below the ConditionType. Application-specific event types, such as MyEventType, can be derived from the standard types at any location.

Figure 3-9



An event type mainly defines the event fields delivered with the event and to which the filter can be applied, such as message text (Message), time stamp (Time) or event source (SourceNode) which are defined at the BaseEventType. This information is specified as filter when creating an event monitored item.

When using the event fields in the filter the BrowseNames path from the event type on becomes relevant For most event fields this path is only as long as one BrowseName. Event fields of the same name at different event types, such as MyEventField, can be selected together. In this case the event type BaseEventType is simply given.

Event signaling objects and event hierarchy

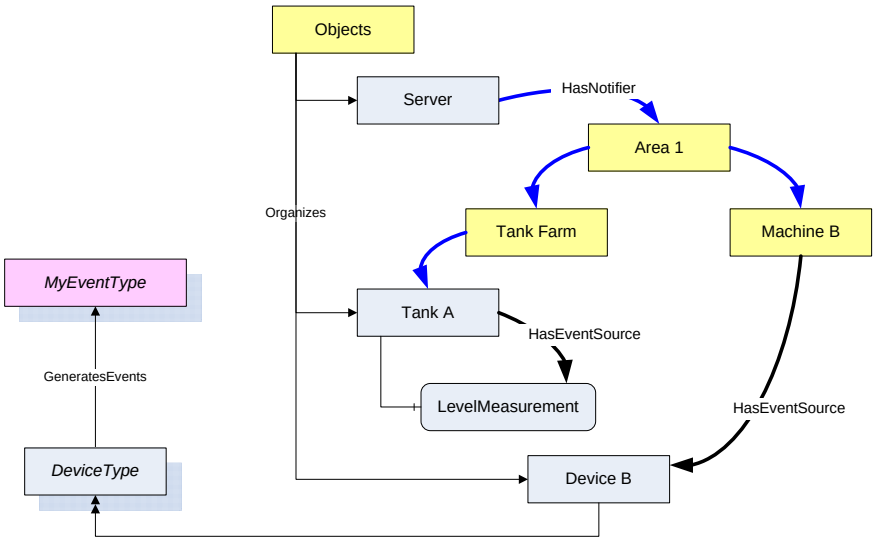
Events are by themselves not visible as nodes in the address space. They can only be received via objects. Not all objects can signal events. Whether an object can signal events is specified at the object by the EventNotifier attribute. Only objects where this attribute has been set can be specified in the Event Monitored Item and received in Clients Events.

All events can be received by the server via the server object defined by OPC UA. It is therefore used by the client in a Monitored Item if the number of events shall be limited only on the basis of the given filter. Specifying a concrete object is a way of prefiltering the number of events.

For this prefilter the server can also form a hierarchy with the so-called areas to permit the clients selecting events from areas. Figure 3-10 illustrates an example for such an event hierarchy. The root node of this hierarchy is the server object. The tree is formed with HasNotifier and HasEventSource references. HasNotifier is used if the target of the reference is an event signaling object. The

HasEventSource reference is used to point to the source of events. The source for events can be variables or objects.

Figure 3-10



The GeneratesEvents reference is used for referencing those event types for an object type which can be triggered by this object.

Events received by the server

If an OPC UA Client wants to receive events from a server, it must create one or several Event Monitored Items in one subscription. For an Event Monitored Item an event signaling object, the EventNotifier attribute and an event filter must be specified. The filter is composed of a selection of the event fields and a filter for the event fields. Table 3-13 shows an example for the settings for an Event Monitored Item.

Table 3-13

Parameter	Examples for Event Monitored Item			
NodeId	Tank A			
Attribute	EventNotifier			
Filter Select Clause	Index	EventType	Path	Attribute
	0	BaseEventType	SourceNode	Value
	1	BaseEventType	Message	Value
	2	BaseEventType	Severity	Value
	3	MyEventType	MyEventField	Value
Where Clause	Index	Operator	Operand 1	Operand 2
	0	And	Index 1	Index 2
	1	GreaterThan	Severity	500
	2	OfType	MyEventType	

The Where Clause in the event filter is used to restrict the number of events which the server supplies for the selected object. In the example only those events of the MyEventType are supplied for which the severity is higher than 500.

This filter is formed of a list for which an entry is composed of an operator and a number of operands depending on the operator. Individual lines can then be connected via AND / OR logic operation.

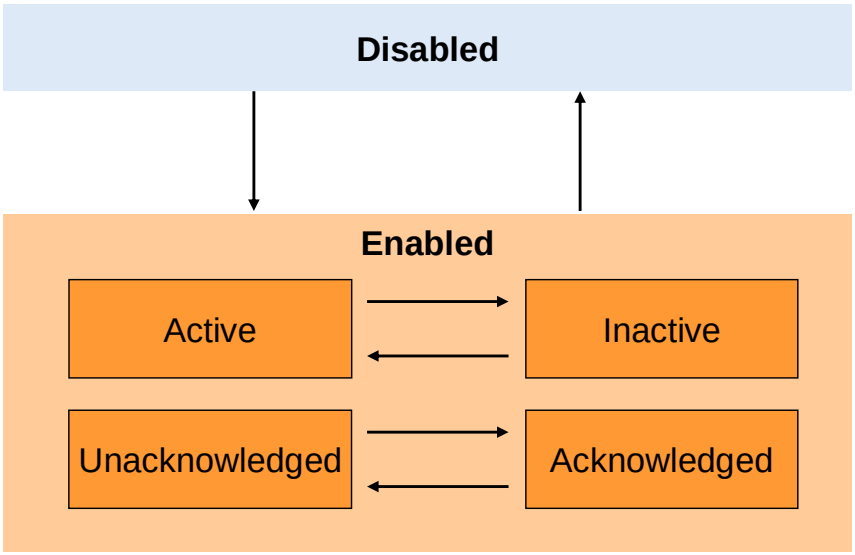
The Select Clause specifies the list of event fields which are reported with an event. This list is independent of the event fields which are used for the filter.

Condition objects and condition events

Condition objects are used to depict the status monitoring. Such a monitoring object is normally not monitored by an OPC UA client individually via Data Access but only interesting states are supplied to an OPC UA client via events. This is normally referred to as alarms. For filling level monitoring, for example, critical filling levels below or above the limits can be reported to the client as an alarm.

An alarm is composed of various nested or parallel state machines. Figure 3-11 shows the most important state machines of an alarm. Monitoring can generally be enabled or disabled. If monitoring is enabled, the alarm can be active or inactive and acknowledged or unacknowledged.

Figure 3-11



The basic type for all condition objects is the condition type. It is derived from BaseEventType. However, as opposed to simple events, condition objects can be visible in the address space of the OPC UA server since even though they do send events for a state change, the current states of the various state machines can always be read. The visibility in the address space is optional though. All mechanisms for alarm processing work even without the condition objects being contained in the address space.

If a condition object changes one or several states, the server sends an event with the requested event fields to the client. Actions from client to the condition objects, such as acknowledging an alarm, occur via the call of defined methods at the condition objects. After establishing a connection between client and OPC UA server via the Refresh method, the currently active alarms can be sent to the client as events.

Event fields of simple events

The BaseEventType defines a set of event fields which can be delivered by the server for all events. Table 3-14 describes the most important event fields. The other event types are also based on these event fields, but they supply additional event fields for the respective type.

Table 3-14

Event fields	Description
SourceName SourceNode	Name or NodeID of the event message source.
Time	Instant at which the event has occurred.
ReceiveTime	Time at which the event was processed in the server.
EventType	NodeId of the event type of the event message.
Message	Descriptive text for the event message.
Severity	Severity of the event message. The range of values of the severity is from 1 to 1000, where 1000 corresponds to the highest severity.
EventId	Unique indentifier for the event

Event fields of condition events

The OPC UA Alarms & Conditions specification defines a set of event fields that is supplied by the server for condition events and alarms. This set consists of the event fields defined by the BaseEventType, and additional event fields for condition events. Table 3-15 describes the most important additional event fields.

Table 3-15

Event fields	Description
ConditionName ConditionId	Name or NodeID of the condition at which a status change has occurred.
ConditionClassName ConditionClassId	Name or NodeId of the condition classification. A separate type hierarchy for classification is intended. Predefined classes are Process, System and Maintenance.
EnabledState	Indicates whether the state monitoring is enabled or disabled.
AckedState	Specifies whether the alarm must be acknowledged.
ActiveState	Indicates whether the alarm is active.
Retain	A flag which indicates whether the condition is in a state interesting for the client and to be displayed.
Quality	Indicates the quality of the value on which the condition is based. This can, for instance, be the level of a tank.
Comment	Comment set for the condition by a client. This can also be performed via acknowledging an alarm
ClientUserId	Text identifier which identifies the application which has acknowledged the alarm or has added a comment.

Methods of condition objects

Condition events can become visible as objects in the address space of the server. However, an OPC UA client can call up methods at the object irrespective of the visibility. Table 3-16 describes the most important methods at a condition object.

Table 3-16

Method	Description
Refresh	When the method is called up an event with the current state is triggered for the calling client for all conditions. Only those conditions are updated for which the Retain flag has been set.
Enable	Enables the monitoring of a condition.
Disable	Disables the monitoring of a condition.
AddComment	Sets a comment at the condition.
Acknowledge	Acknowledges an alarm.

Operators for the event filter

Table 3-17 describes the most important operators that can be used in an event filter in the Where Clause.

Table 3-17

Operator	Description
Equals	True if operand one equals operand two.
GreaterThan	True if operand one larger than operand two.
LessThan	True if operand one smaller than operand two.
GreaterThanOrEqual	True if operand one larger or equal operand two.
LessThanOrEqual	True if operand one smaller or equal operand two.
Like	True if operand one corresponds to a text pattern defined in operand two.
Not	True if operand one is not true.
Between	True if operand one larger or equal operand two and smaller or equal operand three.
InList	True if operand one equal to one of the other operands.
And	True if operand one and operand two are true.
Or	True if operand one or operand two are true.
OfType	True if the event type is equal to operand one or a subtype of operand one.

Operators for the event filter

Table 3-18 describes the operands that can be used in an event filter in the Where Clause.

Table 3-18

Operator	Description
Literal	Specifies a value.
Attribute	Specifies an event field of an event. The event field is specified via the BrowseName.
Element	Via an index it refers to another entry in the Where Clauses list.

4 Functional Mechanisms of the Client Application

The following chapter is addressed to programmers who wish to gain an overview of the functional mechanisms of the OPC UA Client application.

The chapter starts with an overview of the overall architecture. The following section describes the Client API in detail. In the last part of the chapter the functional processes are described in class and sequence diagrams.

General overview

Figure 4-1

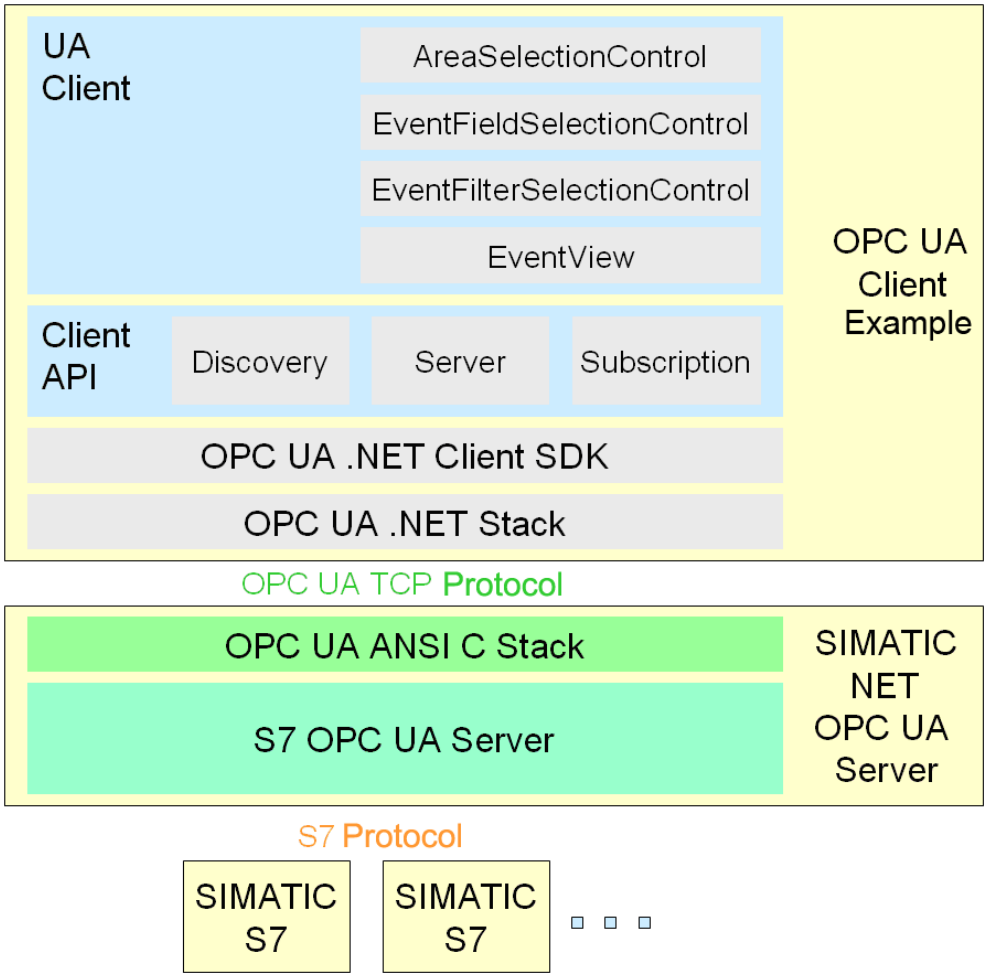


Table 4-1

Module	Description
OPC UA .NET Stack	The .NET based OPC UA communication stack of the OPC foundation.
.NET Client SDK	The .NET based OPC UA client SDK of the OPC foundation.
Client API	Reusable, simplified and tailored to the .NET Client API task. It offers reusable C# classes for discovery, session and subscription handling.
UA Client	Comfortable OPC Client with the functions Discovery, Connect, Disconnect, Monitoring of events, Filter settings for events and Selection of delivered event fields. General functions such as browsing of areas, display of available filters and display of event fields are encapsulated in reusable controls.
ANSI C UA Stack	The SIMATIC NET OPC UA server uses the optimized and portable OPC UA ANSI C stack of the OPC foundation.
S7 OPC UA server	The SIMATIC NET OPC UA server implements the necessary server logic for sessions and subscriptions and the data connection to the S7 stations.

Program overview

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

Figure 4-2

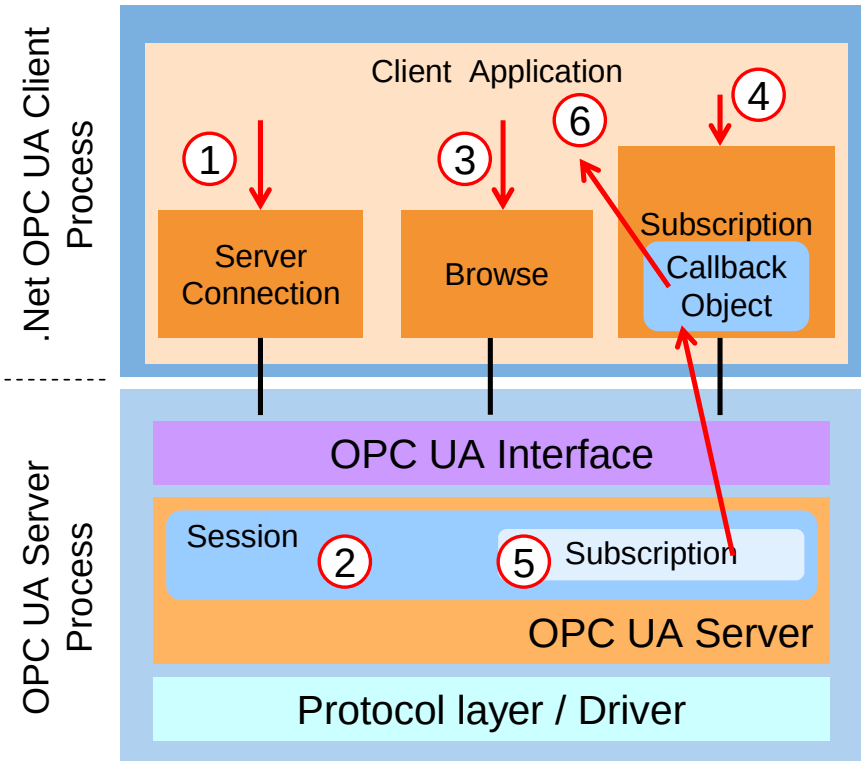


Table 4-2

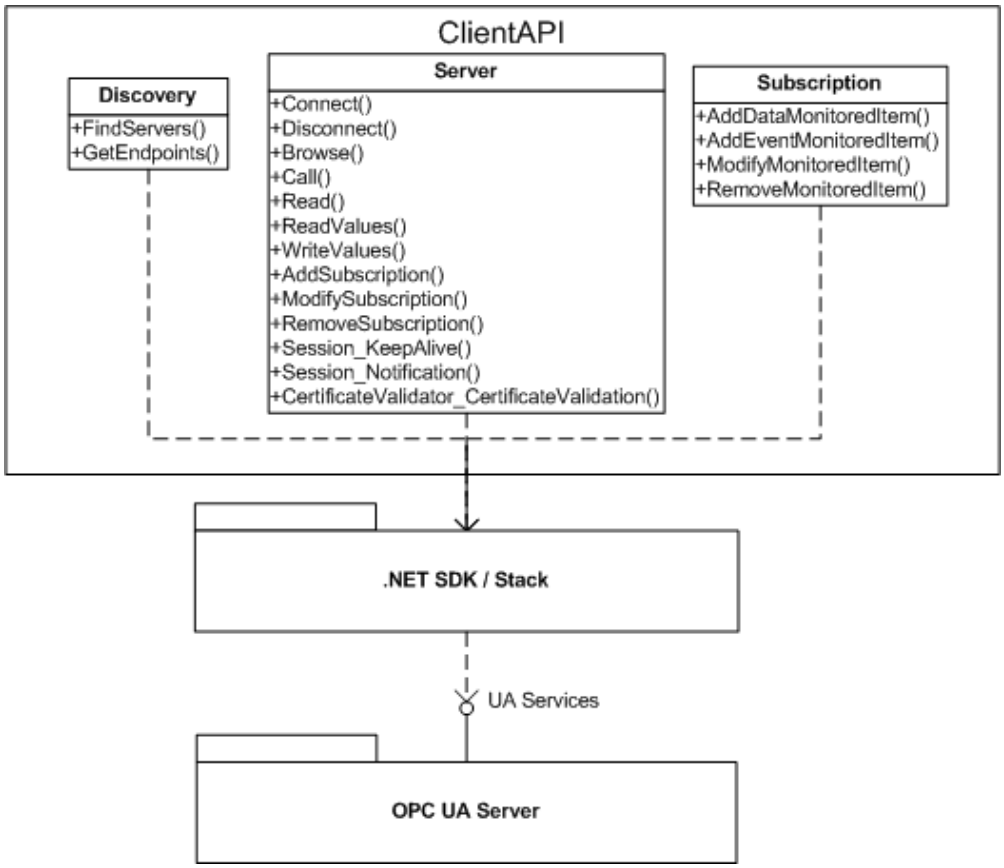
No	Description
1	When establishing the connection between user interface and the OPC UA server, a client API object is generated on the client side. This object manages the connection with the server (2). It furthermore provides all OPC UA services with the exception of services that are related to a subscription.
2	The session object is generated in the server via the OPC UA interface.
3	When starting the configuration dialog, the OPC UA server browses for areas, event types, and event fields, which are displayed in the configuration dialog in a respective tree.
4	During the connection process a subscription object is generated which provides all the OPC UA services related to a subscription. A MonitoredItem is added to the subscription. This is the server object by default – in the configuration dialog a different object or an area can also be selected.
5	A subscription object which manages all subscription relevant settings is generated in the server via the OPC UA interface.
6	To be able to receive events from the server, a callback connection is established. A SubscriptionCallback object is created in the client and connected to the subscription in the server. If changes are sent from the server to the client, it enters the changes into the monitoring window.

4.1 OPC UA Client API

The class diagram in Figure 4-3 shows the OPC UA client API classes. These classes encapsulate the accesses to the OPC UA server in a simple and reusable .NET API.

The classes are summarized in the .NET Assembly Siemens.OpcUA.dll. It has dependencies to the .NET Client SDK Assembly Opc.Ua.Client.dll and to the .NET Stack Assembly Opc.Ua.Core.dll.

Figure 4-3



Discovery class

The **Discovery** wrapper class described in the table below encapsulates the necessary methods for the discovery server.

The class is implemented in the ClientDiscovery.cs file in the ClientApi project.

Table 4-3

Method	Functionality
FindServers	Detects the OPC UA servers on a computer.
GetEndpoints	Detects the available endpoints for one or several servers.

Server class

The **Server** wrapper class described in the table below encapsulates the functionality for the access to the OPC UA server. Moreover, it simplifies the use of those OPC UA services which are need by the client application, with the exception of services for the subscription.

The class is implemented in the ClientAPI.cs file in the ClientApi project.

Table 4-4

Method	Functionality
Connect	Creates a secure channel as communication channel and a session in the OPC UA server.
Disconnect	Deletes the session in the server and disconnects the secure channel connection.
Browse	Supplies the list of nodes which are obtainable from a transferred start node via a reference. The list of results can be influenced via filter settings.
Call	Calls a UA method.
Read	Supplies the values to a list of attributes of a node.
ReadValues	Supplies the values of the attribute value of a list of nodes.
WriteValues	Writes the value of the attribute value of one or several variables.
AddSubscription	Creates a subscription and links it to the session.
ModifySubscription	Changes the settings of a subscription.
RemoveSubscription	Removes an existing subscription.
Session_KeepAlive	Keep-alive callback.
Session_Notification	Called when the OPC UA server sends a reply (publish message).
CertificateValidator_ CertificateValidation	Called when the certificate of the servers is considered untrusted.

Subscription class

The **Subscription** wrapper class described in the table below encapsulates the use of a subscription for receiving value exchange and events from the server.

The class is implemented in the ClientSubscription.cs file in the ClientApi project.

Table 4-5

Method	Functionality
AddDataMonitoredItem	Creates a monitored item to monitor value changes and links it with the subscription.
AddEventMonitoredItem	Creates a monitored item to receive events and links these with the subscription.
ModifyMonitoredItem	Changes the settings of the monitored item.
RemoveMonitoredItem	Removes a monitored item from the subscription.

4.2 OPC UA Alarms&Conditions Client

4.2.1 User interface

The figure and table below describe the user interface of the generic OPC UA client example with which the information of the namespace of an OPC UA server can be conveniently accessed.

Figure 4-4

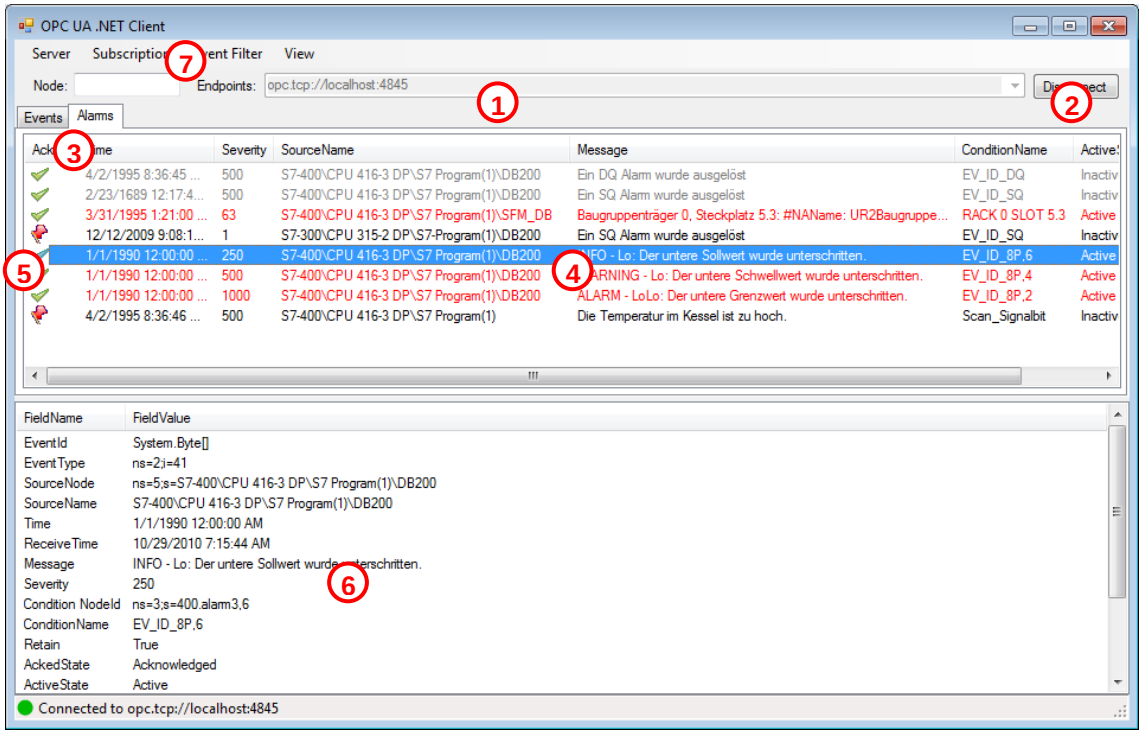


Table 4-6

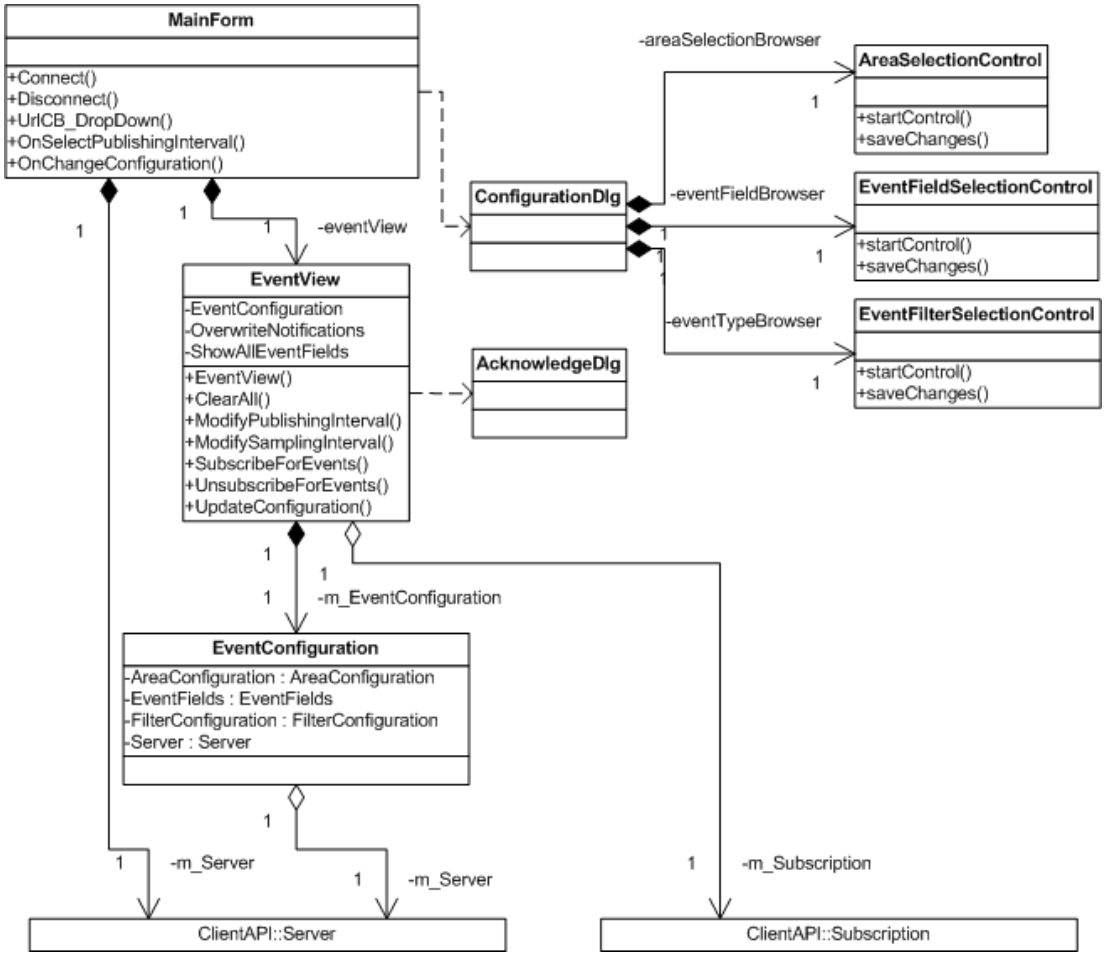
No	Description
1.	The server can be selected via the endpoints selection list. For this purpose, the list of servers and the endpoints are detected by the discovery server. The computer, from which the discovery server is to be prompted, can be entered in the node text field. If the field is empty, the local discovery server is addressed.
2.	The connection to the server can be established or cancelled via the Connect/Disconnect button.
3.	The EventView Control contains two views for selection which can be toggled via a TabControl. o Events tab: displays all events o Events tab: only displays events of the ConditionType (or subtypes)
4.	Displays the received events with preconfigured event fields. The standard event fields are: o In the Events tab: Time, ReceiveTime, Severity, SourceName, Message, EventType and SourceNode o In the Alarms tab: AcknowledgeState, Time, Severity, SourceName, Message, ConditionName, ActiveState and Retain Flag
5.	In the first column of the Alarm tab a symbol indicates whether an event has already been acknowledged. (red flag: unacknowledged, green checkmark: acknowledged)
6.	For the currently selected event in the events list (4) all events fields are displayed which were

No	Description
	supplied for this event.
7.	The publishing interval for the subscription can be changed via the “Subscription” application menu. Via the “Event Filter” application menu the configuration of the event filter, area and event field can be changed. The “Show All EventFields” setting in the submenu specifies whether requested event fields for which no value has been delivered were nevertheless displayed in the list (6).

4.2.2 Class diagram

The following class diagram shows the classes of the OPC UA sample client. These classes realize the functionality of the user interface and use the classes of the UA client API. On both of the following sides, the individual classes are explained in detail.

Figure 4-5



MainForm class

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

The class name corresponds to the file name in the UAClient project.Table 4-7

Method	Functionality
MainForm	Class constructor.
Connect	Implements the functionality to establish a connection with the server and initializes the Browse Control.
Disconnect	Disconnects the connection to the server.
UrlCB_DropDown	Fills all available UA Endpints into the ComboBox. If the Node field is empty, the local DiscoveryServer is polled, otherwise the DiscoveryServer on the computer specified in the Node field.
OnSelectPublishingInterval	Sets the publishing interval for the subscription at the EventView.
OnChangeConfiguration	Shows the configuration dialog (ConfigurationDlg) and after changing the configuration it calls UpdateConfiguration at the EventView.

EventView class

The **EventView** class described in the table below implements the functionality for receiving and displaying the events of the server.

The class name corresponds to the file name in the UAClient project.

Table 4-8

Method	Functionality
EventView	Class constructor.
ClearAll	Deletes all entries in the lists in Alarms tab and Events tab.
ModifyPublishingInterval	Changes the publishing interval of the subscription.
ModifySamplingInterval	Changes the sampling interval for the MonitoredItem.
SubscribeForEvents	Creates a subscription and adds a MonitoredItem to the subscription.
UnsubscribeForEvents	Deletes the subscription.
UpdateConfiguration	Recreates the subscription with the changed parameters.

EventConfiguration class

The **EventConfiguration** class described in the following table contains all parameters for receiving and displaying the events in EventView.

The class name corresponds to the file name in the UAClient project.

Table 4-9

Property	Functionality
AreaConfiguration	Holds the NodeId of the UA object of which events are to be received.
EventFields	Holds the list of event fields to be sent from the server to the events as associated values.
FilterConfiguration	Holds the configuration of the EventFilters. This includes: o Min Severity o Max Severity o EventTypes (list of types of events to be sent)

AcknowledgeDlg class

This class shows a dialog for acknowledging events. During the acknowledgement process a text can be added by the user which is transferred to the server.

The class name corresponds to the file name in the UAClient project.

ConfigurationDlg class

The **ConfigurationDlg** class implemented in the following table implements the display of the configuration for the events displayed in EventView. This includes filter settings, selection of event fields, and the UA object from which the events shall be received. The following controls are stored in the ConfigurationDlg.

Table 4-10

Control	Functionality
AreaSelectionControl	Here the object or area from which events shall be received can be selected.
EventFieldSelectionControl	All event fields can be selected here which the server sends to the events as associated values.
EventFilterSelectionControl	Only events from the event types selected here (or their sub-types) are sent by the server.

Using the client API in the example

The table below lists the files and functions in which the client API is used.

Table 4-10

Client API	Used in
Discovery class	
FindServers	MainForm.cs in the UrlCB_DropDown method
GetEndpoints	MainForm.cs in the UrlCB_DropDown method
Server class	
Connect	MainForm.cs in the Connect method
Disconnect	MainForm.cs in the Disconnect method
Call	EventView.cs in the menuItem_Acknowledge_Click method
Browse	EventFieldSelectionControl.cs in the Browse method
Read	EventView.cs in the getDisplayNameForNodeId method
ReadValues	Not used in this example.
WriteValues	Not used in this example.
AddSubscription	EventView.cs in the SubscribeForEvents method
ModifySubscription	EventView.cs in the ModifyPublishingInterval method
RemoveSubscription	EventView.cs in the UnsubscribeForEvents method

Subscription class	
AddDataMonitoredItem	Not used in this example.
AddEventMonitoredItem	EventView.cs in the SubscribeForEvents method
ModifyMonitoredItem	EventView.cs in the ModifySamplingInterval method
RemoveMonitoredItem	EventView.cs in the UnsubscribeForEvents method

4.2.3 Sequence diagrams

Establishing and terminating the connection to the OPC UA server – User interface

The following sequence diagram shows the procedures which are necessary to establish the connection to the OPC UA server. By clicking **Combobox Endpoints** the user selects an available endpoint first.

Establishing a connection can be started via the **Connect Button** in the user interface or via the **Server Menu**. Once the connection was successfully established the label **Disconnect** appears on the **Connect Button.** The sequence diagram also shows the processes which are triggered through the “Disconnect Server” action via the **Disconnect Button**.

Figure 4-6

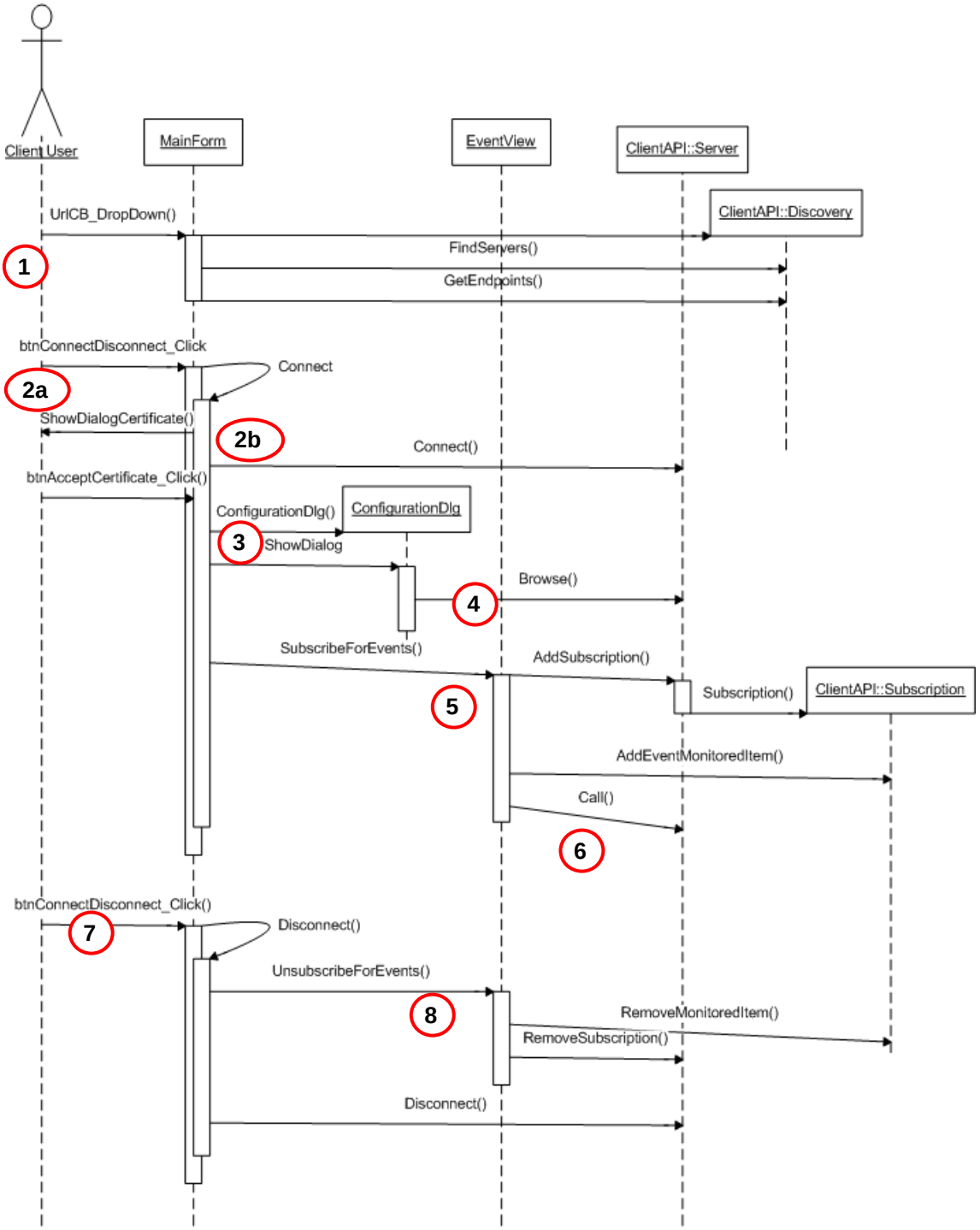


Table 4-11

No	Description
1	When opening the selection list a ClientAPI::Discovery object is created and the methods **FindServers** and **GetEndpoints** are called.
2	The **ConnectDisconnect_Click** method is called at the MainForm object via the “Connect” user action. With this method, the actions to establish a connection to the server are performed. In the first step the ClientAPI::Server object is called and there the **Connect** method (2a). This ensures that the connection to the OPC server, defined by the URL, is established. If a certificate is not considered trusted yet, the user can still verify and accept it and the connecting process is continued (2b).
3	In the case of an existing connection the configuration dialog is displayed in a second step (ConfigurationDlg). Settings regarding area, event fields and filters are made in the configuration dialog.
4	When starting the dialog the server automatically browses for areas, event types and event fields to enable displaying possible settings to the user.
5	A subscription is created in EventView and an event monitored item is added to the subscription.
6	The Refresh method is called. The OPC UA server sends the states of the currently active alarms as events to the client.
7	**ConnectDisconnect_Click** called up at the MainForm object. In the case of an already existing connection, this method calls the **Disconnect** method at the ClientAPI::Server object. This terminates the connection to the OPC UA server.
8	The event monitored item is deleted from the subscription and the subscription created by EventView is deleted.

Establishing a connection to the OPC server and closing it – client API

The sequence diagram below shows the processes during establishing a connection to an OPC UA server in the context of the client API and when closing it.

Figure 4-7

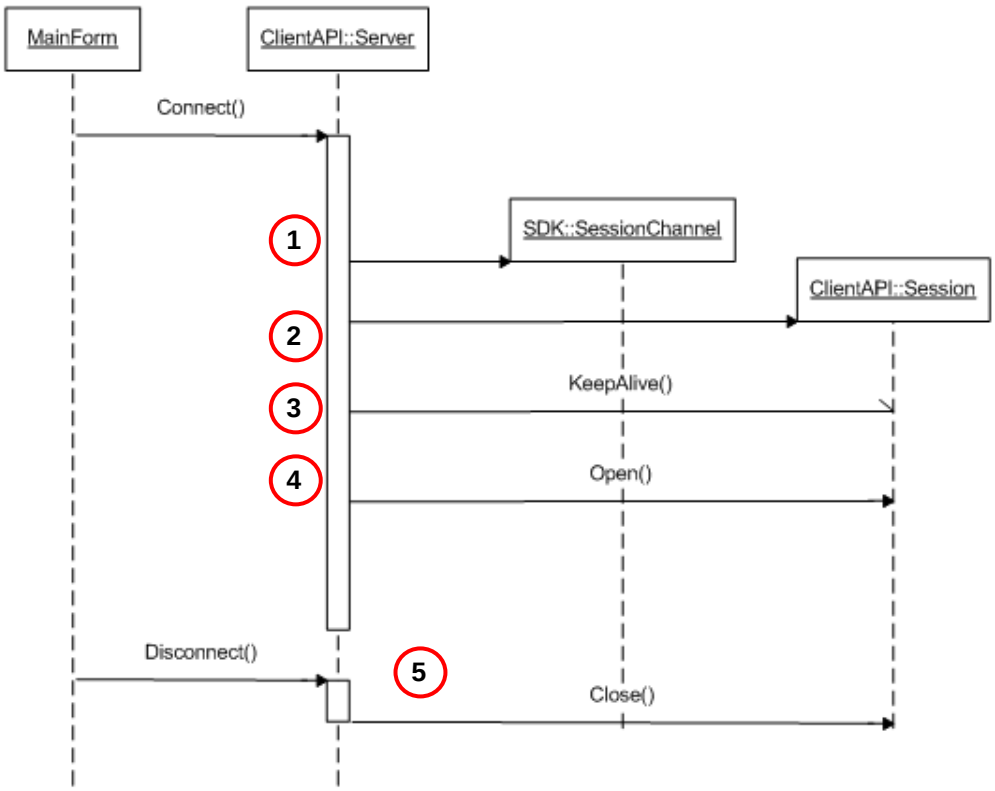


Table 4-12

No	Description
1	The **Connect** action creates an SDK::SessionChannel object for the establishment of a secure connection to the server.
2	Subsequently a ClientAPI::Session object is generated which encapsulates the channel to the server.
3	In the next step the ClientAPI::Session object registers a **KeepAlive Callback** at the OPC UA server.
4	The **Open** call establishes the actual connection between client and server.
5	Within the framework of the **Disconnect** action, **Close** is called on the ClientAPI::Session object.

Changing the event filter configuration

The following sequence diagram shows the sequence steps when changing the configuration for the event monitored items or their event filter.

Figure 4-8

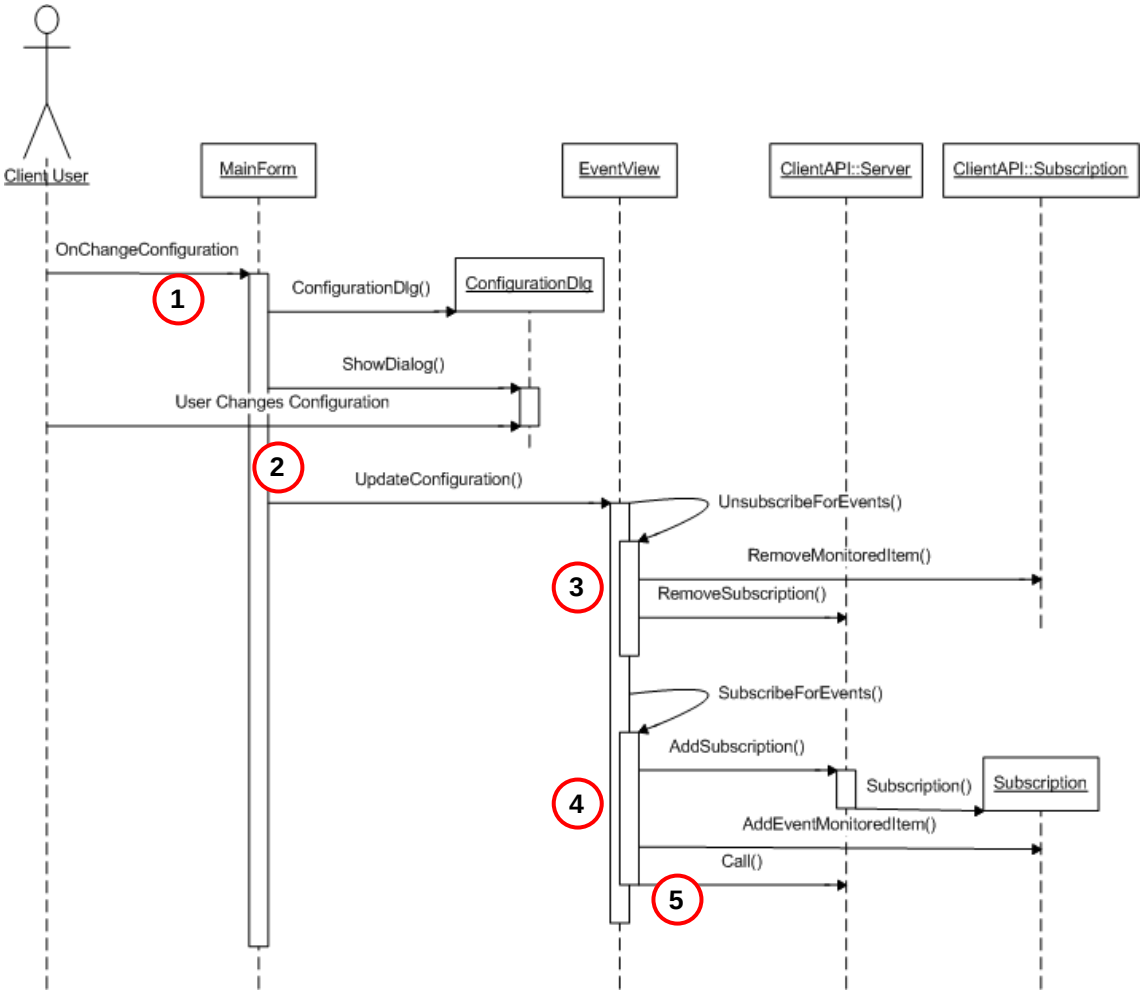


Table 4-13

No	Description
1	The user triggers the ChangeConfiguration action via the EventFiler menu in the menu list. The configuration dialog is displayed.
2	The user changes the desired parameters in the configuration dialog (filter, event types, event fields). If the dialog is closed with **OK,** UpdateConfiguration is subsequently called at the EventView object.
3	The event monitored item is deleted from the subscription. The subscription is deleted
4	A subscription is recreated. An event monitored item is created with the new configuration parameters.
5	The Refresh method is called. The OPC UA server sends the states of the currently active alarms as events to the client.

5 Functional Mechanisms of the S7 Application

The following chapter is addressed to technicians and programmers who wish to gain an overview of the functional mechanisms of the S7 application.

The chapter starts with an overview of the options for configuring and programming of alarms. The following section describes the various alarm types and their informational content. In the last part of this chapter different alarms are generated in an S7 example application.

Type and number of the alarms supported by a S7 CPU depend on their type and version of firmware. A respective overview is available in \1\.

5.1 Extended alarm configuration as of STEP 7 V5.5

This chapter explains the concepts of the alarm signaling method and the basic steps regarding its configuration. The so-called message number procedure is discussed here, which in contrast to the bit message procedure, for which the operating system or the OPC server performs a polled monitoring of individual bits, triggers actually active events from automation system and causes a clearly lower bus load. The message texts for the message number procedure are generated from a common data base and additional texts can also be configured. The messages contain the time stamp from the automation system.

The configuration itself is not new, however, as of version STEP 7 V5.5 the configured alarm information is prepared during “save and compile” for the SIMATIC NET OPC Server (as of V7.1). If the PC station containing the OPC server is downloaded (or XDB Import), the OPC Alarms&Event server also recognizes the configuration and displays message texts, source, and areas as OPC attributes. Configuration via an additional file (scores7.msg), as required in earlier versions of the OPC A&E server, is unnecessary which makes the configuration continuous and consistent.

Introduction

Principally, it is distinguished between three different types of messages, which differ regarding their field of application:

- block-related messages
  Reporting of program-synchronous events, programming via message blocks
- symbol-related messages
  Reporting of program-independent events, configuration via the symbol table (only S7-400)
- user-defined messages
  Reporting of program-synchronous diagnostic events, programming via system functions

Configuration and settings

STEP 7 supports configuration of the message number procedure with the following languages or tools:

- LAD/FBD/STL/SCL
- Symbol table
- PDIAG
- CFC
- S7-GRAPH

In the example on hand, only STEP 7 (STL and symbol table) is used for the configuration.

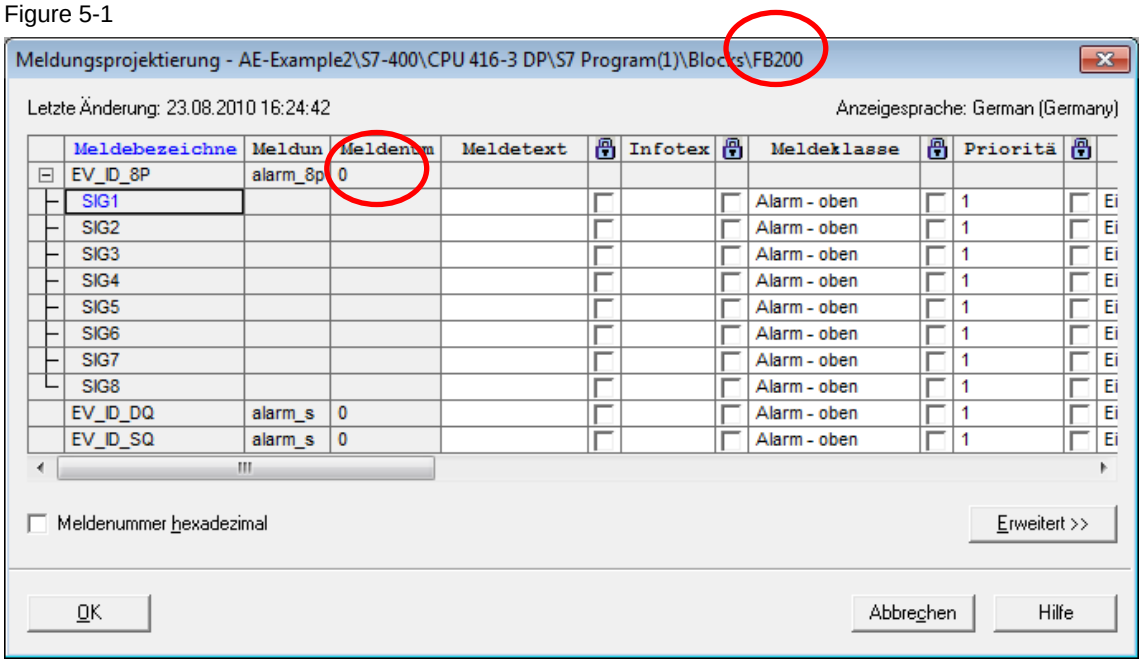
Block-related messages

The message block is the most frequently used alarm. There are various blocks (SFB/SFC) which differ regarding the number of the monitored signals and the number of associated values as well as the acknowledgement capability.

Each of these blocks represents a message type as soon as it has been supplied with the respective formal parameters, e.g. being called up in an FB, therefore representing a message-capable FB. This type can now be configured STEP7 and the properties can be defined or even blocked.

In the Simatic Manager you right-click the FB (which calls the SFB), then “Specific object properties → Message…” and the dialog for configuring the type opens. The message number =0 indicates that this is a configuration of the type.

Figure 5-1



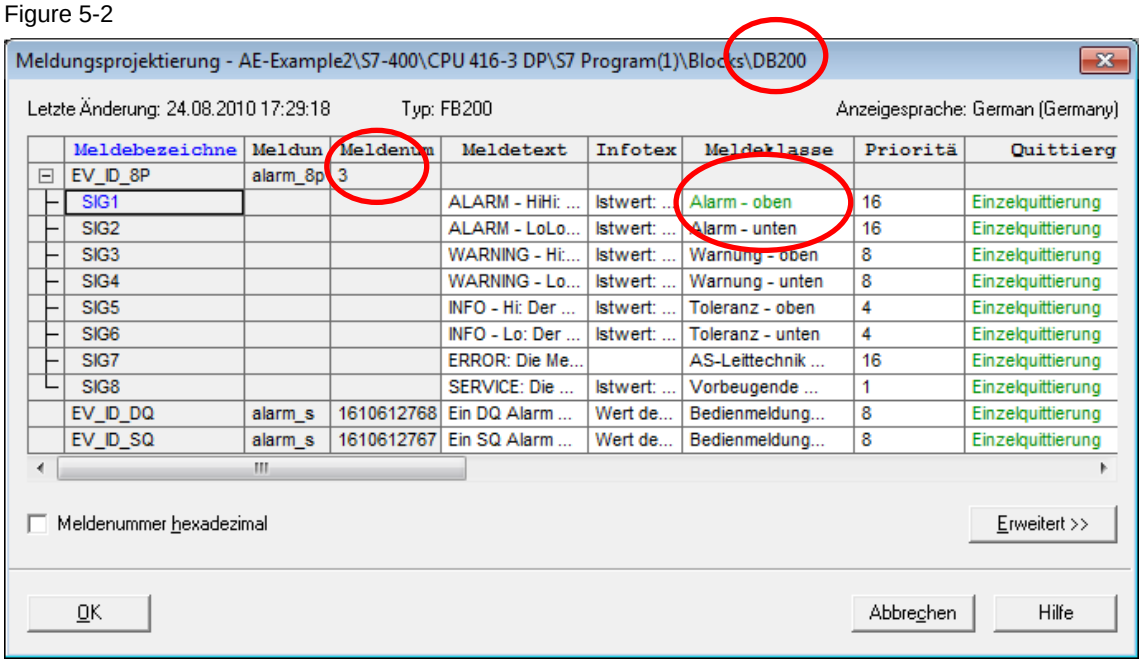
Configuring the type as well as blocking the parameters is always used when a block is to be always used for reporting identical events. Regarding their alarms, analog process tags always have the same structure regarding their alarms, and always supply alarms for exceeding/falling below limit values, exceeding/falling below threshold values and a general error alarm if the process tag is defective. A

respective alarm type is now generated of which later in the program instances are created which always have the same structure.

When calling the FB in OB1 an instance data block is created (Call FB200, DB200). The messages of this instance now look exactly as defined in the message type. The parameters which were not blocked, the instance-specific parameters, can now be filled into the instance.

In the Simatic Manager you right-click the DB (of the multi-instance DB of the FB), then “Specific object properties → Message…” and the dialog for configuring the instance opens. The message number =<no.> (unequal “0”) indicates that this is a configuration of an instance.

Figure 5-2



The parameters are marked in green which still correspond to those of the original type, all others were already changed for this instance.

Symbol-related messages

The symbol-related messages are rare and are only used for monitoring asynchronous events, where asynchronous here refers to the OB1 cycle. It uses up system resources, causes cycle time load for the CPU, and is only possible for S7-400. Monitoring of binary signals is configured via the symbol table. Independent of the OB1 cycle, this bit is monitored (SCAN) in a fixed time grid (100, 500, 1000 ms). The time grid is independent of the CPU type.

After completing the configuration in the symbol table, system blocks (SDB) must be generated and downloaded into the controller.

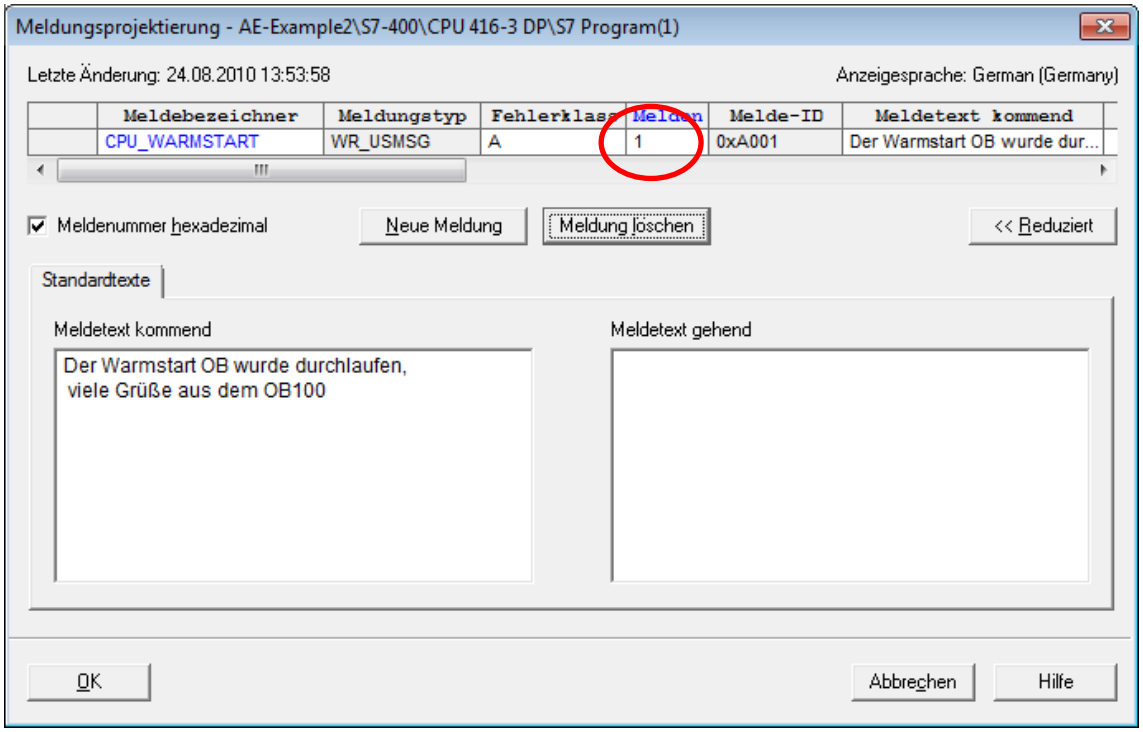
All binary symbols can be monitored (inputs, outputs and memory bits). As soon as a signal change has been detected, the configured “scan alarm” is triggered. This alarm may contain up to 10 associated values.

User-defined messages

The SFC52 system function (WR_USMSG) writes user entries into the diagnostic buffer of the CPU and sends a message at the same time. This type of message is therefore exclusively used for system-relevant events. For two error classes (A and B) 255 respective event numbers are available, for which one text each can be configured for coming and going messages.

In the user program, the system function (Call SFC 52) is then called where suitable (e.g. in OB100, the warm start OB) and supplied with the desired EventID (0xA101).

Figure 5-3



In contrast to the “Report system error” function, the user assigns event numbers here and calls up the SFC 52 in the program. When reporting system errors, the numbers (a specific number range) as well as the texts are given by the system or are compiled by determining diagnostic information during the call.

5.2 Alarms of the SIMATIC S7 station

This chapter shows the possible event messages of a SIMATIC S7 station and describes their display in OPC UA (SIMATIC NET V8.0) in greater detail.

Categories

Event messages that can be signaled by an S7 station are divided into ten categories:

- System message
- Programmed alarm (off normal)
- Programmed alarm (level)
- Programmed alarm (deviation)
- Programmed alarm (rate of change)
- Programmed alarm (trip)
- Programmed alarm (change of state)
- Programmed alarm (device failure)
- Programmed alarm (system failure)
- Connection alarm (statepath)

System message

Events from the class of the system failure message (former diagnostic events) are triggered automatically by S7-CPU, CP, or the I/O module (e.g. DP slaves or PNIO device) and filled into the diagnostic buffer of the respective component. The respective error OBs are called up and 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, unplugging/plugging alarm, etc.)

Events from the class of the user-defined diagnostic messages (SFC52) are principally entered in the diagnostic buffer of the CPU and can be additionally made available as system message via OPC. The WR_USMSG function is called up in the user program which previously was configured with text messages for a coming and going event.

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 5-1

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, hence 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.

In addition to the above described programmed alarms, the S7-400 provides the SCAN alarm, the so-called symbolic message. This is a cyclic monitoring of individual binary signals (input, output and memory bit). The SCAN alarm is configured via the symbol table and should be used “economical” since it causes system load due to the cyclic checking. For the SCAN alarm no program block is called, therefore it is also referred to as “configured” alarm (as opposed to the “programmed” alarm).

Connection alarms

The connection or statepath alarm class is not initiated in the S7 station but in the actual OPC 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.

5.3 Mapping to OPC UA event fields

The OPC UA Alarm & Conditions specification defines event fields that must be contained in an OPC UA alarm and attributes that must 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 event fields and their contents according to the S7 alarms. The default assignments of these fields are described here. Some fields, such as “Source”, “Area”, and “Message” as well as “Time” can also be filled with other contents. (please refer to chapter 5.5, for example)

Event information of the BaseEventType

All events in OPC UA are derived from BaseEventType. The following table shows all standard information supplied with each event:

Table 5-2

BrowseName	Description
EventId	ByteString
EventType	NodeId (NamespaceIndex and category number of the alarm type)
Time	<S7 time when the alarm was called > (changeable via NetPro S7 connection configuration, note: only for statepath the PC time is supplied in UTC)
TimeZone	Time zones deviating from UTC (default zero)
ReceiveTime	Time when the alarm was received by the server (UTC)
Severity	<1..1000> (default=500, can be set via NetPro, S7 connection configuration or directly when calling the block)
Message	<Text#> The text can be changed via STEP7. The configuration is described in chapter 5.1.
SourceNode	NodeID of the source <ConnectionName\Path-to-the-block> (can be modified via the STEP7 block configuration)
SourceName	<ConnectionName\Path-to-the-block> (can be modified via the STEP7 block configuration)

Note

For some blocks, e.g. ALARM, ALARM_8P or NOTIFY, the severity (0-16) 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 event fields for ConditionType events

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

Table 5-3

BrowseName	Description
ConditionNodeID	String NodeID <WindowText#> (given text, consisting of symbolic ConnectionName.Alarm<Number>)
ConditionName	<symbolic name of EventID,SubConditionName
Retain	Information whether or not the alarm shall be displayed

Additional event fields for AcknowledgeableConditionType events

The following table shows fields that are supplied for “AcknowledgeableConditionalType events” in addition to the ones shown table 5-4.

Table 5-4

BrowseName	Description
AckedState	Shows the acknowledgement state, either “Acknowledged” or “Unacknowledged”
EnabledState	Always enabled

Attributes for Category 40 to 47 (Programmed)

In further event fields, programmed alarms of category 40 – 47 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 an alarm as event fields.

Table 5-5

BrowseName	Description
S7AlarmAddData1	Associated value 1 (value and data type)
…	
S7AlarmAddData10	Associated value 10 (value and data type)
S7AlarmAddText1	Text (configured via Step7), special meaning SourceName
S7AlarmAddText2	Text (configured via Step7), special meaning AreaName
...	
S7AlarmAddText9	Text (configured via Step7)
S7AlarmId	<Event ID> (VT_UI4) = “3”
S7Time	<S7 Zeit> (VT_Date)
S7Connection	String NodeID (symbolic connection name)
S7AlarmSubId	<Subevent ID> (VT_UI4) = “3”
S7AlarmState	AckState (status word acknowledgement state), EventState, State

Event fields for Category 14 (Statepath)

In event fields, connection alarms of Category 14 provide fixed information as shown in the following table.

Table 5-6

BrowseName	Description
Message	“statepath”
AckedState	“Acknowledged”
ConditionName	“statepath”
Retain	“True”
Time	PC time in UTC
ReceiveTime	Same as Time

5.4 S7 Program of this example

Introduction

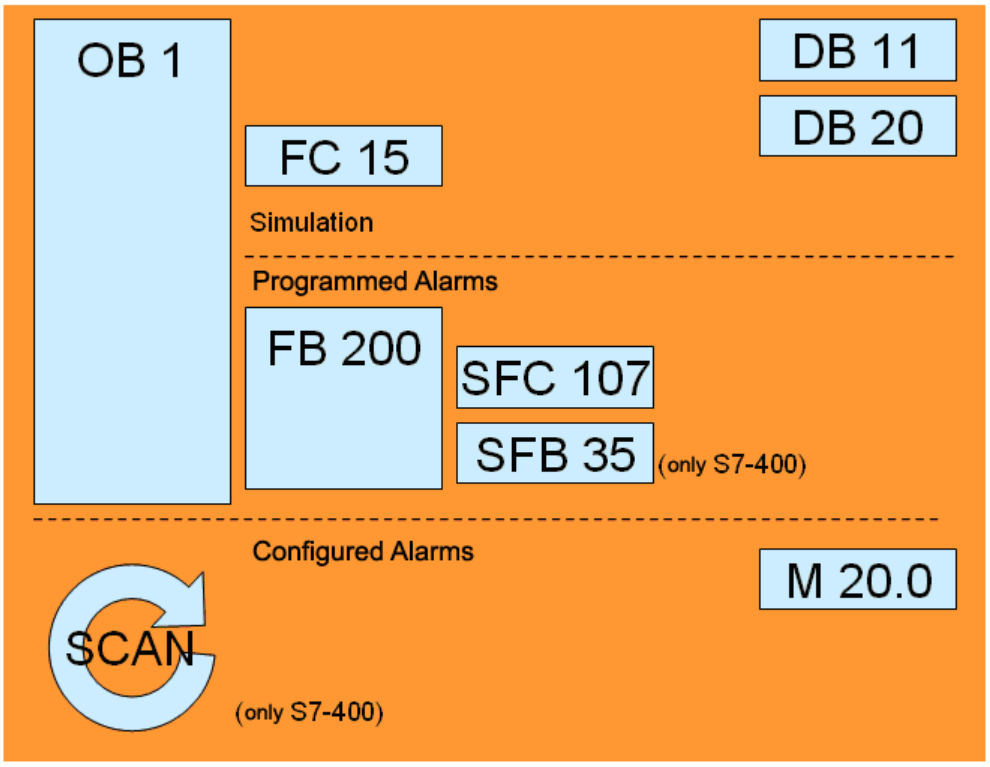
A block call that is preferably executable in all S7 CPUs and that illustrates the basic functions of the “programmed alarms” was selected for this application example. For reasons of clarity, the status word was not evaluated.

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

The S7 program is essentially divided in two parts. Firstly, the binary signals for the alarms are simulated, secondly the alarm blocks are interconnected to the signals and called in FB 200.

Figure 5-4



Simulation of the binary signals

The table below gives a brief overview of program parts and their function for signal simulation. It was a deliberate decision to avoid details; further comments can be found in the STL code.

Table 5-7

Block	Comment
OB1	Cyclic Main first of all, a variable timer is used here whose interval is used for calling further program functions. The data change rate can be set via DB11 byte 0.
FC15	ChangeSignalBits Toggles Boolean variables in DB11 as well as the memory bit 20.0 (symbolic message).
DB11	SimulationAlarms contains global variables for the configuration of data simulation.
DB20	AssiciatedValues contains some exemplary associated values to be sent with triggered alarms.

Programmed alarms

The table below gives a brief overview of program parts and their function for signal simulation. It was a deliberate decision to avoid details; further comments can be found in the STL code.

Table 5-8

Block	Note
OB1	Cyclic Main Calling the alarm block (SFC107) for ALARM_DQ and the alarm block (SFB35) for ALARM_8P via the function block 200. SFB35 only exists in S7-400.
SFB35 + multi-instance DB200	ALARM_8P The alarm block has 8 signal inputs as well as 10 possible associated values. It is interconnected with 8 signal bits and the associated values from DB20.
SFC107	ALARM_DQ The alarm block has one signal input and can supply one associated value.

Configured scan

A scan alarm was configured in S7-400 which monitors memory bit 20.0 in 500ms intervals.

5.5 Example configuration of a SCAN alarm

Introduction

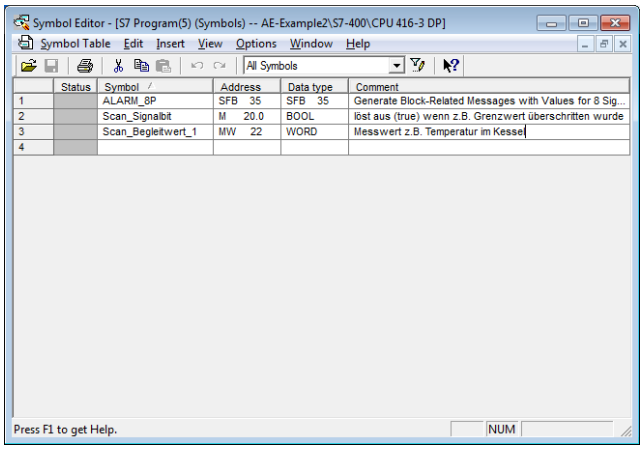
S7-400 provides the option of configuring symbol-related messages. These so-called SCAN alarms are configured via the symbol table of STEP 7. An example SCAN alarm is configured below to illustrate the principle procedure.

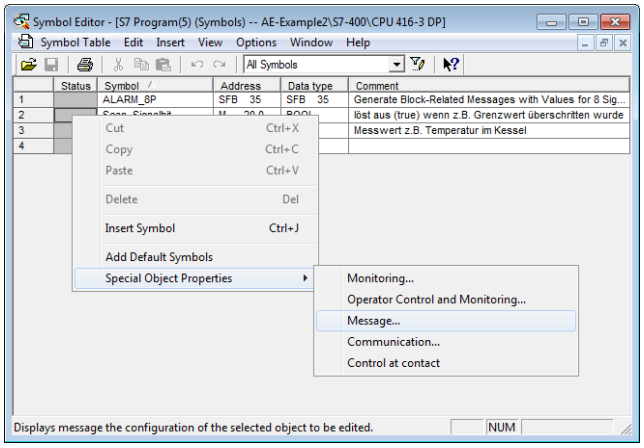
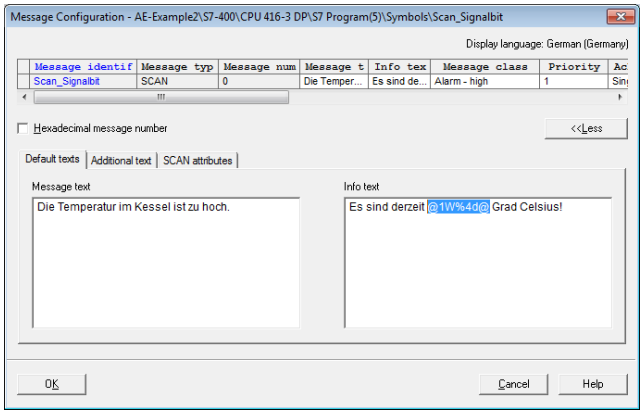
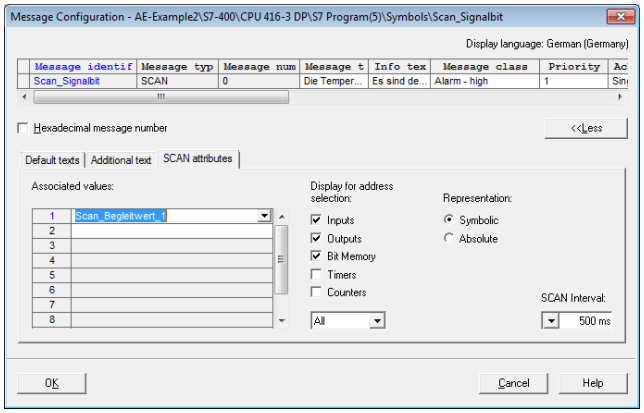
Note Symbol-related messages (SCAN) are only available in S7 400 CPUs. They are transferred via system data blocks (SDB); accordingly, these must be downloaded for modifications, as well as the PC station (OPC Server).

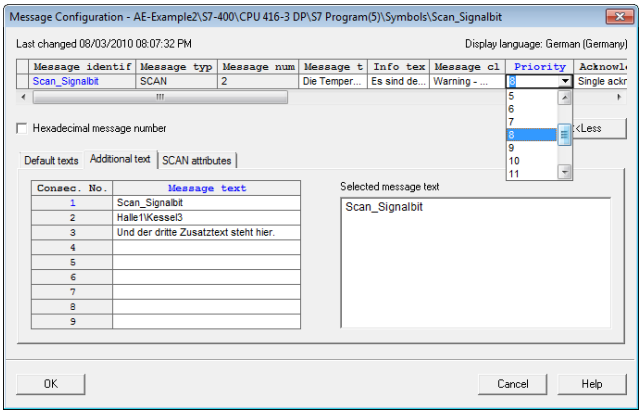
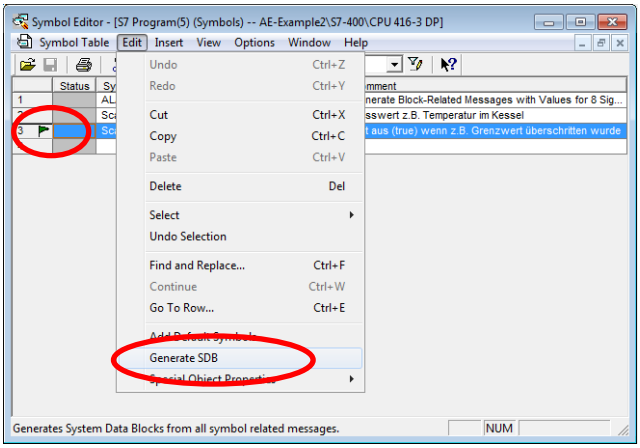
Procedure

For symbolic messages binary signal states are checked acyclical to the program sequence. This makes them particularly suitable for events which are not directly related to the program flow. When, for example, the S7 program depicts a step chain for controlling a goods elevator, the temperature monitoring of the drive motor is monitored with a scan alarm. The end position switches of the doors, on the other hand, would be depicted with programmed alarms (SFC) since they directly influence the program sequence.

Table 5-9

No	Action	Note
1.	Open the Symbol Editor and add two symbols. A memory bit, which represents the signal to be monitored, and a memory word to be used as associated value.	After the symbols were created, the symbol file must be saved. 

No	Action	Note
2.	Right-click the symbol to be scanned (here “Scan_Signalbit”) to open the configuration dialog with “Special Object Properties→Message…”.	
3.	Show the extended view in the configuration dialog of the message configuration.	Message text and info text can be typed in over several lines. If associated values shall be displayed in the message text, they must be preceded and followed by “@”. 
4.	Edit the dummy according to the desired format.	The dummy has the following structure: @<No. of the AssociatedValue><ElementType><Format>@. The format is preceded by “%”, followed by the number of digits and the format identifier. For a detailed description please refer to the STEP7 online help.
5.	Configure the associated value in the “SCAN attribute” tab.	Up to 10 associated values can be configured; filters can be set for a better overview in order to facilitate finding the desired associated values. The SCAN grid is default at 500ms. 

No	Action	Note
6.	Up to 9 additional texts can be specified in the “Additional text” tab. The OPC “Source” attribute is formed from the first additional text, and the “Area” attribute is formed from the second additional text. Further settings such as alarm class and priority can be set separately for each scan alarm. A “Warning – high” is set here as an example. The priority can be set between “0” and “16”.	As opposed to programmed alarms, for configured, symbolic messages a lower value does mean a lower priority. “8” is set here to obtain an average value of 500 as the OPC severity. 
7.	After the configuration of the SCAN alarms has been completed the system data blocks (SDB) must be regenerated and downloaded to the S7-400.	The green flag indicates that a scan alarm has been configured for this symbol. 

Note For each configured SCAN rate system data blocks are generated. The configured signals are checked asynchronous to the running program. It must be noted, that the SCAN function causes system load and should therefore only be used on a small scale.

Note The PC station (OPCServer) must be recompiled (e.g. in NetPro) and then downloaded again.

5.6 Call of an ALARM_8P as an example

Introduction

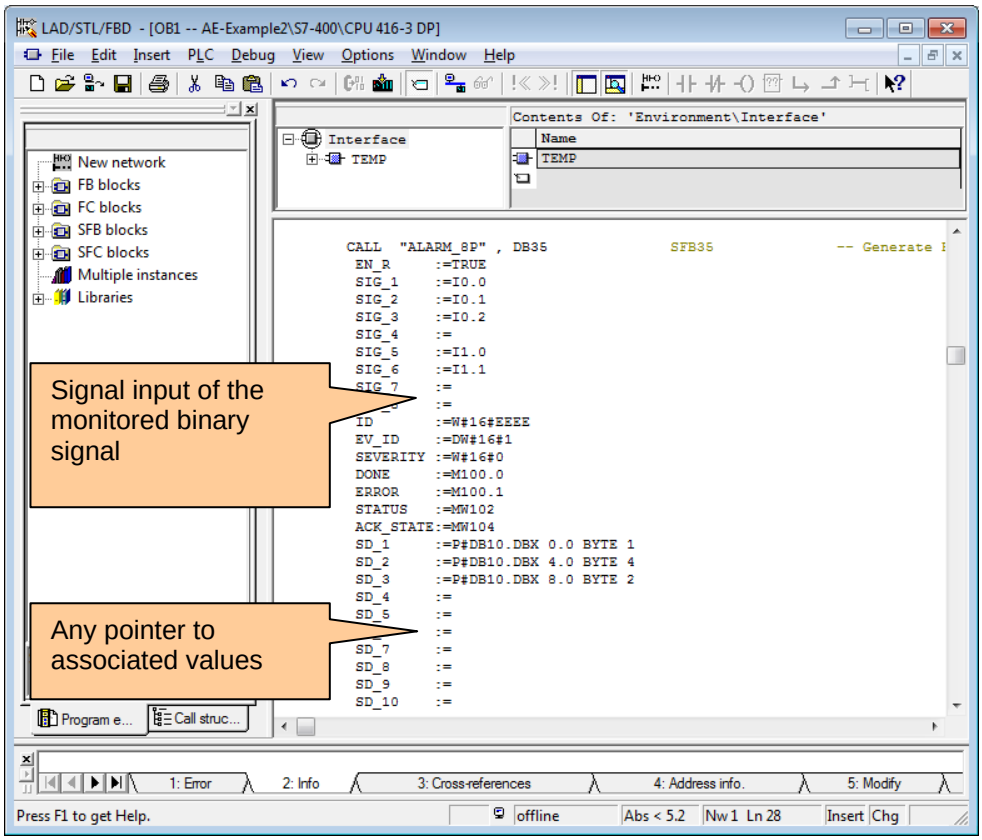
The most frequently used alarm block in S7-400 stations is SFB35 (ALARM_8P). This block provides the maximum functionality possible with regard to Alarm and Events. The call parameters are explained as examples and their meaning for OPC Events is explained.

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. If the SFB is called up within a FB, the parameters can be stored in the multi-instance DB. The required system attributes (e.g. S7_server and S7_a_type) and the respective values are automatically assigned as soon as a symbolic IN parameter has been created here and specified at the SFB 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 5-5



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.

Associated value parameters

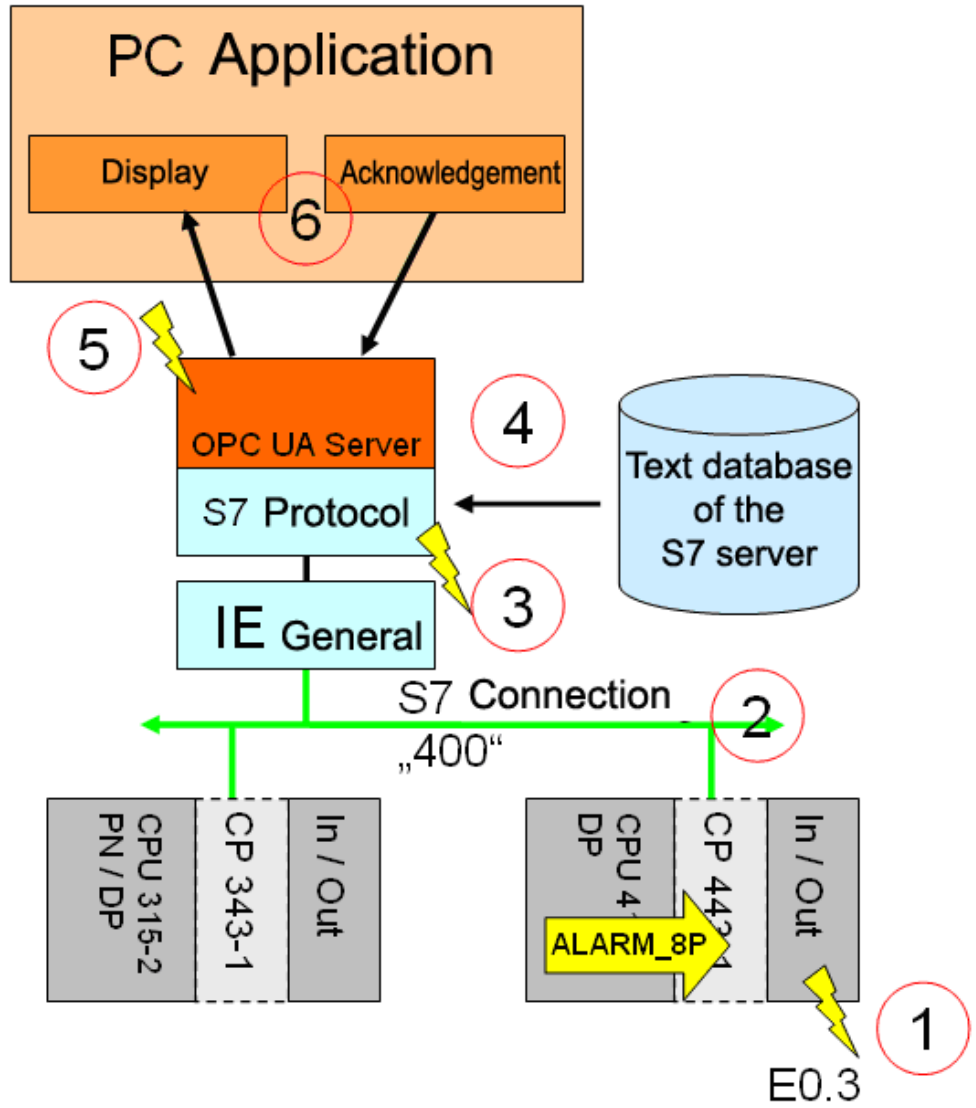
Up to 10 associated values can be parameterized. These values are ANY pointers that point, for example, to the DB10 data block as shown here. The value included there is supplied as an associated value when triggering the alarm.

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 5-6



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

Table 5-10

No	Description
1	The status of the E0.3 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**. The “ALARM1,3” identifier (event ID=1 and signal =3) is assigned to the event and the event fields 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 reported to the OPC clients, the text database of the S7 server is searched for a possibly stored message text, area or source for “ALARM1,3”; if yes, the respective **OPC event fields** are filled. 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 UA server sends a message to the S7-400 CPU. The status can be checked in the flag word

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”.

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

5.7 Example configuration of system error messages

Introduction

In the SIMATIC stations there is the option of configuring messages which describe system errors. Blocks and text messages required for this are preconfigured and read out hardware information, for example, and add these to texts. An example of how these messages are generated is illustrated below.

Note The blocks and message texts required for this are generated by STEP 7. The created blocks must only be downloaded to the CPU by the user. The OPC server must subsequently also be reloaded in order to update the text database.

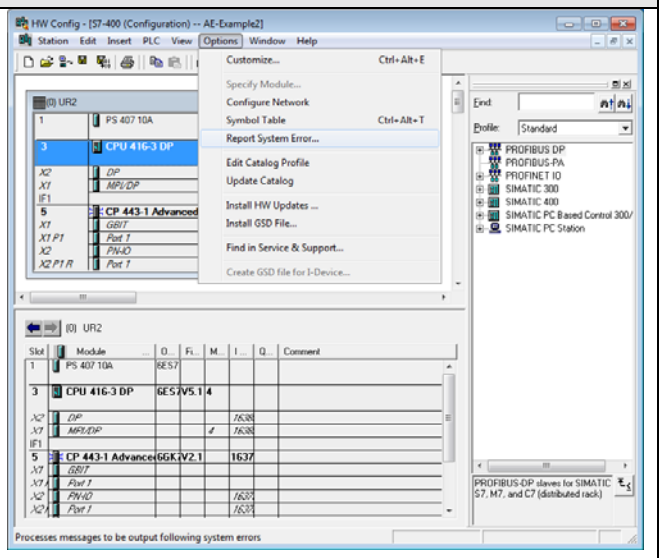
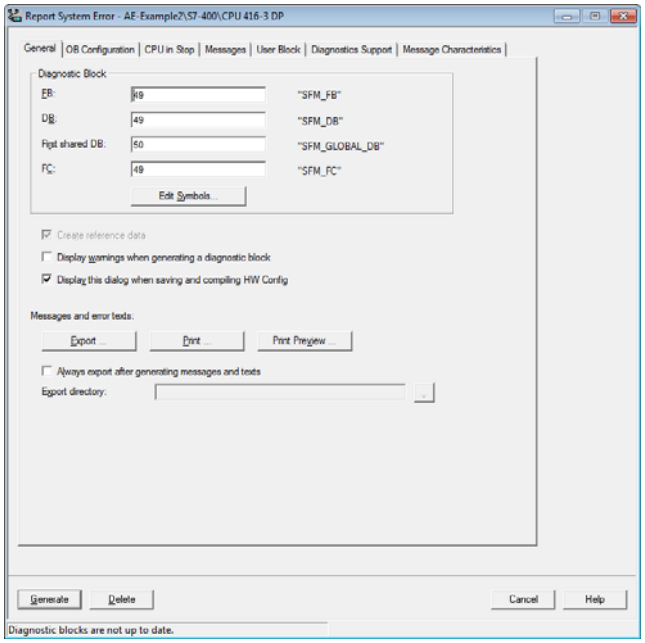
Procedure

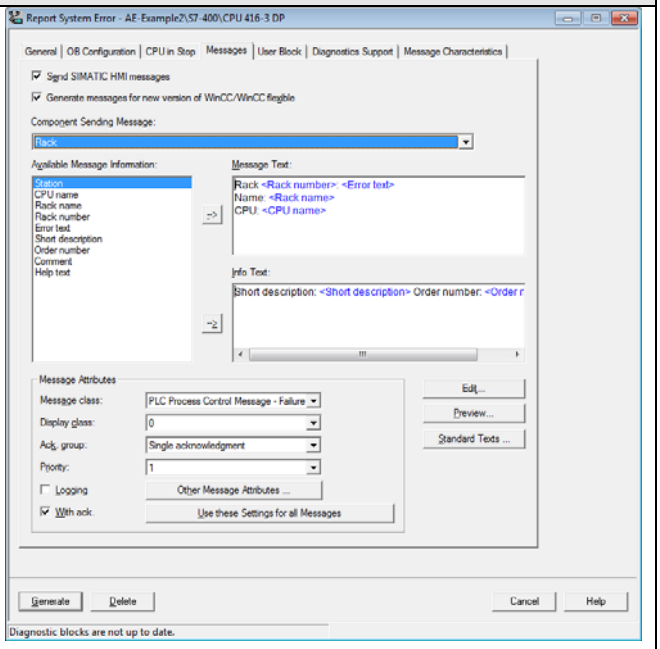
The components of the S7-300 stations, S7-400 stations, PROFINET IO-Devices, DP slaves and WinAC are supported by "Report System Error" as long as they support functions such as diagnostic alarm, unplug/plug alarm and channel-specific diagnostics. The diagnostic data records according to the Profibus/Profinet modules are read via SFB52 (RDREC) and SFB54 (RALRM).

The required blocks are generated by STEP 7.

Table 5-11

No	Action	Note
1.	In HW Config you select the CPU via Options→Report System Errors…	These steps must be repeated for each CPU for which system errors shall be reported

No	Action	Note
		
2.	Target FB and DB number are defined in the “General” tab.	It must be ensured that this number has not been used elsewhere in the program. If necessary, make the corrections here. 
3.	Configure the “Messages” tab.	For each reported component the correct message class should be assigned here as well as the priority be adjusted accordingly.

No	Action	Note
		
4.	Click on the Generate button to generate the blocks.	After the blocks have been generated, the OPC server must also be reloaded (e.g. NetPro -> Save and compile all and then download.

Principle of operation

The diagnostic block created by 'Report system error' (FB with assigned instance DB and one or several global DB(s) and one FC) evaluates the local data of the error OB and reads additional diagnostic information of the hardware component which triggers the error.

If now a module is unplugged/plugged the respective error OB (OB8x) is called. In this OB the SFM block is called which creates the message (internally an alarm_s is triggered).

Depending on the used CPU, error OBs 7x (redundancy errors) and error OBs 8x (time/ hardware errors) are created. Should these already exist, the code for creating messages, the call of the generated FB is attached at the end.

Note For a detailed description of the individual parameters, please refer to the STEP7 online help.

5.8 Notes on the alarm configuration of S7-300

General information

The controller of the SIMATIC S7-300 family only supports the alarm blocks ALARM_S and ALARM_SQ as well as blocks ALARM_D and ALARM_DQ. (D and DQ only as of firmware version > 2.5.0). The S7-300 supports system error messages.

Furthermore, it depends on the type of CPU how many alarm blocks can be called simultaneously since this uses up system resources. Depending on the CPU, 20, 40 or 300 alarms may be pending simultaneously. An overview is available in \1\.

Configuration of alarm texts

The configuration of the alarm texts with STEP7 requires a project where the S7-300 as well as a PC station with OPC server exist. An S7 connection must exist between these two components. Only then will the alarm configuration, including the text database for the OPC server, be generated during “Save and compile”. The OPC server must be downloaded each time as soon as the alarm texts were modified.

Note A configuration of alarm texts for a unilaterally connected S7-300 via STEP7 is only possible as of STEP7 version 5.5 including HotFix 4.

6 Configuration and Settings

The following chapter is addressed to technicians and programmers who wish to configure and commission alarms on a system.

The following chapter gives a detailed description of the configuration of the S7 controller, the PC station with OPC UA server and the configuration of the OPC UA server and its security settings.

6.1 Configuring the SIMATIC S7 stations

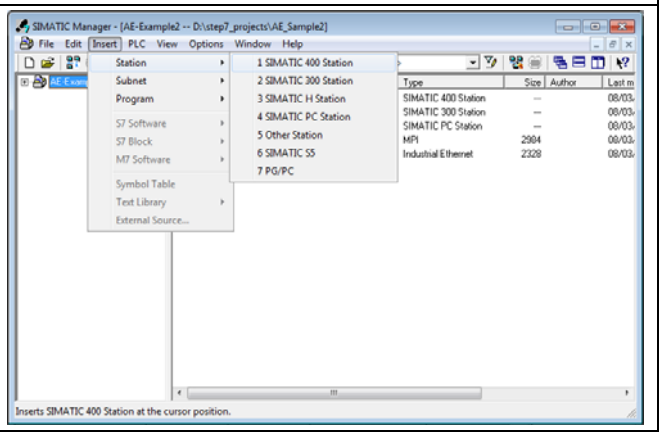
It is assumed that all hardware and software components have been successfully installed and cabled.

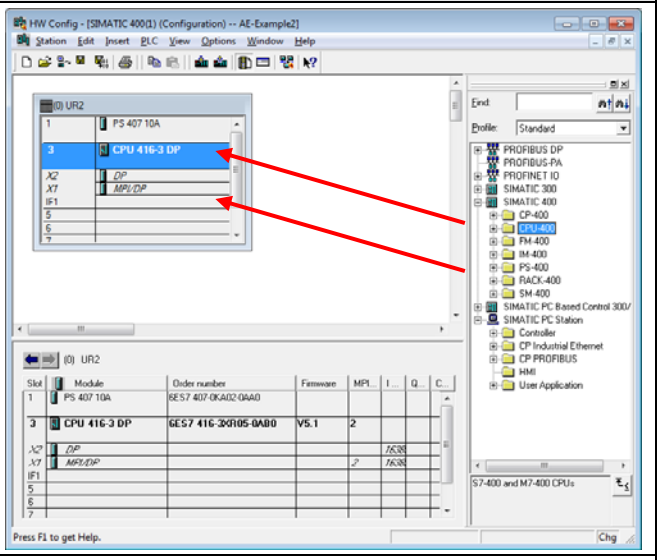
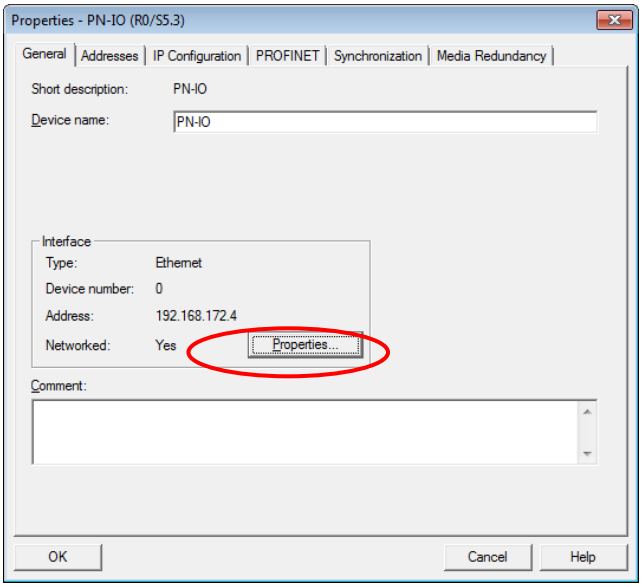
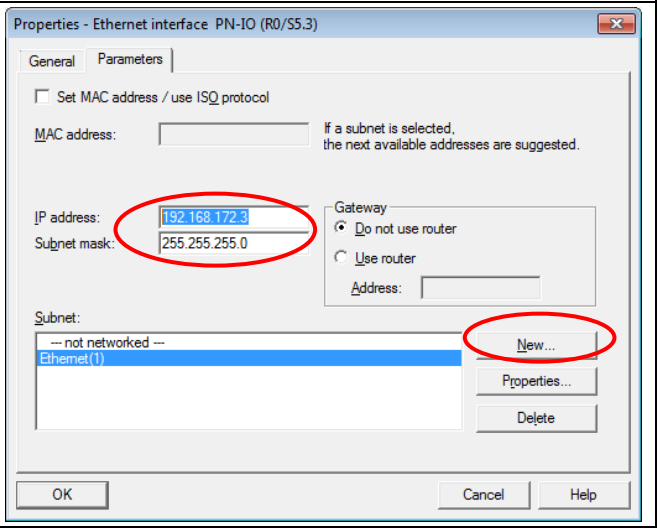
The following configuration steps of the SIMATIC S7 stations exemplify the procedure. Adjust the configuration independently, as required for your hardware.

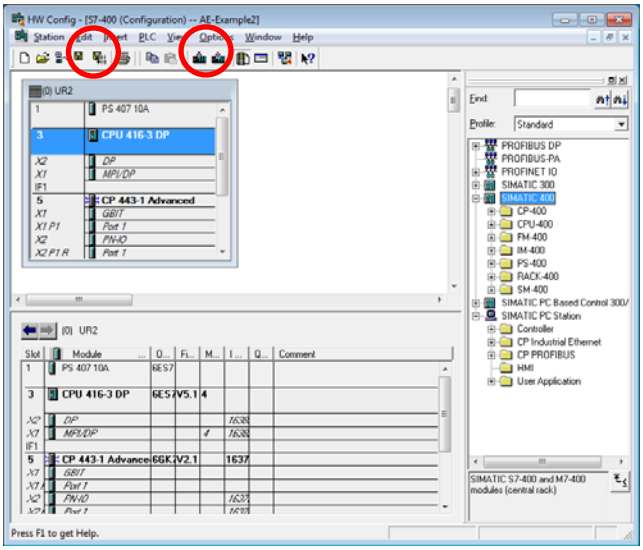
Note After saving and compiling, all configuration information is overwritten.

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

Table 6-1

No	Action	Note
1.	Start STEP 7: Open the SIMATIC Manager and create a new project.	The name “AE-Sample2” was used here.
2.	Add a SIMATIC 400 station and assign a name (here “S7-400”). Add a SIMATIC 300 station and assign a name (here “S7-300”).	

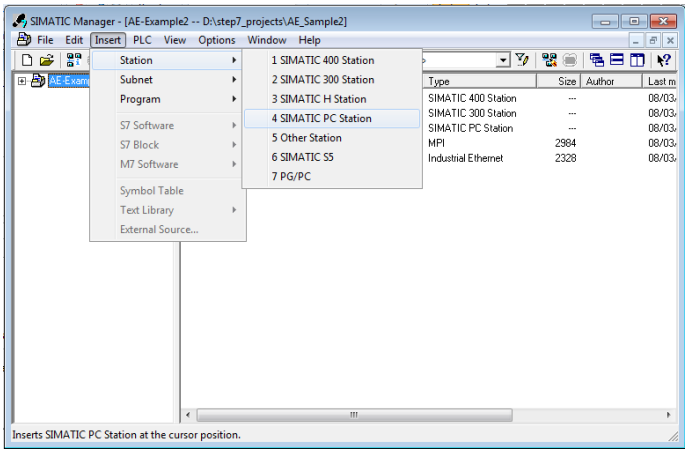
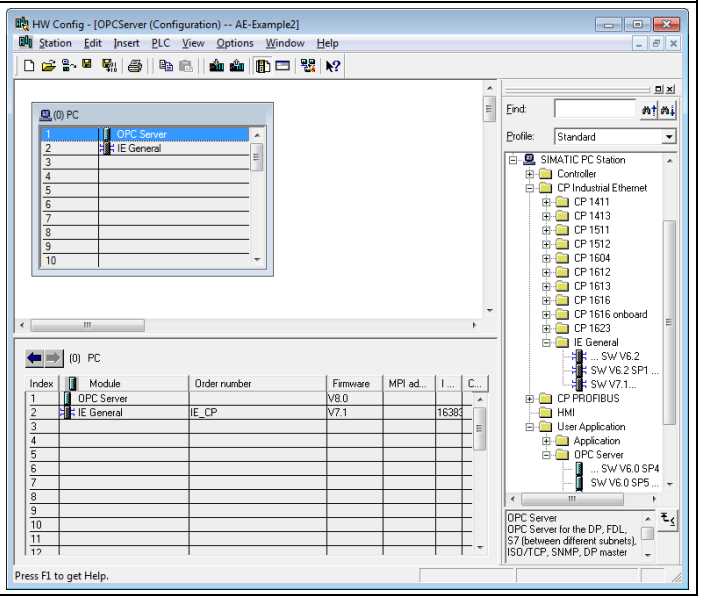
No	Action	Note
3.	Open the SIMATIC stations with HW Config and add the CPU and CP as well as other components.	
4.	Open the Properties dialog and set the IP address.	
5.	Set the IP address (here 192.168.0.52) as well as the associated subnet mask. Create an Ethernet network. A MAC address is only entered if the station is to communicate via ISO transport layer 4.	

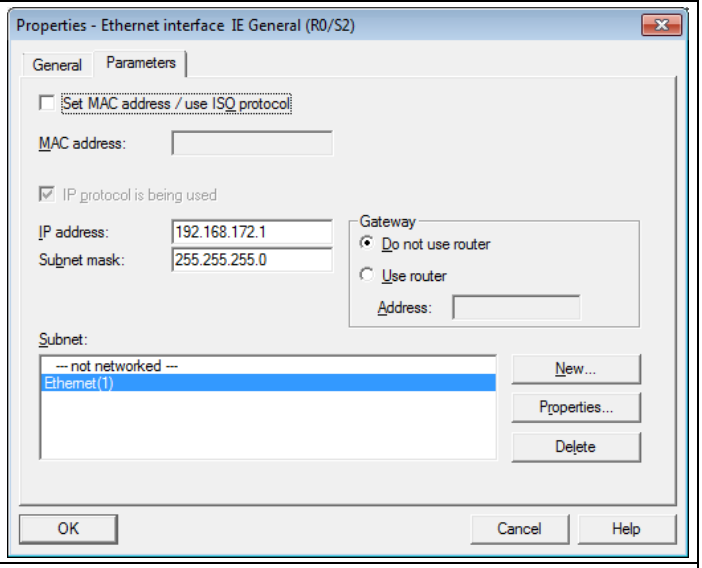
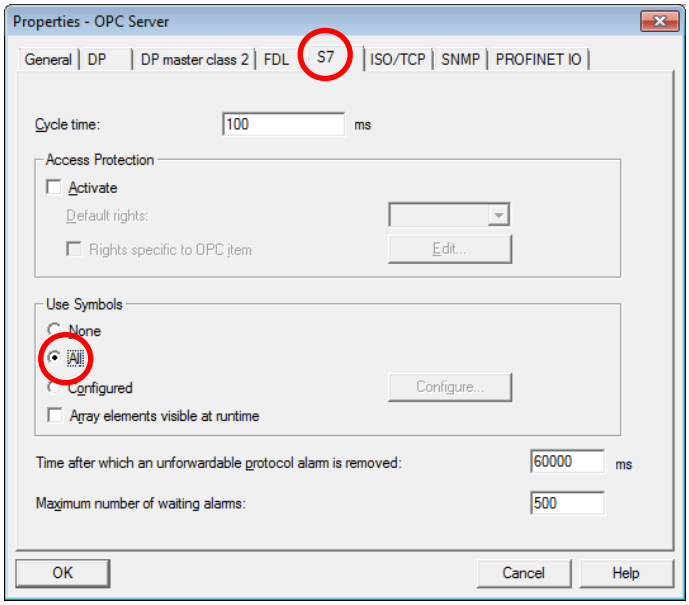
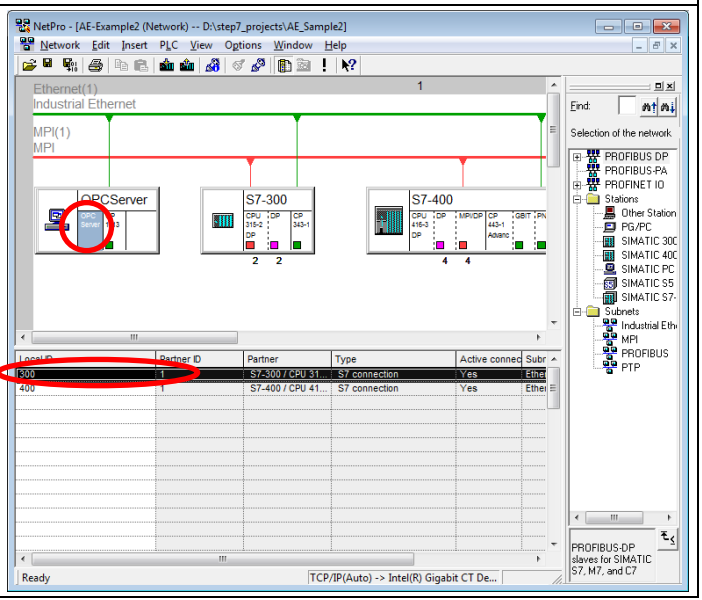
No	Action	Note
6.	Repeat the steps for both SIMATIC stations and then load both stations.	Perform this step for the S7-300 and the S7-400 station. 
7.	Restart the modules.	The stations are restarted (warm restart). Confirm the respective dialog with “YES”.
8.	Create S7 connections	The connection configuration is described together with the configuration of the PC station (in chapter 6.2)

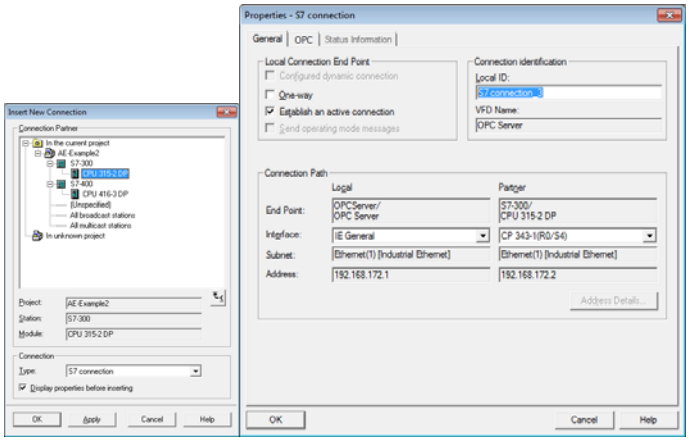
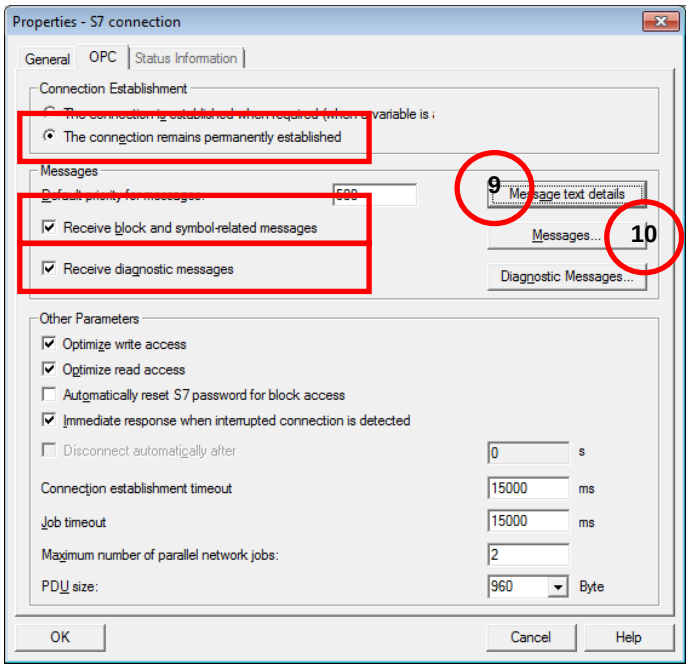
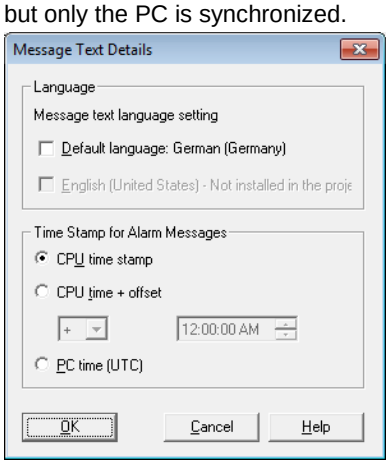
6.2 Configuring the PC station

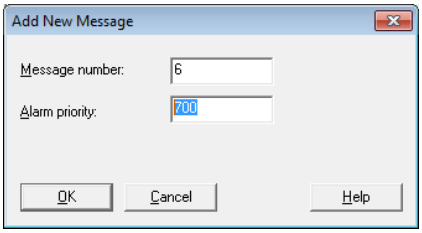
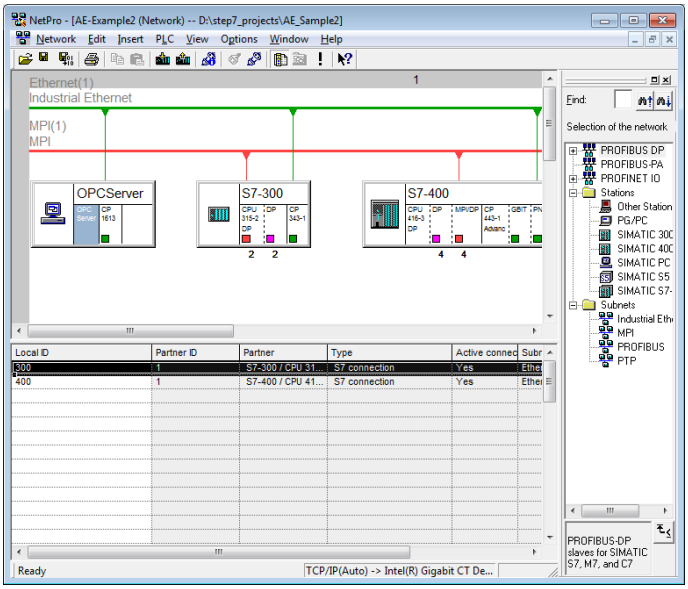
The configuration of the SIMATIC PC station is performed with STEP 7 and described step by step. Alternatively, the NCM PC software package can also be used for the configuration. The procedure is identical, but unilaterally configured connections are used.

Table 6-2

No	Action	Note
1.	Start STEP 7: `SIMATIC Manager` and open the project.	Open the previously created “AE-Sample” project.
2.	Insert a SIMATIC PC Station and assign a name. The name of the PC station must be identical with the “Windows name” of the PC (see `Workstation` → `Properties` → `Computer name`).	
3.	Open the PC station with `HW Config` and add the OPC server and as well as the CP. The slot must be identical with the index which was assigned in the configuration console, here index “2” for Ethernet card. The “OPC Server” application was inserted in slot “1”.	

4.	For the Ethernet card you assign an IP address (here “192.168.172.1”) and connect the card with the Ethernet network.	
5.	On the property pages of the OPC server (double-click on OPC server) the use of the symbolic addressing is activated in the “S7” tab.	
6.	`NetPro` is used for creating two S7 connections from OPC server to both 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, you select the second tab (OPC Connection Parameter). Connection-specific settings are made here. Set the Connection Establishment to “permanent” so the connection can be maintained even through times without any communication. Configure the connection for the transmission of “Block and Symbol-related Messages” and “Diagnostic Messages”.	
9.	Set the language (for multi-language messages) and the time stamps. “Messages” button (see step 8) The dialog of the used time stamp is accessed via the “Message text details...” button.	The message concept is designed for different time stamps. Here it is configured which time stamp shall be used for the message. Either the S7-CPU time (should be synchronized accordingly), the S7-CPU time in addition to a fixed offset (e.g. in order to form a certain time zone), or the PC time of the OPC servers in UTC (this setting is used, for example, if not the S7 but only the PC is synchronized. 

10.	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 reached via the “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 priority” always prevails. 
11.	After the connections have been created and configured, the project must be compiled with “Save and Compile”. Then the stations must be downloaded. As described in chapter 7.4, the PC station can also be downloaded with the XDB file.	

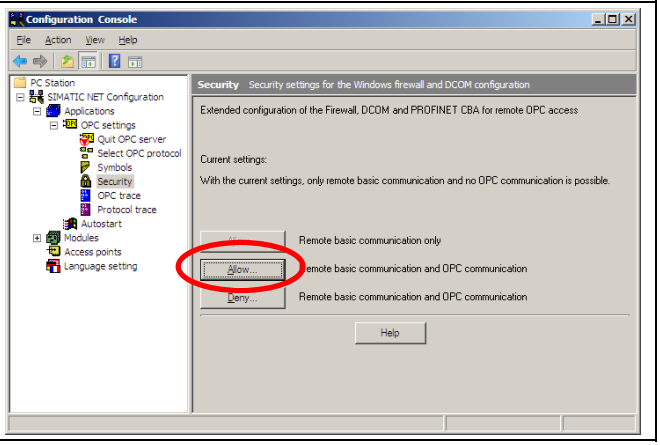
6.3 Configuration of the OPC UA security

The security mechanisms of the OPC Unified Architecture are set at different levels. Encryption and signature of the transmission as well as authentication for the connection establishment can be set separately from each other. After the installation of the SIMATIC NET OPC UA server, secure connections are generally possible. Apart from this encrypted communication, non-encrypted connection is also possible. The server accepts authentication with user and password or also the anonymous connection establishment. These settings are “insecure” and are only used to simplify commissioning. The OPC UA server can be configured in a way that it only accepts an encrypted transfer with user authentication.

6.3.1 OPC UA remote communication

All settings necessary on the server side, regarding the Windows firewall can simply be set and can also be removed again with the “Set PC Station” (Configuration Console) configuration tool.

Table 6-3

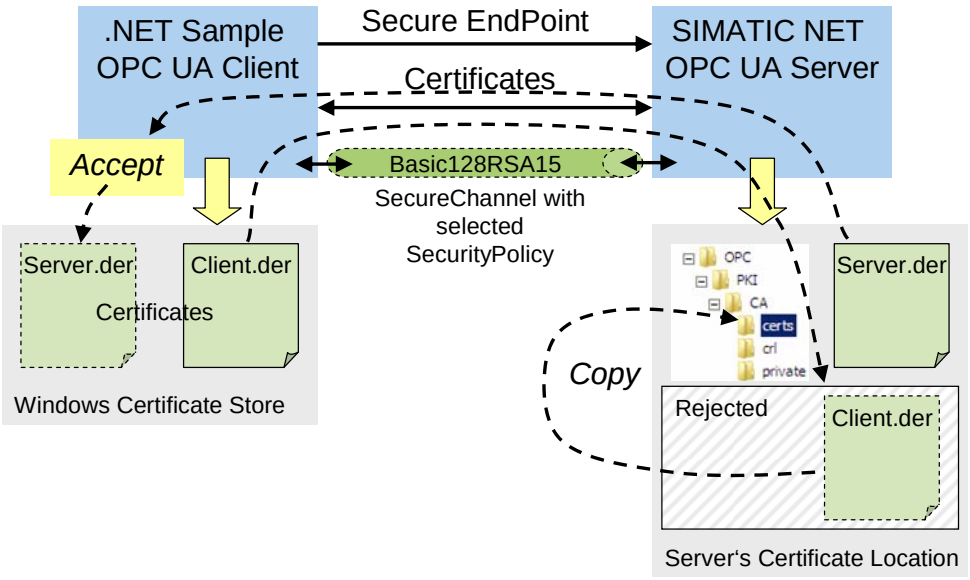
No	Action	Note
12.	Start the configuration tool: `Configuration Console` and select the "`Security`” sub-item.	On the “OPC Server Station” server PC: Start→Programs→Simatic→SimaticNET→Configuration Console
13.	With a single push of a button all necessary settings are performed in the firewall to permit remote communication or to block it again.	

Note Please note, that an exception for the application and for the TCP port (4845) also has to be entered at the firewall of the “OPC Client Station” PC.

6.3.2 Certificate storage

Certificates are exchanged when establishing a secure connection between OPC UA client and OPC UA server. Both applications have to check and accept the corresponding certificate of the counterpart so that a connection can be established.

Figure 6-1



The OPC UA client of this example uses the Windows Certificate Store. This is where the public certificate of the client is located. When establishing an encrypted connection, server and client exchange their certificates. The client displays the certificate and the user has to trust this certificate. By accepting, the server certificate is stored in the Windows certificate store.

The OPC UA server uses its own certificate directory and is independent from the Windows certificate store. First of all, the OPC UA server will reject each certificate of an unknown client and will saves it in a “rejected-folder”, for reasons of security. An administrator has to copy this client certificate in the list of trusted certificates, just as with other server services, to allow the corresponding client access to the server. The location at which the OPC UA server stores and manages its own and the certificates of the OPC UA clients, is the data directory of the OPC UA server.

C:\ProgramData\Siemens\OPC\PKI\CA

This is where three subfolders are located with the following content:

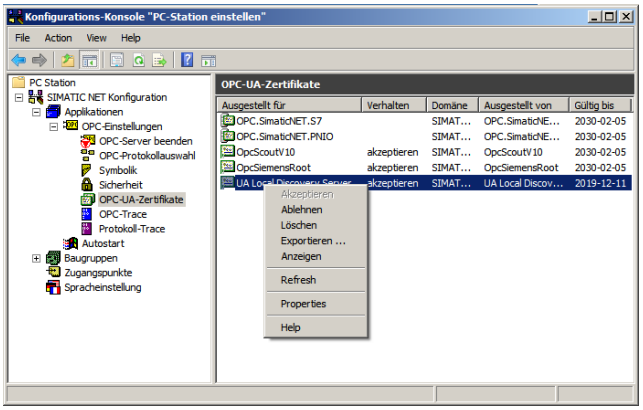
- **\certs**
  contains the public certificate of the server as well as all trusted certificates of clients. Public certificates from OPC UA clients have to be copied in this folder so that the server accepts them.
- **\crl**
  contains a file with a list of untrusted certificates, the so called “RevocationList”
- **\private**
  contains the private certificate of the OPC UA server. This certificate must not be accessible to anybody.

The server independently creates the \reject\ folder underneath of \certs\ and first of all saves all unknown client certificates in this “rejected-folder”. By simply “moving” the file, the certificate can be made trusted.

Configuration server with the SIMATIC NET CD 2010 (V8.0)

From SIMATIC NET CD V8.0 this is possible with the “Configuration Console” configuration tool.

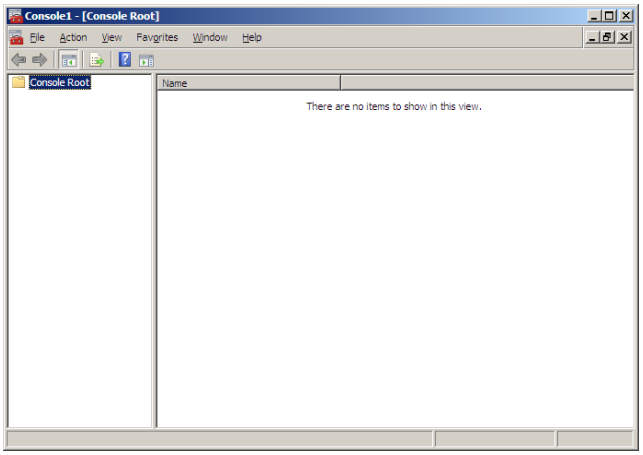
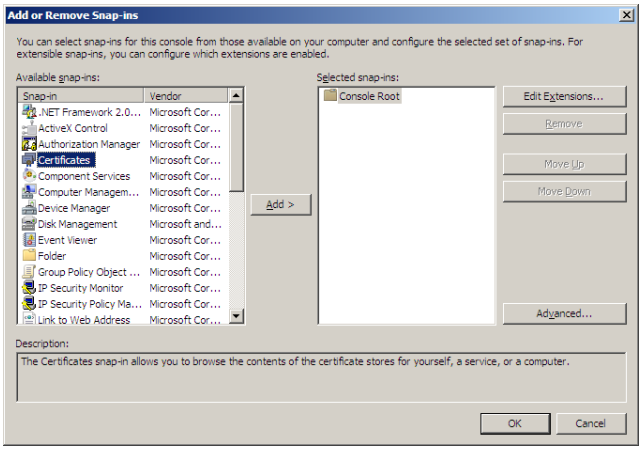
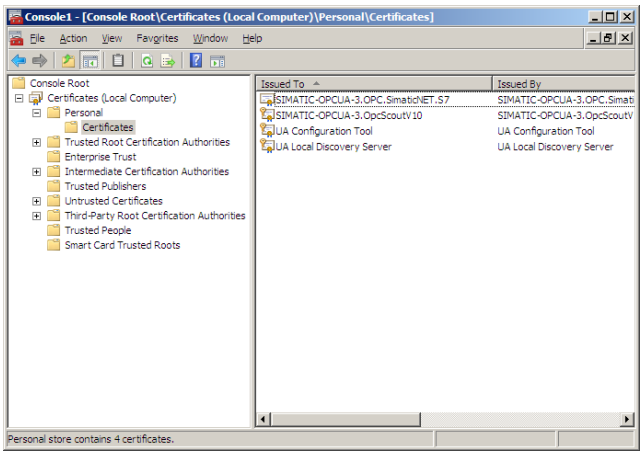
Table 6-4

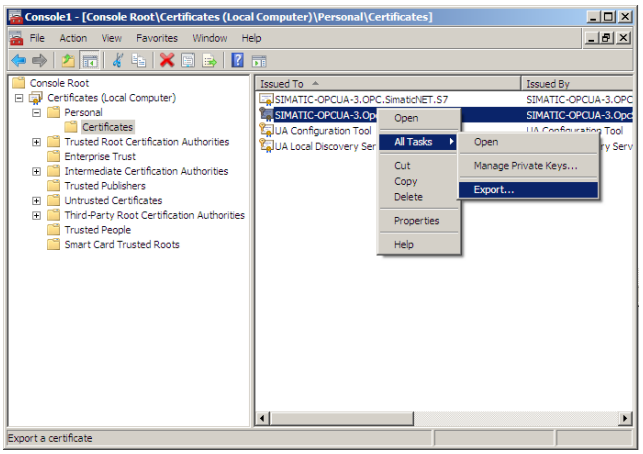
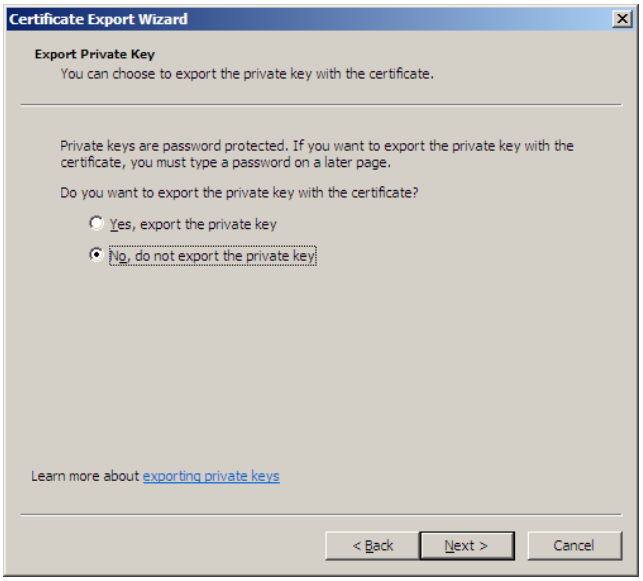
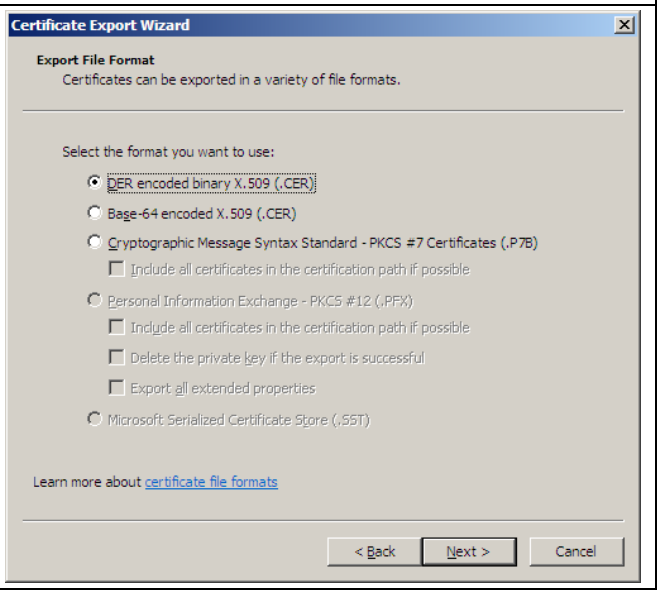
No	Action	Note
14.	Start the configuration tool: `Configuration Console` and select the “`OPC UA Certificates`” sub-item.	On the “OPC Server Station” server PC: Start→Programs→Simatic→SimaticNET→Configuration Console
15.	All public UA certificates known to the server are found here. With a right-click on a selected certificate you can accept it.	

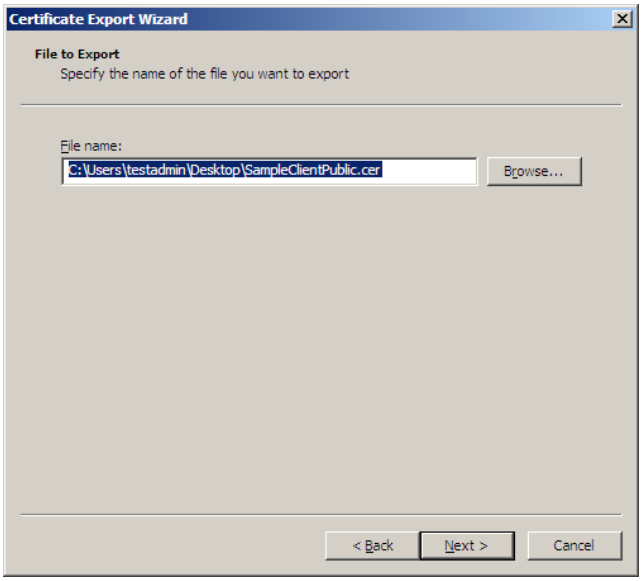
Configuration server with the SIMATIC NET CD 2008 (V7.1)

In the SIMATIC NET CD 2008 (V7.1) the trusted client certificates have to be copied manually to the certificate directory of the server. They are first exported out of the Windows certificate store from the client PC.

Table 6-5

No	Action	Note
1.	Open the Windows certificate store in the Management Console.	On the “OPC Client Station” client PC Start→ execute → “mmc” 
2.	In the file menu select “Add or Remove SnapIn” and afterwards select “Add>”.	
3.	Select the certificates for the “Computer account” of the local computer.	

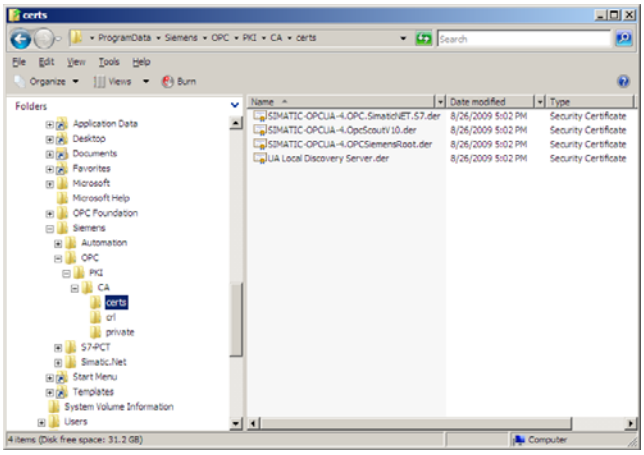
No	Action	Note
4.	Select the certificate to be exported (right-click→ All Tasks → Export…).	The Export wizard is started 
5.	The private key is NOT exported but only the “public” key	
6.	Select **DER** coding	

No	Action	Note
7.	Enter a file name and store it in the certificate file.	

Manual import of client certificate at server

In the SIMATIC NET CD 2008 (V7.1) trusted client certificates have to be copied directly to the certificate directory of the server. The certificate file from the client PC, exported from the Windows certificate store is renamed and copied on the server PC (in the certificate directory of the server).

Table 6-6

No	Action	Note
1.	Open the directory in the Windows explorer	On the “OPC Server Station” server PC
2.	Copy the file on the “OPC Server Station” server PC and change the file ending to **DER**	Example: “SampleClientPublic.der”
3.	In the certificate directory of the servers you will find all public OPC UA certificates trusted by the server. This is where the client certificate has to be copied to be accepted by the server.	The directory is located in the SIMATIC NET software data directory. C:\ProgramData\Siemens\OPC\PKI\CA\ 

6.3.3 Authentication, SecurityPolicy and MessageSecurityMode

In the SIMATIC NET CD 2008 (V7.1) as well as in the SIMATIC NET CD 2010 (V8.0) the following steps must be configured manually. As of SIMATIC NET CD 2011 (V8.1) these settings can be configured in “Set PC Station” via the user interface. The OPC UA server supports the authentication of clients during connection establishment. Two types of authentication are supported:

- Anonymous
- UserName / Password

After the installation both modes are active, to make commissioning easier. The server can be reconfigured so that anonymous logons are no longer possible. User name and password have to be indicated by the client and the server checks it against the Windows user administration. Thus, only clients which have a Windows account on the server machine can connect.

The server has two connection endpoint configurations which can be used by the clients. Each of these endpoints represents another encryption mechanism for data transmission

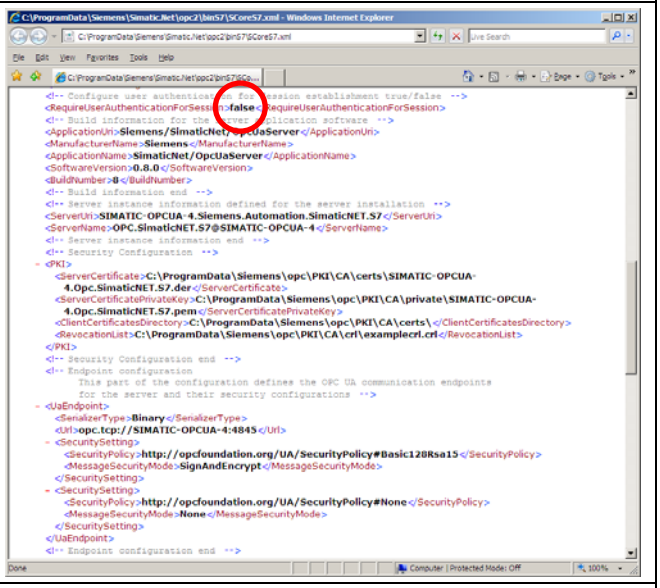
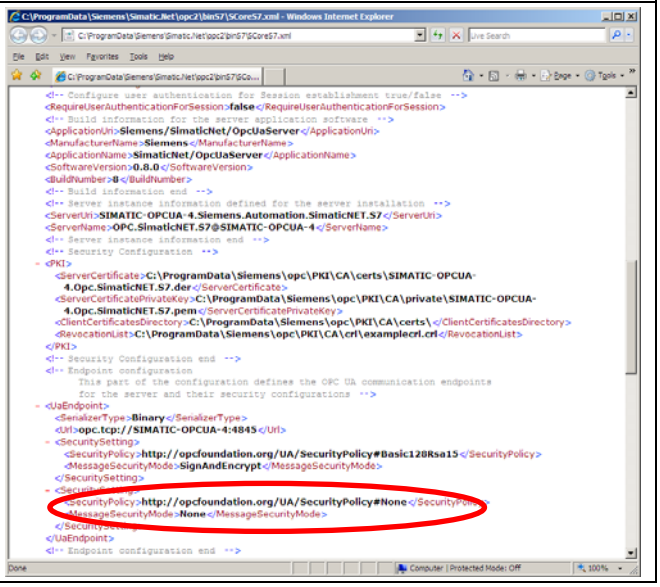
- None
- Basic128RSA15

After the installation both SecurityPolicies are active to make commissioning easier. The server can be reconfigured to only use secure encrypted connections. Thus, only clients can connect which know how to handle the Basic128RSA15 encryption.

If you want to change the security mechanisms of the OPC UA server, edit the configuration file of the OPC UA server and afterwards restart the server.

C:\ProgramData\Siemens\Simatic.Net\opc2\bins7\ SCoreS7.xml

Table 6-7

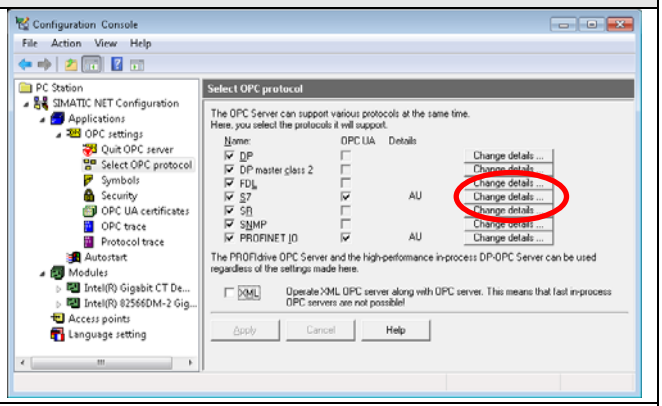
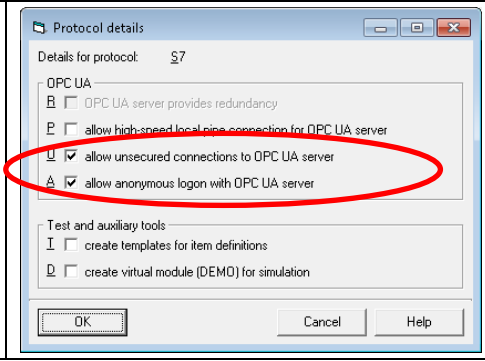
No	Action	Note
1.	Open the OPC UA "SCoreS7.xml" server configuration file in an editor (e.g. notepad)	The file is located in the SIMATIC NET software data directory. C:\ProgramData\Siemens\Simatic.Net\opc2\bins7
2.	Set the RequireUserAuthenticationForSession to "true" if each OPC UA client is to authenticate itself by username and password to be able to establish a session.	
3.	Delete the complete SecuritySetting entry for SecurityPolicy "none" and MessageSecurityMode "none" from the UAEndpoint configuration. Afterwards the server will only allow encrypted transmissions of the "Basic128RSA15" type with signature and encryption. Only clients which also support this encryption type can connect themselves (if their certificate is trusted).	

Note

In the follow-up version of the OPC UA server on the SIMATIC NET CD V8.1 these configuration steps will be possible via the "Configuration Console" configuration tool.

Table 6-8

No.	Action	Remarks
4.	Open the "Configuration Console" tool and select the "Select OPC protocol" option	In the S7 line (Change details…) the settings for the S7 OPC UA server security are made.

No.	Action	Remarks
		
5.	In the dialog of the protocol details the unsecured (unencoded) endpoint can be switched on and off.	

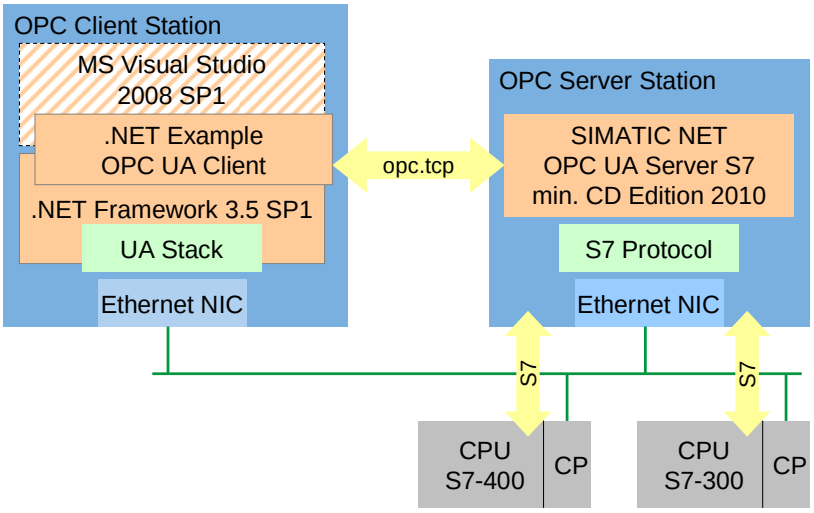
7 Installation and Commissioning

This chapter describes which hardware and software components have to be installed. It is also important to read the descriptions, manuals and any delivery information supplied with the products.

Overview

The figure below shows the hardware setup of the application as well as the necessary software components.

Figure 7-1



7.1 Hardware and Software Installation

This chapter describes which hardware and software components have to be installed. It is also important to read the descriptions, manuals and any delivery information supplied with the products.

Installation of the hardware

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

Table 7-1

No	Action	Note
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.

Installation of the standard software

For the software components, please refer to chapter 2.

Starting up the example requires the following components from the SIMATIC NET CD 2010 (V8.x):

- SIMATIC NET PC products
- SIMATIC NCM PC/S7

Note SIMATIC NCM PC/S7 needs only be installed if no STEP 7 has been installed on the PC.

The configuration tool SIMATIC STEP 7 V5.5 is only required at the PC station if the S7 controllers need 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 safety of the SIMATIC PC station and if necessary install Windows updates.

Table 7-2

No	Action	Note
1	Install the SIMATIC NET CD 2010 (V8.1) on the SIMATIC OPC server station.	
2	Install the STEP 7 V5.5 HF4 on the engineering station (e.g. laptop).	Only required for programming and configuring the connection of the S7-300/400 and for creating alarm and symbol information
3	Install the .NET Framework 3.5 SP1 on the OPC client station.	This step is only necessary when the framework has not yet been installed by any other application.
4	Install the Microsoft Visual Studio 2008 SP1 on the OPC UA client station. Install the C# development environment.	This step is only necessary if the code is verified or modified.

7.2 Application software installation

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

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

Table 7-3

No	Action	Note
1	Install the application software on the OPC UA client station. Copy the Executables as well as the Assemblies (file: “bin”) and also the source code (file: “scr”) in a directory to which you have access rights.	You only need the source code if you want to make modifications or verify individual functions.
2	Install the STEP 7 program. Copy and unzip the STEP 7 project.	If necessary, adjust the configuration of your hardware.

7.3 Commissioning the SIMATIC S7 stations

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

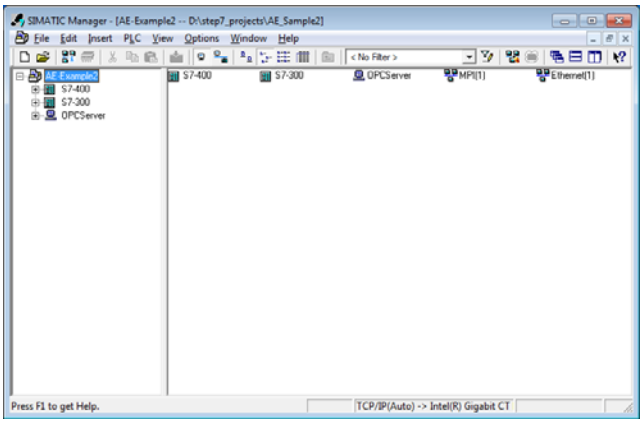
Note

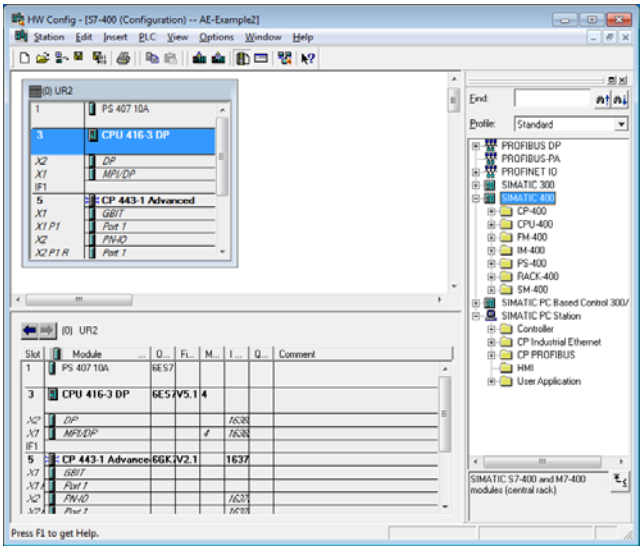
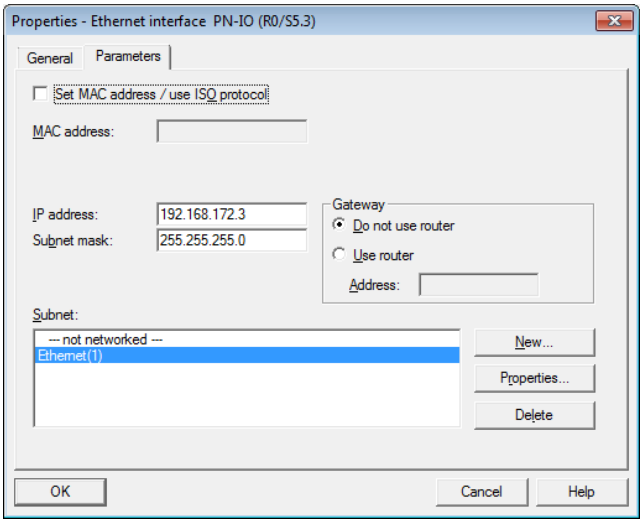
The project file delivered with this example contains the completely configured SIMATIC S7 stations according to the description in chapter 2.3. This STEP7 project can only be used without adaptation if the hardware is identical to the configuration.

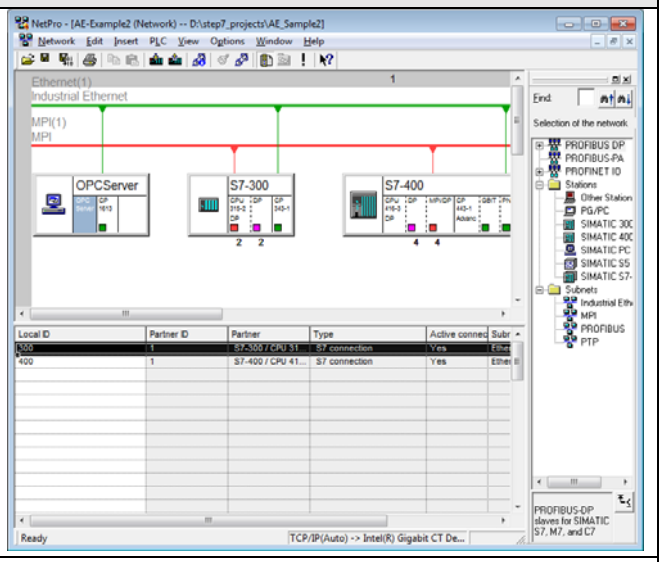
If different hardware is used, the configuration of the SIMATIC stations has to be adjusted. This is particularly necessary when the hardware releases differ or when the Ethernet addresses are not identical. The STEP 7 project included in the delivery then has to be opened and adapted accordingly. After 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 7-4

No	Action	Note
1.	Retrieving the project	Unzip the 26548467_OPC_AC_STEP7_v10.zip file
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	Note
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` dialog of the communication 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	Note
		
7.	Select the station to download the configuration. Click the  icon in the `SIMATIC Manager`	Perform this step for the S7-300 and the 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 the respective dialog 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

7.4 Commissioning the PC station

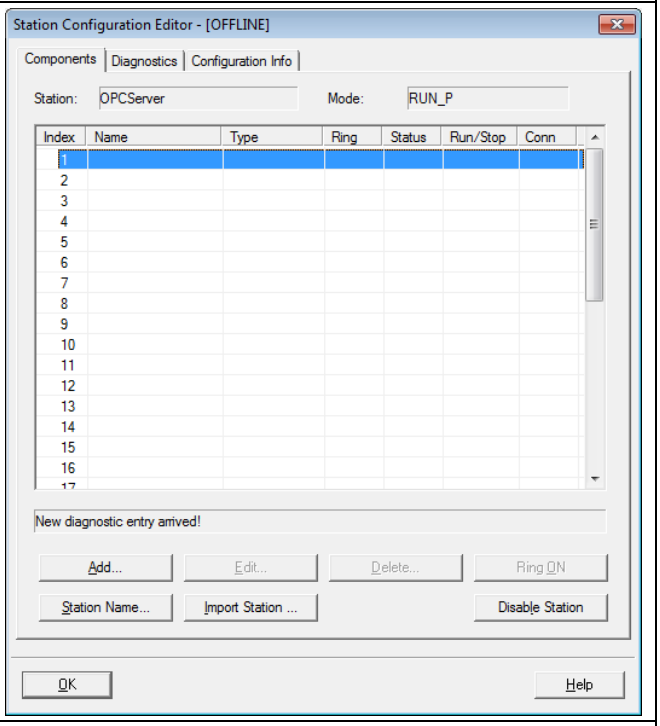
It is required that all hardware and software components have been successfully installed and cabled.

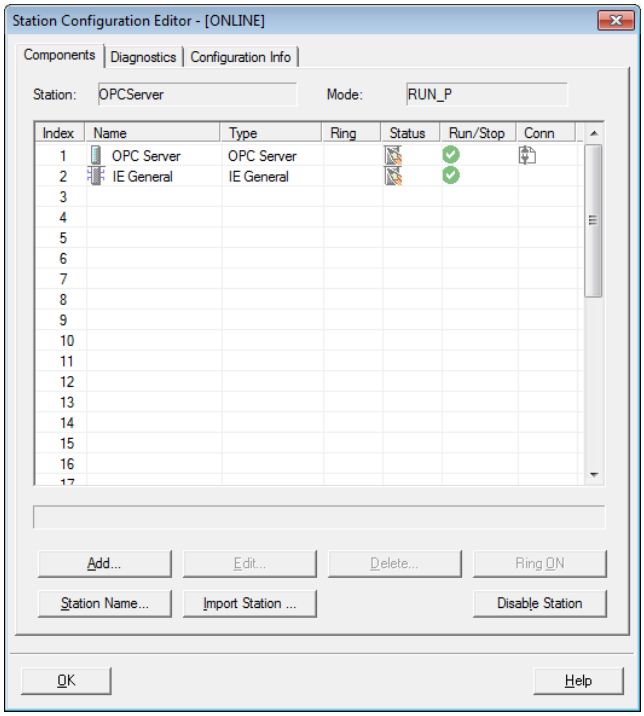
Note The project file (XDB) delivered with this example contains the completely configured PC station. This file can only be used without adjustment if the hardware is identical with the configuration.

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 has to be opened and adapted accordingly (see chapter 6.2). After saving and compiling, all configuration information is overwritten in the XDB file.

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

Table 7-5

No	Action	Note
1.	Retrieve the project.	Unzip the 26548467_OPC_AC_STEP7_v10.zip file
2.	Open the `Station Configurator` tool by double-clicking the  icon in the task bar.	
3.	Click on `Import Station`	Confirm the query with Yes
4.	Import the XDB file	The XDB file is located in the XDB subdirectory of the extracted ZIP file in the directory tree of the STEP7 project.

No	Action	Note
5.	After importing the XDB file the PC station has been configured.	

8 Operation of the Application

Here you will learn …

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

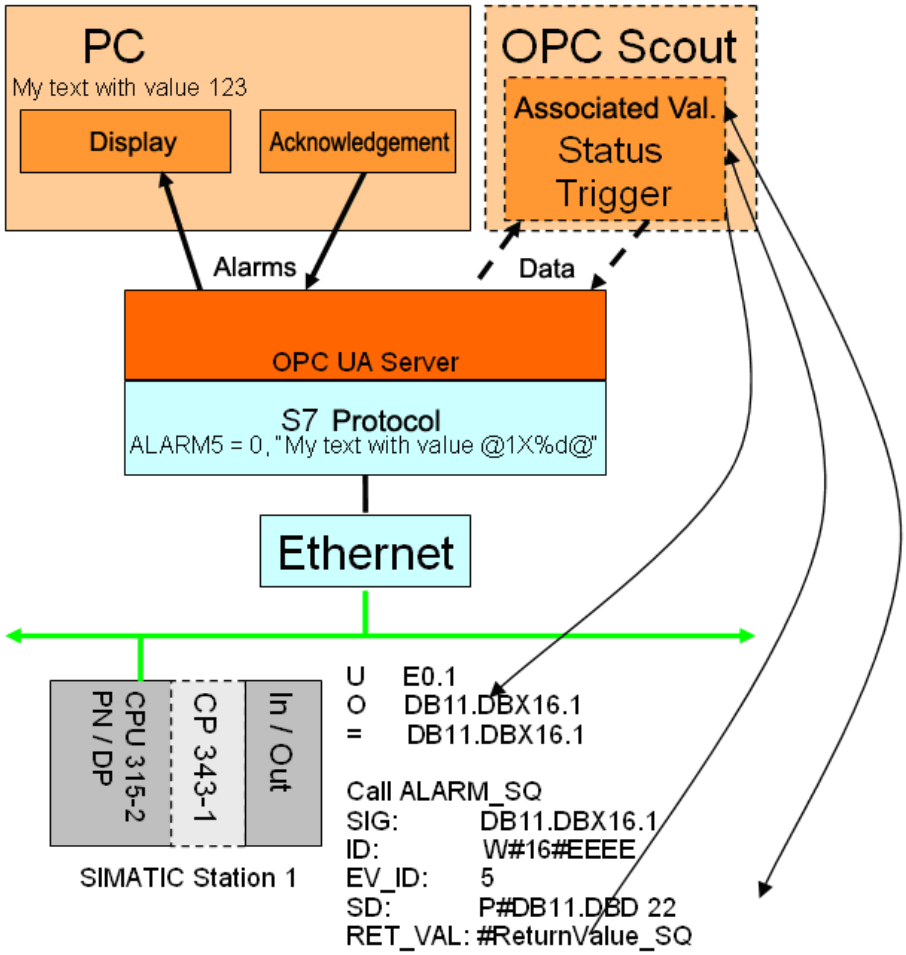
Note

To provide a simplified representation of the principle of operation, an ALARM_SQ via the FB200 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 **0.1 input** or the **DB11.DBX 16.1** data block bit changes to “1” status.

Figure 8-1



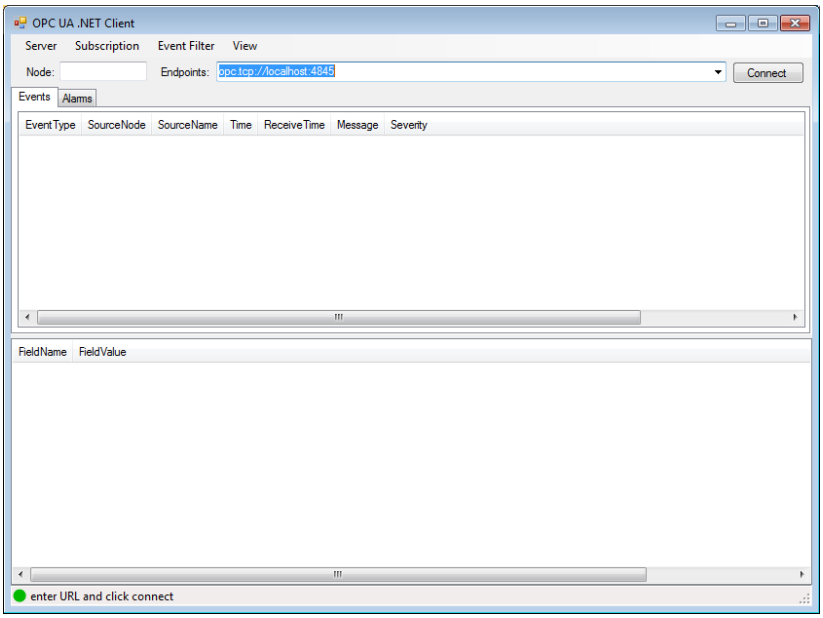
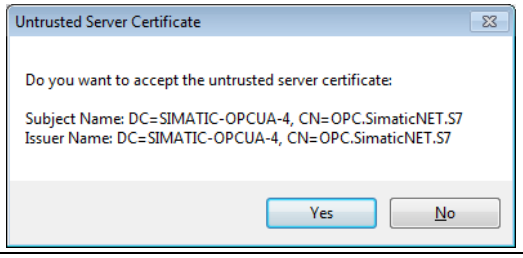
The alarm block receives a fixed ID, which is CPU wide unique and was interconnected with an associated value (here an any-pointer which points to a double-word in data block 11). The return value of the alarm block is written to a local variable.

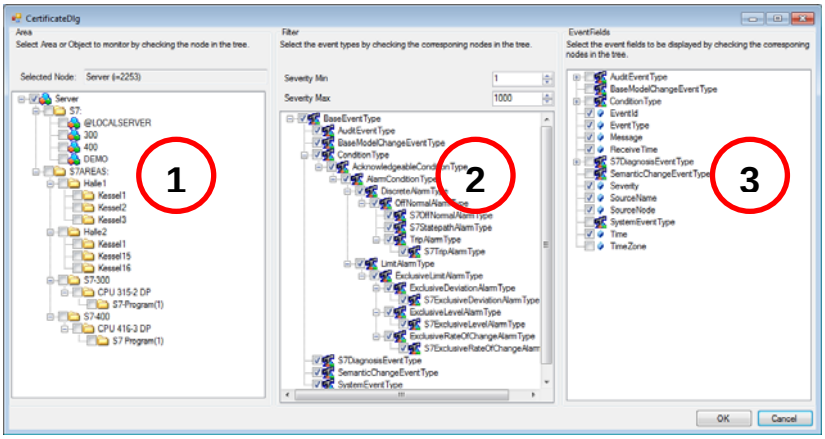
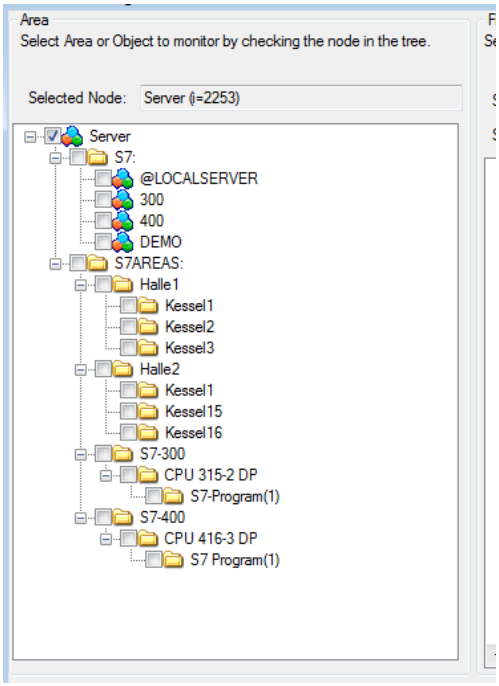
Triggering alarms

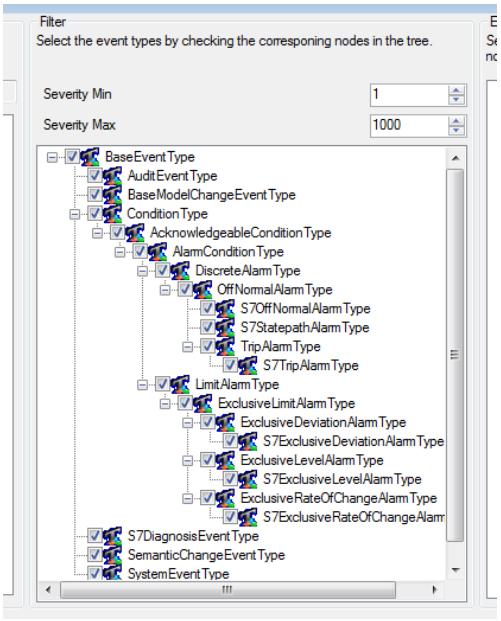
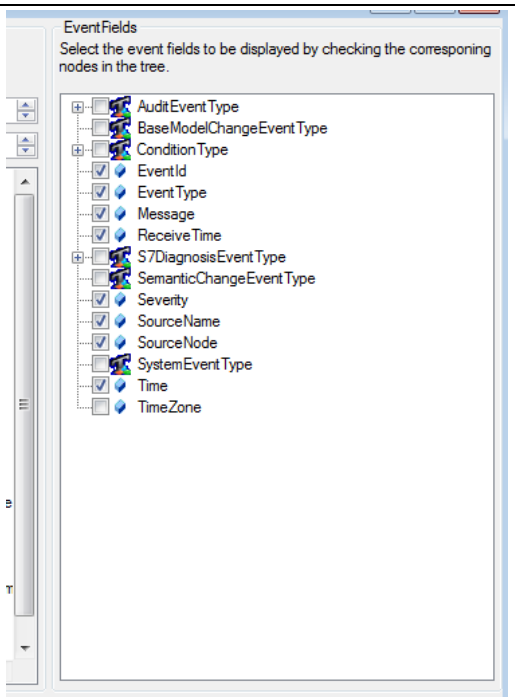
To simulate an alarm, the 16.1 bit in data block 11 can now be used in addition to the physical E0.3 input. The memory bit can now be written with OPC Scout (trigger). An alarm with ID number 5, for example, is triggered when the status changes from “0” → “1”, an additional alarm is triggered in the event of a status change from “1” → “0”. This corresponds to the “came in” and “went out” principle of status-controlled alarms.

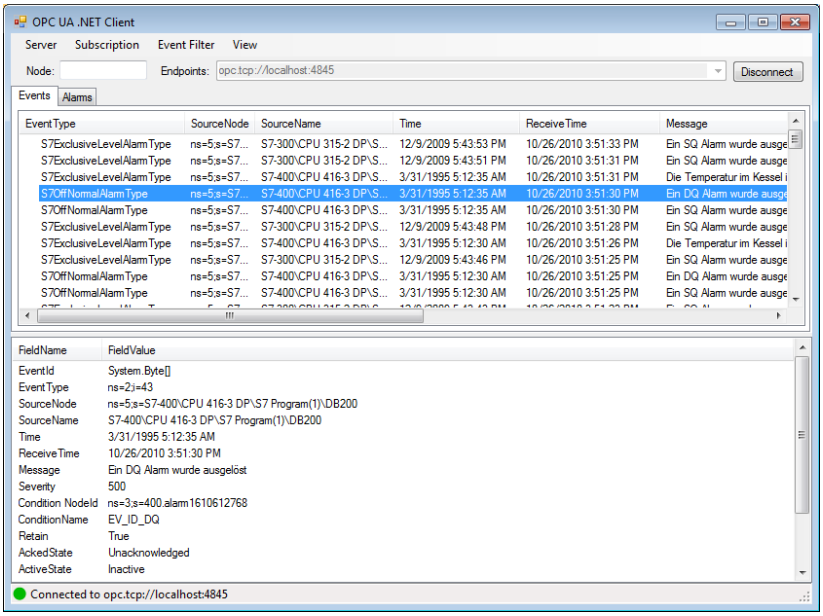
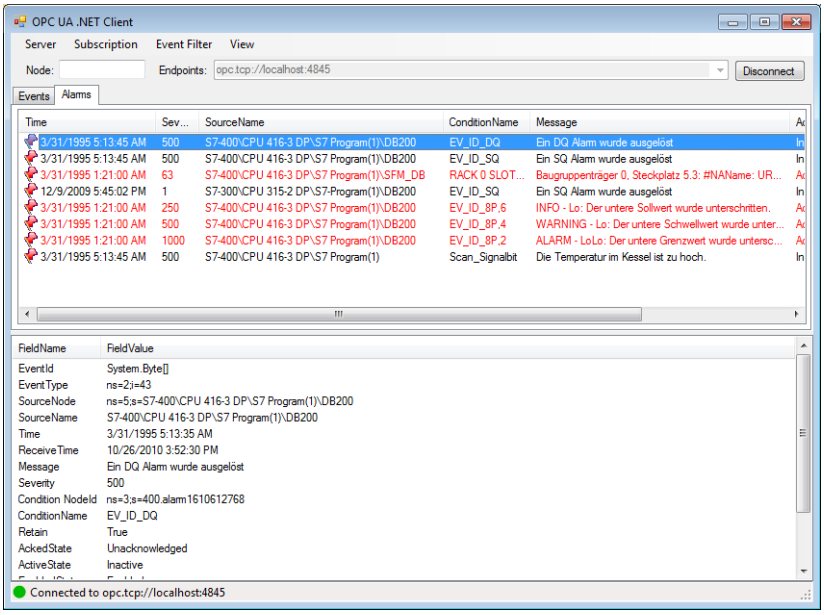
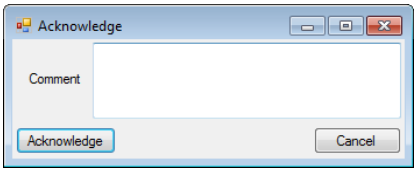
A permanent automatic alarm simulation can be switched on via bit 2.0 in data block 11 (DB11.DBX2.0) or the “SimulationAlarms.SimulationActive” symbol respectively. This bit can be either set via STEP7 with the variable table (VAT_1) or via OPC DataAccess (OPC Scout).

Table 8-1

No	Action	Note
1.	Start the OPC UA A&C Client (this example).	**Administrator rights** are required during the first start, since the client creates its own certificate and stores it in the Windows certificate store. Subsequently, the client can also be started with user rights. 
2.	Selecting the server	In the drop-down box of the user interface, the OPC UA server can be selected. If it does not appear there, the UA server is not logged on at the local Discovery server, then the URL must be entered manually (opc.tcp://localhost:4845)
3.	Connect	The Connect button is used to establish the connection to the server. After the Connect a dialog for accepting the certificate appears. 

No	Action	Note
4.	Selecting the 1) Nodes 2) Filter 3) Event fields	
5.	Selecting the nodes for which events shall be received. Root node “Server” is default to receive all events which the server triggers	

No	Action	Note
6.	Selecting the filters. A certain priority area can be filtered here, the maximal range (1-1000) is default. Furthermore, different event types can be filtered to.	
7.	Selecting the fields. The event fields required for the user interface have already been selected. Further fields can be checkmarked and the respective information be displayed in the bottom window (detailed view).	

No	Action	Note
8.	View of all events in temporal sequence. Each state change is written to a new line.	
9.	View of all pending alarms. Each alarm is displayed in one line, a state change causes a change in color.	
10.	Acknowledging an alarm via right-click.	An acknowledged alarm is represented with a green checkmark. 

9 Further Notes, Tips & Tricks, etc.

Reusability and expansion of client API

The client API is realized as an independent, reusable assembly DLL. It can be directly used in other applications. An expansion for additional OPC UA features such as method calls can be easily achieved.

Reusability of the GUI controls

The GUI elements for browsing, listing of attributes and for monitoring of variable values have been created as controls. For reusability it makes sense to store these controls in an independent assembly DLL.

Storing of nodeIds

NodeIds are made up of an identifier and the namespace index. Although the namespace index does not change as long as the OPC UA server is running, it is always possible that the index changes during a restart of the OPC UA server. Although this is not the case with the SIMATIC NET OPC UA server, a OPC UA client should nevertheless be prepared for it when the storing the nodeIds.

When storing, the index must not simply be saved but the namespace URI has to be saved. This URI remains constant, even when the index changes.

There are two strategies to save the namespace URI:

- Instead of the saving the index, the URI is saved with the identifier for the nodeID. This is the easiest variant but has the disadvantage that a great deal of redundant information is saved when the namespace URI is the same for all stored nodeIds of the server.
- The index is saved with the identifier but the appropriate namespace table with the namespace URIs is stored in parallel. This variant is more efficient. However, it requires an additional storage location for the table.

For both variants the namespace table has to be read from the server, after establishing a connection with the server, and the namespace URI has to be reimplemented in the current index.

Optimizing of the nodeIds for read and write

NodeIds may contain long texts and are therefore not suitable to be used in cyclic calls of read and write since this causes an unnecessary overhead on the network and during the processing on the server.

OPC UA provides the special services RegisterNodes and UnregisterNodes, to be able to achieve an optimization. RegisterNodes supplies a list of optimized nodeIds with numeric identifiers for a list of original nodeIds, which can be used like Handles. These nodeIds are only four bytes long on the network and can be used for very fast data access on the server.

Since the Handle is also a nodeId, it can be used in all services instead of the original nodeId. However, the optimized nodeId is only valid within the session.

If registered nodeIds are no longer needed, they should be released with UnregisterNode to release resources on the server.

10 Links & Literature

10.1 Literature

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

Table 10-1

	Topic	Title
/1/	STEP 7	Automatisieren mit STEP7 in AWL und SCL (Automating with STEP7 in STL and SCL) Hans Berger Publicis MCD Verlag ISBN 3-89578-113-4
/2/	OPC UA	OPC Unified Archticture Mahnke, Leitner, Damm Springer Verlag ISBN 978-3-540-68898-3

10.2 Internet links

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

Table 10-2

	Topic	Title
\1\	Quantity framework on the message number procedure	In product support under the entry ID: 2654846 http://support.automation.siemens.com/WW/view/en/26548467
\2\	Siemens I IA/DT Customer Support	http://support.automation.siemens.com
\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 Unified Architecture	Specification on the OPC Foundation website for download for OPC members www.opcfoundation.org
\5\	SIMATIC NET *Commissioning SIMATIC NET PCStations – Instruction and quick start for SIMATIC NCM PC / STEP 7 from Version V5.2*	Description of or information on: • General information on the PC tools. • Functions of NCM PC. Installed by SIMATIC NET, see: Start → Simatic → Documentation → English. In product support under the entry ID: 13542666 http://support.automation.siemens.com/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. In product support under the entry ID: 2044387 http://support.automation.siemens.com/WW/view/en/2044387

11 History

Table 11-1

Version	Date	Modification
V1.0	12/2010	First issue