Domain Modelling & Formal Specification of a Generic OPC Alarm Collector

Master's thesis by:

Nicolai Henriksen (s992069)
nicolai.h@webspeed.dk

Supervisor: Mads Nyborg

Department of Informatics and Mathematical Modelling
Technical University of Denmark
September 2005 - Lyngby - Denmark

Abstract

An alarm generally indicates that something has gone wrong and needs immediate attention. Such alarms are very important and should be handled carefully. This paper documents the development of an OPC[1] Alarm Collector using UML[2] and OCL[3] to model the domain and the RAISE[4] Specification Language to formally describe critical areas of the domain. The result of the developments include: a generic framework for creating Windows Services (specifically designed for Zonith A/S), a generic model for collecting alarms from various OPC Servers, as well as a new product in Zonith A/S's product line.

Keywords: RAISE, OCL, UML, OPC, C#[5], .NET[6], XML[7]/SOAP[8]

[1] O(bject Linking and Embedding) for Process Control
[2] Unified Modelling Language
[3] Object Constraint Language
[4] Rigorous Approach to Industrial Software Engineering
[5] A programming language supported by the .NET Framework
[6] A Framework developed by Microsoft primarily intended for developing web applications
[7] eXtended Markup Language
[8] Simple Object Access Protocol

Acknowledgements

This section is dedicated to thanking the many people who have contributed to the completion of my Master's thesis.

I would like to thank the following people and companies:

Signalix for supplying equipment on which the the product of this Master's thesis could be tested.
Andrzej Swietek from Signalix, for many insightful conversations about OPC and it's inner workings.
Rik S. Rose from Schneider Electric for guiding me in the right direction while searching for useful information regarding the OPC standard, and means of establishing connections. I would also like to thank Rik for taking the time to have me visit him and test the final product with Schneider Electric's PLC's and OPC Servers.
Zonith A/S for the opportunity to do a project for them as my Master's thesis, and for supplying a great workstation in a comfortable environment during the development.
Klaus Bom, Henrik Buch, Kristian Jensen and Kristian Stiesmark from Zonith A/S for their many good advices and for many insightful conversations regarding the design of the product.
Mads Nyborg, my supervisor at DTU, for the good discussions and pieces of advice along the way.

Last but not least I would like to thank my family and friends for being so understanding the past 7 months where I haven't spent as much time with them as I would have liked to.

Contents

List of Tables

List of Figures

1 Introduction

This report constitutes the Master's thesis of Nicolai Henriksen (s992069) written in conjunction with Zonith A/S.

The thesis is split into two parts: A new product, OPC Listener, in Zonith A/S's product line, and a report documenting the work carried out, as well as integrating the concepts of Domain Modelling and Formal Specification. OPC Listener is the term used throughout this report to represent the Alarm Collector. The reason for this name, is that Zonith A/S already have a series of alarm collectors for other types of networks, and these are all called listeners.

OPC Listener is a product which enables Zonith A/S's existing Alarm Control System (ACS) to collect alarms from any type of automation network which is OPC compliant. The product itself consists of a Windows Service which collects the alarms, and a graphical configuration tool allowing a user to set up which alarms should be collected, etc. The product is to be written in C# for the .NET Framework.

There are several reasons why this project was chosen. First of all I had not attended any courses at DTU focusing on the .NET Framework and therefore felt that this was a great opportunity to get to know the Framework. Secondly, the project which Zonith A/S presented to me had several aspects where the concepts of Domain Modelling and Formal Specification where applicable. Other reasons for choosing this project were: the clean cut between practical implementation and theoretical application, as well as the central geographic location of Zonith A/S minimizing time spent on travelling.

The report is structured into the following sections:

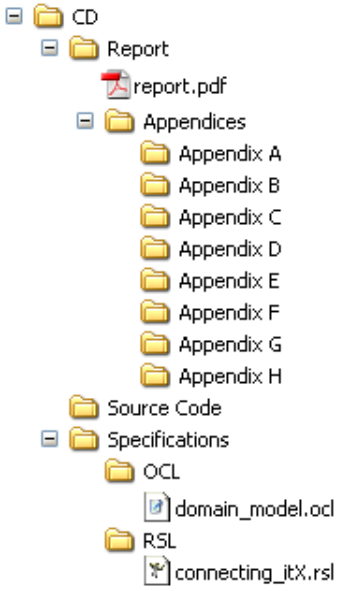
Objectives	Clarifying the project and specifying the goals
Theory	Introduces the theory which will be applied
Method	Describes the method followed during the project
Analysis	Analysis of what the product must be able to do
Design	A design of the product which has the required functionality
Implementation	The implementation of the design
Test	Describes the test procedure and mentions other possibilities
Discussion	Discusses the good/bad things and possible extensions
Conclusion	The conclusion summarizing what has been done
References	References used during the project
Appendices	Appendices such as user's guide, test cases, etc.

1.1 Legend

The font used in this line is the font used throughout the report for normal text. Some words that need special attention will in some cases be written in *italics*. OCL and C# code will be written in `this font`. RSL code is incorporated using a module specifically designed for incorporating RSL code into LaTeX documents, and therefore keywords will be highlighted and the code will be "pretty-printed".

1.2 CD Contents

The contents of the enclosed CD is as follows:



2 Objectives

This project has two main goals: 1) To add a new product to Zonith A/S's product line increasing their spectrum of possible customers, and 2) to write a thesis covering the work done as well as incorporating theory from the fields of Domain Modelling and Formal Specification.

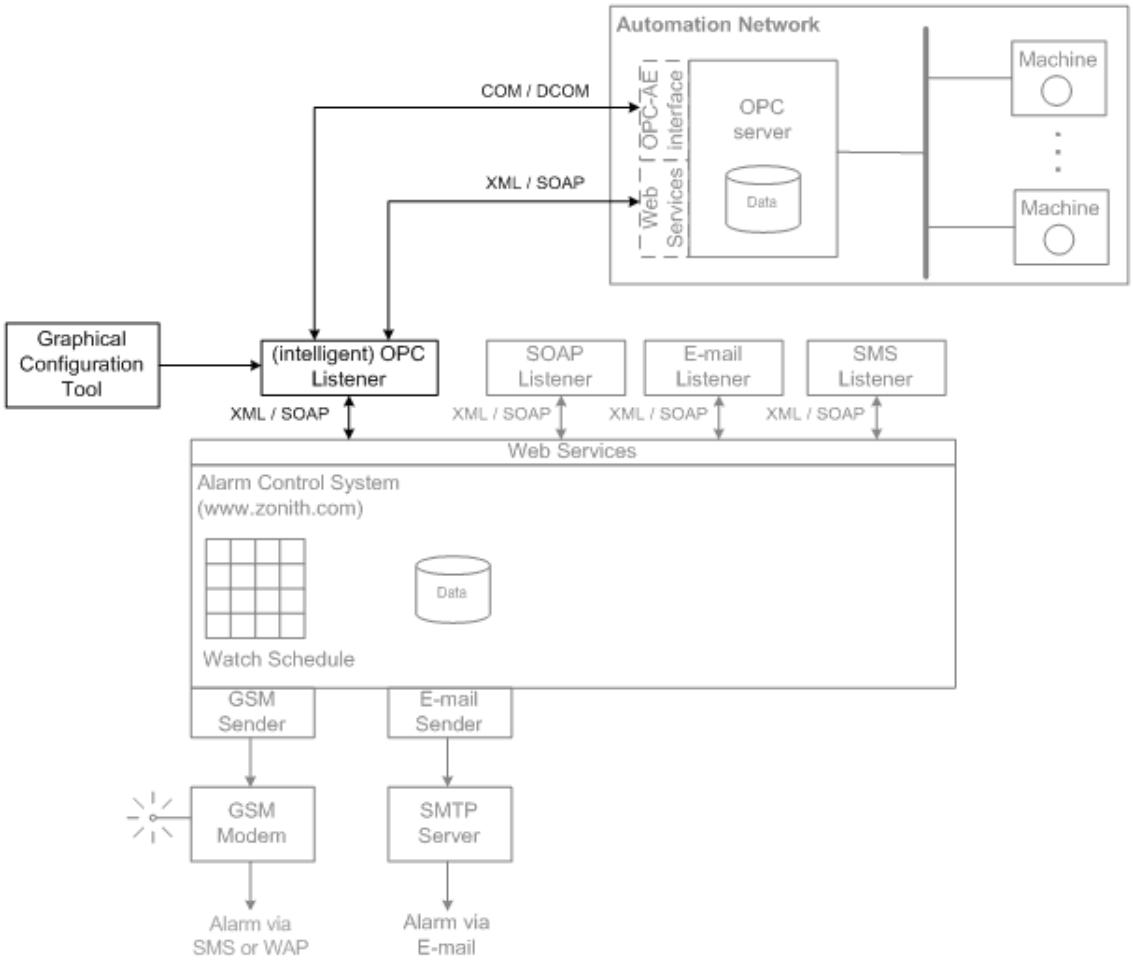


Figure 1: System diagram

The product which has to be created is an OPC Listener. The purpose of the product is to survey a number of OPC Servers and collect any alarms that may occur. These alarms are then to be delivered to Zonith A/S's existing product Alarm Control System (hereafter ACS). Access to OPC Servers is done using a standard set of interfaces defined by The OPC Foun-

dation [6]. These interfaces are commonly accessed through COM/DCOM objects/interfaces which are limited to Microsoft platforms. Future releases from the OPC Foundation will include an XML/SOAP based solution allowing access to OPC Servers through Web Services. Communication with the ACS is done using XML/SOAP, by sending a message to a Web Service. An overview of the complete system is shown in Figure 1.

For a broader scope diagram covering the ACS in general, see Appendix G.

As can be seen from the diagram in Figure 1, the OPC Listener has a graphical configuration tool. The purpose of this tool is to make it easy for any user to select which OPC Servers to survey as well as which alarms to collect.

The thesis has to cover the analysis, design, implementation and test of the product described above. Furthermore it must describe and incorporate theory from the fields of Domain Modelling and Formal Specification described in detail in the next section. The applied theory will formally describe selected parts of the project, and thus aid in constructing a more complete and functional system.

In order to ease the installation and use of the OPC Listener for the end users, a *Users Guide* must also be written. This document can be found in Appendix A.

3 Theory

This section introduces the theory which is applied in later sections and tries to give the reader a basic understanding of fundamental ideas behind the theory.

3.1 OPC

This subsection is introduced to give the reader an overview of what OPC is. OPC is an acronym for "OLE for Process Control", where OLE is an acronym for the Microsoft Technology known as "Object Linking and Embedding". OPC has over the years become the defacto standard for reading/writing values from/to automation networks. To make an automation network OPC compliant, it is required that a computer, running an OPC Server, is added to the network. The server has an internal database which keeps track of all the alarms, events, state changes, etc. which are generated by the nodes attached in the network. An OPC Server must implement a set of interfaces defined by the OPC Foundation [6], thus allowing OPC Clients to access the data stored in the database. A simple example could be a laundromat controlled and surveyed remotely (depicted as the Automation Network in Figure 1). In this example all the washing machines would be nodes in the network, and an OPC Server would collect/receive data from each of them. There are numerous proprietary protocols for the communication between the nodes and the server, but since the OPC Server is OPC compliant and thus implements a set of standard interfaces, clients do not have to worry about this communication at all. In this example OPC Client would then be able to obtain the states of the washing machines, start/stop the machines, be notified if errors occur, etc. simply by accessing the OPC Server using the standard interfaces.

3.2 Domain Modelling

Domain Modelling is basically a formal model of the real world. The aim of a domain model is to extract which entities are important, what relations exists between entities, as well as which constraints are associated with them. There exists a number of accepted ways of representing a domain model, however in this report domain models will always be represented using Unified Modelling Language (UML) in association with various constraints written in Object Constraint Language (OCL).

3.2.1 Example

Consider the domain of soccer clubs. A very simplified model could have the following entities:

- Club
- President
- Coach
- Player
- Contract

One could also add some constraints to the model, as for example:

- Every club has to have one and only one President
- Every club has to have at least 11 players
- A player must be 16 years old before he/she can have a contract
- A player must have a contract in order to be sold

A visual representation of this model, shown in UML, can be seen in Figure 2.
The constraints listed above can be expressed in OCL as follows:

```
context Club inv:
    self.president->asSet()->size() = 1
    self.players->size() >= 11

context Contract inv:
    player.age >= 18

context Club::sell(p: Player)
    pre: p.contract->asSet->size() = 1
    pre: self.contracts->includes(p.contract)
    post: self.contracts->excluding(p.contract)
    post: self.contracts->self.contracts@pre - 1
```

Domain models are used in various aspects of Software Engineering, often helping developers gain a better understanding of a clients world (domain). Creating a good domain model before beginning the development process will in most cases result in a better and more solid solution in the end.

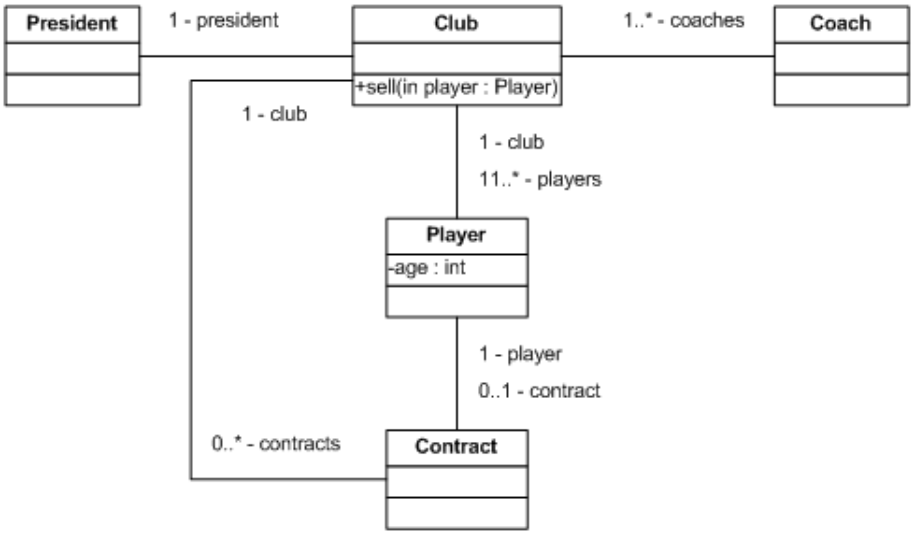


Figure 2: Domain Model of the domain of soccer clubs (example)

3.3 Formal Specification

Formal Specification in many ways is like Domain Modelling. It is concerned with specifying the contents of a system such as data types, relations and functions. The first specification created is very much like the domain model, although it is based on a somewhat mathematical representation. The following iterations makes the specification more and more specific, and at some point ends up in a strict mathematical specification which can be machine-translated into code (C++, Java, etc.) obeying the relations and constraints defined in the specification.

Extensive use of pre- and post-conditions on functions, as well as other forms of axioms, allows the specification to precisely indicate what is acceptable and what is not, thus in many cases eliminating potential design/implementation errors. The use of an iterative approach allows the designer to easily backtrack and correct any mistakes/misunderstandings.

Formal specifications are represented in various ways, however in this report the specifications will all be presented in the RAISE Specification Language [4]. An example of a RAISE specification (in an early iteration) of the domain of soccer clubs can be seen below:

scheme Soccer =
  class
    type

```
        President,
        Coach,
        Contract,
        Player = Int,
        Contracts = Player -m-> Contract,
        Club =
            President × Coach-set × Player-set × Contracts

    value
        sell : Club × Player → Club

    axiom
        ∀ C : Club, P : Player •
            sell(C, P) ≡
                let (president, coaches, players, contracts) = C in
                    if (P ∈ dom contracts)
                    then
                        (president, coaches, players, contracts \ {P})
                    else (president, coaches, players, contracts)
                    end
                end
end
```

Formal specifications are used in a variety of areas. In software engineering it is often used to model areas with very stringent policies with regards to security, loss of data, etc.

3.4 HSQL

HSQL is an acronym for Hypersonic SQL, where SQL is an acronym for Structured Query Language. HSQL started as a personal project for a man named Thomas Mueller, but fairly quickly evolved into an Open Source project which it still remains. HSQL was originally written in Java for Java applications, but has since then been ported to other languages, C# included. It is a DBMS which supports a rich subset of ANSI-92 SQL (BNF tree format) plus SQL 99 and 2003 enhancements. The Java version of HSQL is currently being used as the database engine in OpenOffice.org 2.0.

3.5 WSDL

WSDL is an acronym for Web Service Definition Language. It defines an XML grammar used to describe network services abstractly in order for it to be applicable for most systems. Basically a WSDL file defines a number of methods which a given service provides, as well as which arguments the methods take, and what the return values are. In other words it is a schema indicating what a client can expect from the given Web Service defined in the WSDL file.

4 Method and Planning

The method found most appropriate for this project is the Rational Unified Process (RUP) [3]. RUP is an iterative approach based on Use Cases and focusing on risks at an early stage in the process to identify and deal with them in an orderly manner before it is too late. The reason for following such a process in primarily to enable a company to handle projects in a predefined and structured manner, gaining experience with each project and thus continuously increasing the efficiency and productivity of the organization. Although the process is primarily aimed at organizations dealing with software development, previous experiences have taught me that following a structured process even for a one-man project will result in a better solution in the end.

Three important artifacts are included in this project. These are: A timetable, a risk analysis, and a list of requirements. These artifacts are pre-development artifacts meaning their initial version (first iteration) should be created before any development takes place. The following subsections will describe each artifact in detail.

4.1 Timetable

In order to make the best use of the time available, a timetable was created to layout the tasks at hand, and to estimate the time required to complete them. Since the project is a time limited project, some constraints were placed on the duration of each task in order to fit everything into the given time period. With all the tasks in place within the given period, the next step is to create a number of milestones. A milestone is a point where a tangible sub-goal has been reached, commonly where an artifact has been created, or some development iteration finished.

The timetable for this project is created using Microsoft Project, and updated continuously throughout the course of the project. The end-result, how the timetable looked at the end of the project, can be seen in Appendix B.

4.2 Risk Analysis

A risk analysis is often created in order to identify risks which a project is facing and to define procedures which will minimize, eliminate, and/or deal with these risks. The risks are physical as well as technical and personal problems. In other words, all problems that

#	Risk	Procedure	S	F	P
1	Focusing more on the practical problem than the thesis itself	Create a specific date in the timetable where development is stopped, and focus is placed only on the report.	5	2	10
2	Server breakdown	Although local backups are performed on a daily basis, a breakdown of the server could happen. To always have a fairly recent backup, weekly backups will be stored on a remote server accessible through ftp and on a CD-ROM.	5	1	5
3	Corruptive changes to source code	Using a versioning systems, SVN (an improvement to CVS), always allows for easy access to previous versions of source code.	3	3	9
4	Illness	Setting up a VPN connection along with Remote Desktop access allows me to work on the project from my own computer at home, thus being able to make progress even in case of illness.	3	3	9
5	Job openings	Making a decision NOT to take any job no matter how tempting until the thesis is finished.	4	2	8
6	Vacation	Allow space in the timetable for 2 weeks of vacation thereby postponing some of the milestones a little bit.	1	2	2

Table 1: Risk Analysis

in some way could obstruct the course of the project. There are many ways to create such a risk analysis. For this project a list of risks is created along with a means of dealing with each of them. Furthermore the risks are given a priority (P) based on their severity (S) and frequency (F), thus the priority of a given risk can be stated as follows:

$$P = S \times F$$

Severity and frequency are numbers ranging from 1 to 5 where 1 is the least severe/frequent and 5 is the most. Table 1 shows the risk analysis associated with this project. The risk analysis presented here does not include risks which the product itself is facing, merely risks which the thesis (project) is facing.

4.3 Requirements

Traditionally in software engineering a developer, in association with the client, develops a list of functional and non-functional requirements. However in this project there is no specific client in mind, and therefore the requirements will not be as specific as if a particular customer demanded a product fulfilling a need. The requirements for this project, listed in Table 2, are requirements that surfaced while getting to know the world of OPC through playing with numerous OPC Clients and Servers. Each requirement is numbered, and given a priority indicating the importance of its implementation. Priorities range from 1 to 5 where 1 is the least important, and 5 is the most.

#	**Non-functional Requirement**	**Priority**
1	The OPC Listener must be executed as a Windows Service	5
2	The OPC Listener must comply with OPC Foundation standards in order to function with any Standard OPC Compliant OPC Server	5
3	The OPC Listener must be able to run on Windows 2000, Windows XP, and more importantly Windows Server 2003	5
4	The OPC Listener must have useful logging features	4
5	The graphical part of the Configuration Tool must be designed from an end-user perspective	3
6	The OPC Listener must store all alarms/events locally until the alarm/event, with certainty, is received by the ACS.	5
7	The OPC Listener must be easy to install (an Installer should be created)	2
8	The Configuration Tool must have access to starting/stopping the OPC Listener (Windows Service)	1
9	The OPC Listener must inspect certain alarms/events, namely DA events, to verify that the event received indeed is an alarm which concerns the ACS before transmitting it	4
#	**Functional Requirement**	**Priority**
1	Alarms intercepted by the OPC Listener must be persisted locally until the ACS, with certainty, has received the alarm	5
2	The local database of alarms must be a single instance (singleton)	4
3	Alarms must be distinguishable. i.e. they must have a unique id.	5
4	Connections to OPC servers should be reused if possible.	3
5	There must be an Event Handler for each subscription created	5
6	The OPC Listener should continuously try to send locally stored alarms to the ACS	3

Table 2: Requirements

5 Analysis

This will analyze what the product must be able to do in order to meet the requirements, as well as identify critical areas which need special attention.

5.1 Application Scope

Identifying what an application must be able to do is critical before starting the development. In this project, the complete scope covers: alarm/event generation, capturing of alarms/events, storing alarms/events, acting appropriately according to alarms/events. In modelling the domain, the complete scope will be modelled, however the areas which are outside of the project, such as ACS and Automation Networks, will not be described in great detail. The design and implementation sections to follow will focus only on the part of the model which is relevant for the OPC Listener project.

The Use Case Model shown in Figure 3 shows the different actions which can take place in the system, as well as who can perform them.

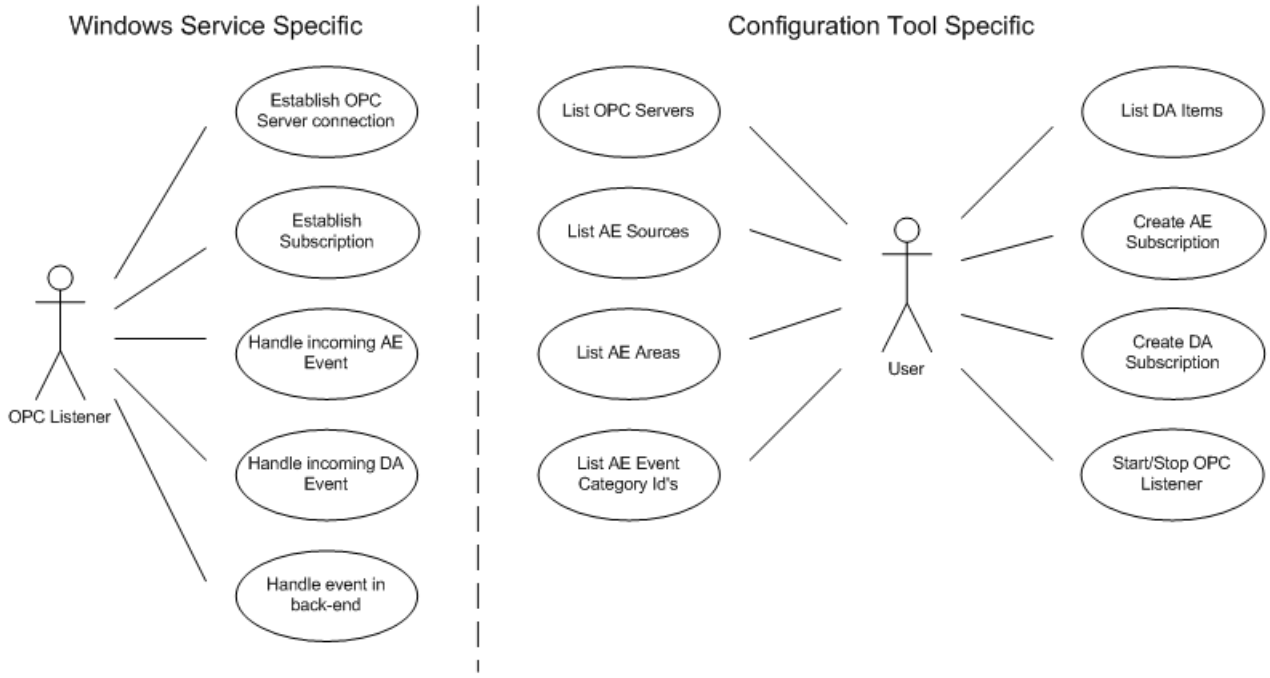


Figure 3: Use Case Model of OPC Listener

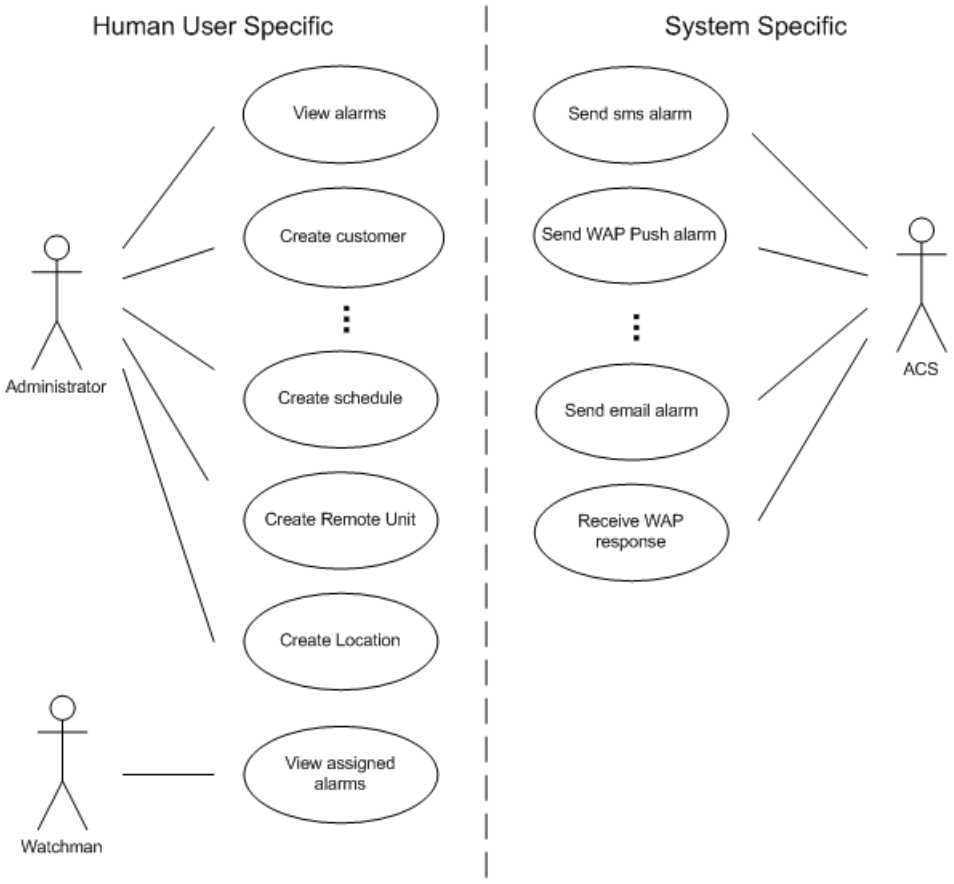


Figure 4: Use Case Model of ACS

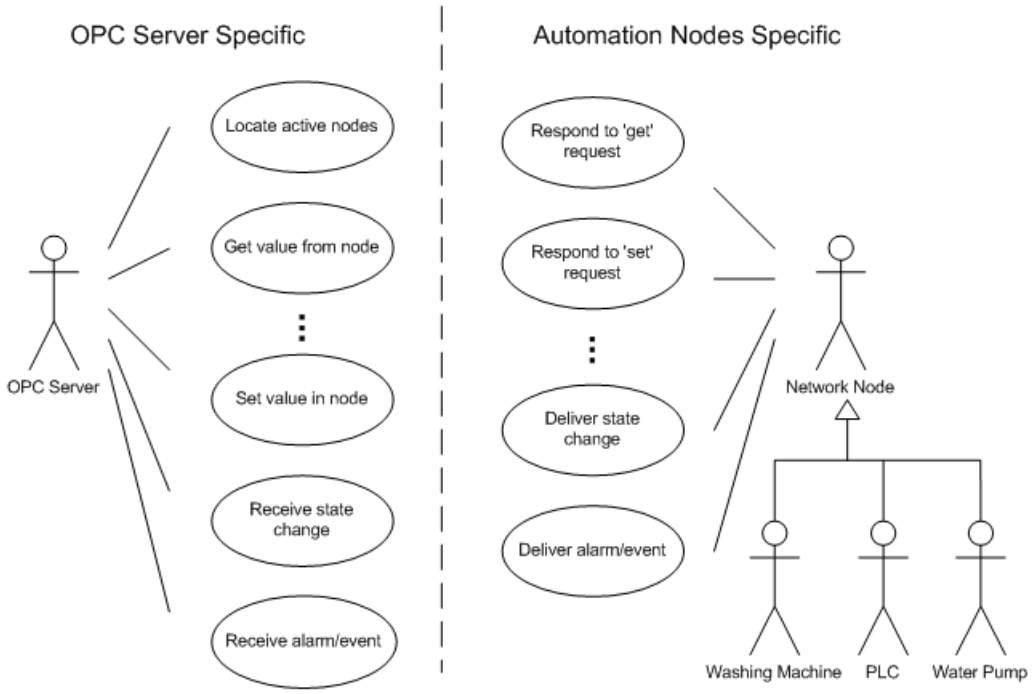


Figure 5: Use Case Model of an Automation Network

The following subsections will describe the overall system as well as go into details with each Use Case of the OPC Listener. Futhermore they will briefly comment on the Use Cases associated with the ACS as well as Automation Networks.

5.2 System Description

5.2.1 System Diagram

The system diagram shown in Figure 6 gives an overview of how the different systems are connected. Furthermore it shows that alarms/events are treated equally using an interface. The diagram could also be considered as a traditional domain model, since the domain model introduced later in this section will be more specific that domain models generally are. Alarms/events will naturally occur in the machines connected to the OPC Server. Once this happens, the OPC Server notifies the OPC Listener, via the OPC interface, that something has changed state/value, and in case it is considered an alarm, it is transmitted to the ACS via the XML/SAOP interface from where a watchman is alerted that an alarm/event

occurred which he/she needs to take care of.

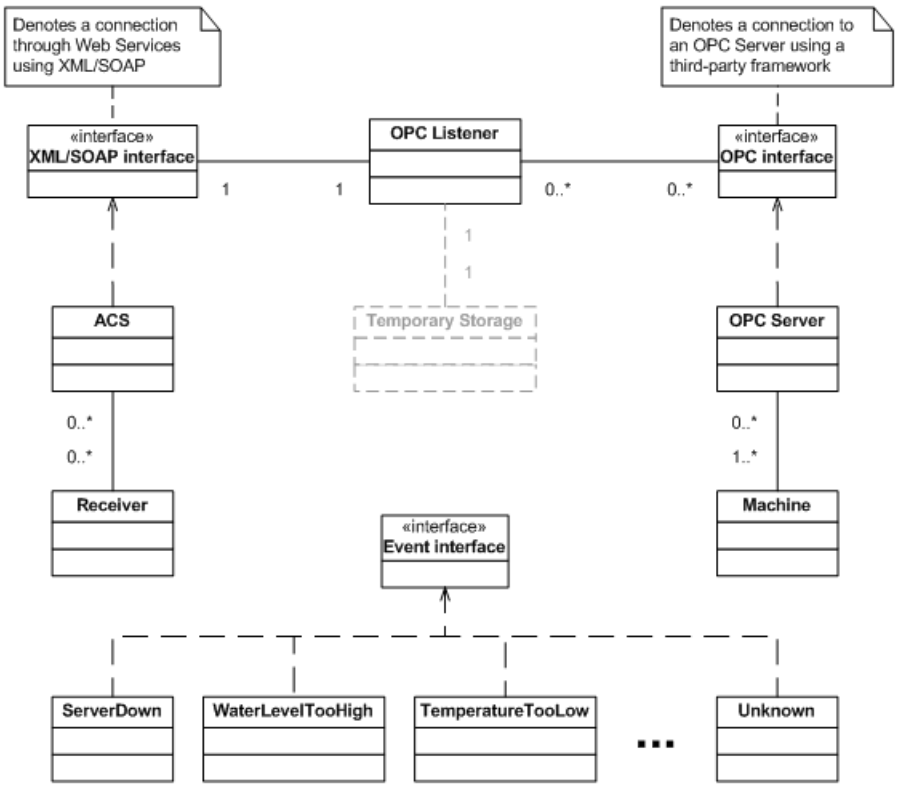


Figure 6: System Diagram

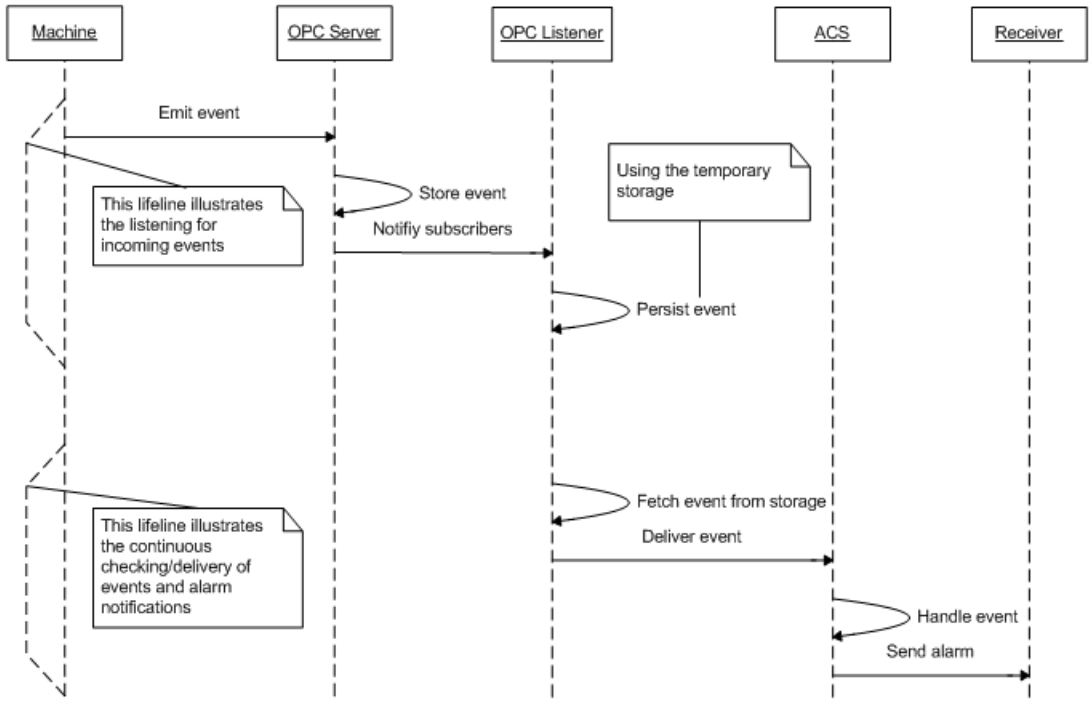


Figure 7: System Sequence Diagram

5.2.2 System Sequence Diagram

The System Sequence Diagram shown in Figure 7 shows the basic flow of information in the system. An alarm occurs in a machine in some OPC compliant automation network, the alarm/event is delivered to the OPC Server which stores it and notifies all appropriate subscribers of the given change of state. The OPC Listener persists the event locally using a temporary storage, before it is delivered to the ACS from where a receiver is notified.

5.3 Use Cases

This section will briefly describe the Use Cases associated with the ACS and Automation Networks in general before going into details with the Use Cases associated with the OPC Listener. Some of the Use Cases for the OPC Listener refer to button clicks and so on which are associated with the Graphical Configuration Tool. In order for the reader to obtain a better understanding, screenshots of the OPC Listener Configuration Tool have been included in Appendix C. Use Cases referring to these screenshots will have explicit

references.

5.3.1 ACS Use Cases

These Use Cases are divided into two groups: Human User Specific, and System Specific. The Human User Specific Use Cases are concerned with actions which users, who log into the system, can perform, and the System Specific Use Cases are concerned with the actions which the system automatically performs on incoming alarms/events. ACS is a complex system, developed over a period of 2 years, so a complete analysis of the system is not relevant here, a short introduction will do just fine. ACS contains an internal database storing information such as user accounts, customers, locations, remote units, a schedule for who is on-call (duty) at which time, as well as a series of policies of what to do in the event of an incoming alarm/event. Basically ACS is used to create a number of customers where alarms/events should be surveyed. Each customer may have an arbitrary number of locations where errors may, occur. For each of these locations, a remote unit is defined and given a unique ID. The remote unit is a locations entry point to the ACS, thus the remote unit ID is considered as the sender of an incoming alarm/event. Once an alarm is detected at a location, the remote unit is notified and either a SOAP/XML-, GSM-, e-mail- or an OPC message is sent to one of the ACS listeners which ensures the alarm/event is delivered to the ACS where appropriate action is taken. The Use Cases shown in Figure 4 will not be described in greater detail, as they are only included to give the reader a basic understanding of the ACS which the OPC Listener communicates with.

5.3.2 Automation Network Use Cases

Automation Networks come in various sizes and shapes, and it is therefore difficult to identify a set of Use Cases which describe the actions that take place. The Use Case Model shown in Figure 5 is therefore focused on Automation Networks from an OPC point-of-view. An OPC compliant Automation Network basically consists of one or more OPC Servers connected to a series of nodes (machines). The servers can continuously keep track of which nodes are connected and what states they are in. Furthermore the servers have the ability to read/write values directly to the connected nodes. The nodes themselves are basically only able to respond to read/write requests, as well as deliver information about state changes or alarms/events to the OPC Server. The Use Cases shown in Figure 5 will not be described in greater detail, as they are only included to give the reader a basic understanding of the type of networks the OPC Listener will service.

5.3.3 Use Case 1: List OPC Servers

Primary Actor: User
Stakeholders and Interests:
• User: Wants to list all the OPC Servers installed on a given machine identified by an IP address or a DNS name.
Preconditions: OPC Listener along with its required components are installed on computer which User is using.
Postconditions: A list of available OPC Servers for a given machine is presented to User.
Screenshots: Screenshots 1, 2 and 3

Basic Flow:
1. Selects OPC Configuration from the start menu.
2. Clicks on "Add Server"
3. Clicks on "Select..." next to the OPC Server Name
4. Enters the IP address or DNS name of the wanted computer and clicks "List Servers"

Frequency of occurrence: Rare. Only when alarms/events are to be added/removed from the surveillance system.

5.3.4 Use Case 2: List AE Sources

Primary Actor: User
Stakeholders and Interests:
• User: Wants to list all the AE Sources available on the selected OPC Server.
Preconditions: An OPC Server has been selected (using Use Case 1)
Postconditions: A list of available AE Sources for the selected OPC Server is presented to User.
Screenshots: Screenshots 4, 5 and 6

Basic Flow:
1. Clicks on "Add A&E Subscription"
2. Clicks on "Select..." next to Sources

Frequency of occurrence: Rare. Only when alarms/events are to be added/removed from the surveillance system.

5.3.5 Use Case 3: List AE Areas

Primary Actor: User
Stakeholders and Interests:
• User: Wants to list all the AE Areas available on the selected OPC Server.
Preconditions: An OPC Server has been selected (using Use Case 1)
Postconditions: A list of available AE Areas for the selected OPC Server is presented to User.
Screenshots: Screenshots 4, 5 and 7

Basic Flow:
1. Clicks on "Add A&E Subscription"
2. Clicks on "Select..." next to Areas
Frequency of occurrence: Rare. Only when alarms/events are to be added/removed from the surveillance system.

5.3.6 Use Case 4: List AE Event Category Id's

Primary Actor: User
Stakeholders and Interests:
• User: Wants to list all the AE Event Category Id's available on the selected OPC Server.
Preconditions: An OPC Server has been selected (using Use Case 1)
Postconditions: A list of available AE Event Category Id's for the selected OPC Server is presented to User.
Screenshots: Screenshots 4, 5 and 8

Basic Flow:
1. Clicks on "Add A&E Subscription"
2. Clicks on "Select..." next to Event Category Id's
Frequency of occurrence: Rare. Only when alarms/events are to be added/removed from the surveillance system.

5.3.7 Use Case 5: List DA Items

Primary Actor: User
Stakeholders and Interests:

• User: Wants to list all the DA Items available on the selected OPC Server.
Preconditions: An OPC Server has been selected (using Use Case 1)
Postconditions: A list of available DA Items for the selected OPC Server is presented to User.
Screenshots: Screenshots 9, 10 and 11

Basic Flow:
1. Clicks on "Add DA Subscription"
2. Clicks on "Select..." next to Item
Frequency of occurrence: Rare. Only when alarms/events are to be added/removed from the surveillance system.

5.3.8 Use Case 6: Create AE Subscription

Primary Actor: User
Stakeholders and Interests:
• User: Wants the system to capture the AE alarms/events which he/she specifies in the subscription details.
Preconditions: An OPC Server has been selected (using Use Case 1)
Postconditions: An AE subscription is added to the configuration file.
Screenshots: Screenshot 12

Basic Flow:
1. Clicks on "Add A&E Subscription"
2. Enters details for the subscription (Use Cases 1-4 + specialities)
3. Saves the configuration (CTRL+S or via the menu "File/Save (as...)")
Frequency of occurrence: Rare. Only when alarms/events are to be added/removed from the surveillance system.

5.3.9 Use Case 7: Create DA Subscription

Primary Actor: User
Stakeholders and Interests:
• User: Wants the system to capture the DA alarms/events which he/she specifies in the subscription details.
Preconditions: An OPC Server has been selected (using Use Case 1)

Postconditions: A DA subscription is added to the configuration file.
Screenshots: Screenshot 12

Basic Flow:
1. Clicks on "Add DA Subscription"
2. Enters details for the subscription (Use Cases 1 and 5 + specialities)
3. Saves the configuration (CTRL + S or via the menu "File/Save (as...)"))
Frequency of occurrence: Rare. Only when alarms/events are to be added/removed from the surveillance system.

5.3.10 Use Case 8: Create connection to OPC Server

Primary Actor: OPC Listener
Stakeholders and Interests:
• User: Want the system to capture the alarms/events he/she has created in Use Cases 6 and 7.
• ACS: Wants to capture all important alarms/events.
• OPC Listener: Wants to establish a connection to an OPC Server defined in the configuration file.
Preconditions: OPC Listener (Windows Service) has been started.
Postconditions: An OPC Connection has been established and added to the collection of established connections.

Basic Flow:
1. Reads configuration file (XML)
2. Iterate through all subscriptions and extract OPC Servers
3. Establish the connection to the OPC Server and add the connection to the list of established connections
Frequency of occurrence: From time to time. Whenever the Windows Service is restarted. This should be done whenever the configuration file is changed.

5.3.11 Use Case 9: Create a Subscription

Primary Actor: OPC Listener
Stakeholders and Interests:
• User: Want the system to capture the alarms/events he/she has created subscriptions for

in Use Cases 6 and 7.
• ACS: Wants to capture all important alarms/events.
• OPC Listener: Wants to create a Subscription (AE or DA) to the selected OPC Server.
Preconditions: OPC Listener (Windows Service) has been started, and a subscription element has been located in the configuration file (Use Case 8).
Postconditions: A Subscription has been established and an event listener handling the incoming alarms/events has been initialized.

Basic Flow:
1. Reads the subscription details from the subscription element
2. Create an event listener to handle the incoming events
3. Create subscription to the OPC server with the details read from the configuration file
4. Associate the event listener with the subscription
Frequency of occurrence: From time to time. Whenever the Windows Service is restarted. This should be done whenever the configuration file is changed.

5.3.12 Use Case 10: Handle incoming AE Event

Primary Actor: OPC Listener
Stakeholders and Interests:
• User: Want the system to capture the alarms/events he/she has created subscriptions for in Use Cases 6 and 7.
• ACS: Wants to capture all important alarms/events.
• OPC Listener: Wants to handle the incoming AE event according to filters defined in the subscription details. **Preconditions:** OPC Listener (Windows Service) has been started, and a subscription has been created (Use Case 9).

Postconditions: The alarm/event is logged, and transmitted to the back-end.

Basic Flow:
1. An incoming alarm/event causes the event listener to call its handling function
2. Constructs an ACS Event object and initializes it with the values fetched from the incoming alarm/event
3. Sends the ACS Event to the back-end
Frequency of occurrence: Often. Every time an AE alarm/event is sent to the OPC

Listener from connected Automation Networks.

5.3.13 Use Case 11: Handle incoming DA Event

Primary Actor: OPC Listener
Stakeholders and Interests:
• User: Want the system to capture the alarms/events he/she has created subscriptions for in Use Cases 6 and 7.
• ACS: Wants to capture all important alarms/events.
• OPC Listener: Wants to handle the incoming DA event according to filters defined in the subscription details. **Preconditions:** OPC Listener (Windows Service) has been started, and a subscription has been created (Use Case 9).

Postconditions: The incoming alarm/event is checked. If the alarm/event meets requirements/criteria defined in the subscription, the alarm/event is logged, and transmitted to the back-end. Otherwise it is ignored.
Basic Flow:
1. An incoming alarm/event causes the event listener to call its handling function
2. Checks if the alarm/event meets the requirements/criteria defined in the subscription details associated with the event listener. If so, this Use Case continues, if not, it terminates here
3. Constructs an ACS Event object and initializes it with the values fetched from the incoming alarm/event
4. Sends the ACS Event to the back-end
Frequency of occurrence: Often. Every time an DA alarm/event is sent to the OPC Listener from connected Automation Networks.

5.3.14 Use Case 12: Handle event in back-end

Primary Actor: OPC Listener
Stakeholders and Interests:
• User: Want the system to capture the alarms/events he/she has created subscriptions for in Use Cases 6 and 7.
• ACS: Wants to capture all important alarms/events.
• OPC Listener: Wants to temporarily store the incoming ACS Event, and transmit it to the ACS as soon as possible. **Preconditions:** OPC Listener (Windows Service) has been

started and an AE or DA alarm/event has been sent to the back-end.

Postconditions: The ACS Event is temporarily stored, and once transmitted to ACS, it is deleted from temporary storage.

Basic Flow:

1. An ACS Message is received from the front-end
2. The ACS Message is temporarily stored in a local database
3. As soon as possible, the ACS Message is retrieved from the local storage, and sent to the ACS where it is handled

Frequency of occurrence: Very often. Every time an alarm/event of any kind passes through the system.

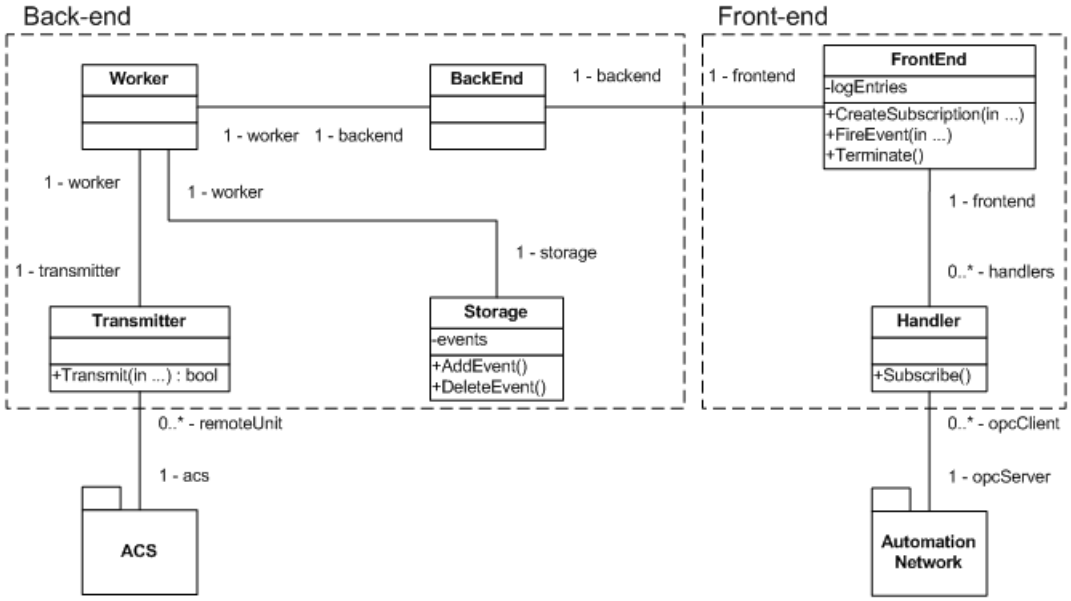


Figure 8: Domain Model

5.4 Domain Model

Traditionally a domain model represents the domain as seen from an end users point-of-view with focus placed on the physical entities present in the system. However in this case such a model is insufficient, as the intention is to model the system with some very specific requirements involving restrictions on some low-level entities. The system diagram, Figure 6, is in this project considered sufficient as a "traditional domain model" showing the relations between the physical entities, while the domain model shown in Figure 8 digs a bit deeper into the low-level specifics.

In Figure 8 there are two packages shown: ACS and Automation Network. These two packages represent the domain models for the ACS and for automation networks in general. The domain models for these are shown in Figure 9 and Figure 10 respectively. As mentioned earlier in the report, these models will not be discussed in great detail, as they are not really a part of the project itself. The focus will therefore be placed on the entities in Figure 8 and the constraints which are associated with these entities.

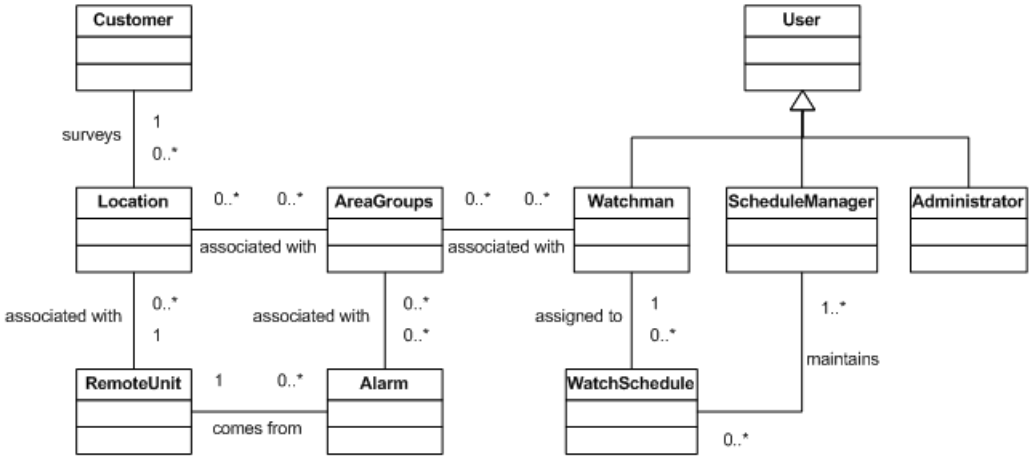


Figure 9: ACS Domain Model

5.4.1 Entities

The entities in the Domain Model, do not represent physical concepts as they do in a traditional domain model. This section will describe what the entities represent, as well as what purpose they serve.

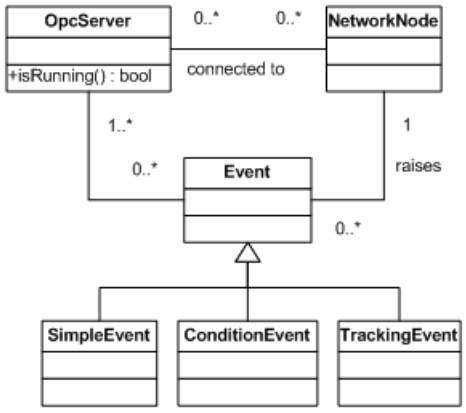


Figure 10: Automation Network Domain Model

The Domain Model is split up into two parts: A Back-end and a Front-end.
The Front-end is the part which handles connections to OPC Servers and listens for incoming alarms/events. The Front-end basically consists of a `FrontEnd` which contains a number of handlers, namely one for each subscription. The `Handler` is simply a means for performing the correct task with the incoming alarms/events. After the `Handler` has processed the alarm/event, it sends it backwards to the Back-end via the `FrontEnd`.
The Back-end is the part which maintains a local storage of incoming alarms/events until they with certainty have been delivered to the ACS. The Back-end basically consists of a `BackEnd` which initializes a `Worker` which sole purpose is to store alarms when requested to, and continuously try to transmit alarms stored in the storage (and delete them on successful transmission). The `Storage` and `Transmitter` are abstract entities, where `Storage` represents some form of storage, in this case an relational database, and `Transmitter` represents some form of SOAP transmission module, in this case a C# class generated using a WSDL file. The Back-end has a functional requirement requiring that only one `Storage` is present, and also that only one `Worker` is running. These requirements are further specified in the constraints below.

5.4.2 Constraints

The constraints described in this section will all be modelled using OCL [7]. This allows for specification of pre and post conditions as well as invariants (expressions that must evaluate to true for any allowed input parameters). The following is the OCL code associated with

the Domain Model in Figure 8:

```
-- Constraints for the Handler
context Handler inv:
    opcServer->size() = 1
context Handler::Subscribe() : Bool
    pre  : self.opcServer.isRunning()
    post : self.opcServer.opcClients->size()
            = opcServer.opcClients@pre->size() + 1
    post : selc.opcServer.opcClients->include(self)

-- Constraints for the FrontEnd
context FrontEnd inv:
    backEnd->size() = 1
context FrontEnd::CreateSubscriptions(configFile : ConfigFile)
    pre  : self.handlers->size() = 0
    post : self.handlers->size()
            = configFile.subscriptionCount()
context FrontEnd::FireEvent(event : Event) : Bool
    post : self.logEntries->size()
            = self.logEntries@pre->size() + 1
context FrontEnd::Terminate()
    post : self.handlers->size() = 0

-- Constraints for the Worker
context Worker
    inv: transmitter->size() = 1
    inv: storage->size() = 1

-- Constraints for the Transmitter
context Transmitter inv:
    acs->size() = 1
context Transmitter::Transmit(event : Event) : Bool
    pre  : not self.acs.events->includes(event)
    post : self.acs.events->includes(event)
    post : self.acs.events->size()
            = self.acs.events@pre->size() + 1

-- Constraints for the Storage
context Storage inv:
    events->forAll(e1, e2 | e1 <> e2 implies e1.id <> e2.id)
context Storage::AddEvent(event : Event) : Bool
    pre  : not self.events->includes(event)
```

```
    post : self.events->size() = self.events@pre->size() + 1
context Storage::DeleteEvent(event : Event) : Bool
    pre  : self.events->includes(event)
    post : self.events->size() = self.events@pre->size() + 1
    post : not self.events->includes(event)
```

In order to give the reader a better understanding of the OCL code, key points will be highlighted in the following textual descriptions.

`Handler`
An Event Handler can be associated with one and only one OPC Server. This is enforced in the invariant, `opcServer->size() = 1`, for the `Handler`.
The `Subscribe()` function makes sure that the OPC Server is running, and adds itself to the OPC Servers internal list of subscribers (clients).

`FrontEnd`
A Front-end can be associated with one and only one Back-end. This is enforced in the invariant, `backEnd->size() = 1`, for the `FrontEnd`.
The `CreateSubscriptions(...)` function takes a configuration file as argument, and creates handlers for all the subscription details found in the configuration file.
The `FireEvent(...)` function adds a log entry indicating that an event was fired. Naturally like other functions it performs other tasks/actions, however these are not described using OCL, but rather using RSL (R(AISE) Specification Language) discussed in the next subsection.
The `Terminate()` function removes all existing Event Handlers and shuts down the Front-end.

`Worker`
The Worker can be associated with only one transmitter, and only one storage. This is enforced in the invariants: `transmitter->size() = 1` and `storage->size() = 1` for the `Worker`.

`Transmitter`
A Transmitter can be associated with one and only one ACS. This is enforced in the invariant, `acs->size() = 1`, for the `Transmitter`.
The `Transmit(...)` function adds the given event to the ACS's internal list of events. It is

a precondition that the event is not already in the ACS.

`Storage`
All events in the storage are unique. This is enforced in the invariant, `events->forAll(e1, e2 | e1 <> e2 implies e1.id <> e2.id)` , in the `Storage`, indicating that all events have unique id's.
The `AddEvent(...)` function adds the given event to the list of events given that it does not already exists in the list.
The `DeleteEvent(...)` function removes the given event from the list of events given that it exists in the list.

5.5 Formal Specifications

This section presents the formal specifications created for the selected parts of the system. The specifications evolved from very basic to more complicated/specific through a number of iterations. In order to keep from crowding the report with all the specifications, only the first and last iteration of each specification is included. The remaining iterations can be found in Appendix D. The specifications give an initial insight into the data structure and enforce the functional requirements stated in the previous section.

The initial iterations are very abstract imperative specifications whereas the closer one moves towards the final iteration, the more applicative and concrete they become.

5.5.1 Creating Subscriptions to OPC Servers

These specifications formally specify how the OPC Listener should connect to OPC Servers according to a configuration file specified by the user (using the OPC Listener Configuration Tool). The first iteration of the specification is shown below:

scheme Connecting_it1 =
  class
    type
      Details,
      SubscriptionElem = **Text** × Details,
      ServerElem = **Text** × SubscriptionElem-**set**,
      ConfigFile = ServerElem-**set**,

Handler = **Text** × SubscriptionElem

value
empty : ConfigFile,
createSubscriptions : ConfigFile → Handler-**set**

axiom
empty ≡ {},
∀ C : ConfigFile •
createSubscriptions(C) ≡
{(T1, S1) |
(T1, S1) : Handler •
∃
(T2, S2) : ServerElem,
(T3, D) : SubscriptionElem
•
(T2, S2) ∈ C ∧ (T3, D) ∈ S2 ∧
(T3, D) = S1 ∧ T1 = T2}

end

This specification defines a simple data structure describing the contents of the configuration file, and also contains an element, `Handler`, which is an Event Handler handling incoming requests from the OPC Server. Furthermore there is one function specified namely `createSubscriptions()` taking a configuration file as argument. This function has an axiom defined ensuring that it returns a list of Event Handlers matching all the handlers specified in the given configuration file.

After a few iterations the specification evolved into a more object oriented specification, being more applicable for actual implementation in a language like C#. The final iteration of this specification is shown below:

object Connecting_it4 :
class
variable
connectedServers : ServerElem-**set**,
handlers : Handler-**set**,

```
    /∗ The list within the OPC server of all the connected
        subscribers ∗/
    subscribers : Handler-set

type
    AeSpecifics,
    DaSpecifics,
    RemoteUnit,
    AreaGroups,
    MessageFormat,
    CommonDetails =
        RemoteUnit × AreaGroups × MessageFormat,
    Details ==
        mk_AE_details(CommonDetails, AeSpecifics) |
        mk_DA_details(CommonDetails, DaSpecifics),
    SubscriptionType == AE | DA,
    ServerType == AE | DA | BOTH,
    SubscriptionElem ==
        mk_subscription(
            subscription_name : Text,
            subscription_type : SubscriptionType,
            subscription_details : Details),
    ServerElem ==
        mk_server(
            server_name : Text,
            server_type : ServerType,
            server_subscriptions : SubscriptionElem-set),
    ConfigFile = ServerElem-set,
    Handler ==
        mk_AE_subscription(
            ae_name : Text, details : AeSpecifics) |
        mk_DA_subscription(
            da_name : Text, details : DaSpecifics)
```

```
value
  connectToServers : ConfigFile → Bool,
  connectToAeServers : ServerElem-set → Bool,
  createAeSubscription : SubscriptionElem → Bool,
  connectToDaServers : ServerElem-set → Bool,
  createDaSubscription : SubscriptionElem → Bool

axiom
  /∗ Connect to servers in config file
       post: There must be created the same number
             of handlers as there are subscriptions
             in the config file.
  ∗/
  ∀ C : ConfigFile •
    connectToServers(C) ≡
      let
        servers = {S | S : ServerElem • S ∈ C},
        aeServers =
          {server |
           server : ServerElem •
             server ∈ servers ∧
             (server_type(server) = AE ∨
               server_type(server) = BOTH)},
        daServers =
          {server |
           server : ServerElem •
             server ∈ servers ∧
             (server_type(server) = DA ∨
               server_type(server) = BOTH)}
      in
        (connectToAeServers(aeServers) ∧
         connectToDaServers(daServers))
          post
          (card handlers =
```

```
                card {s |
                        s : SubscriptionElem •
                            (∀ serv : ServerElem •
                              serv ∈ C ∧
                              s ∈
                                server_subscriptions(serv))})
        end,
/* Connect to AE servers
        post: The given set of servers must be a
                subset of the servers connected to.
*/
∀ S : ServerElem-set •
    connectToAeServers(S) ≡
        let
            subs =
                {subscription |
                 subscription : SubscriptionElem,
                 server : ServerElem •
                    server ∈ S ∧
                    subscription ∈
                        server_subscriptions(server) ∧
                    server_type(server) = AE ∧
                    createAeSubscription(subscription)}
        in
            (connectedServers := connectedServers ∪ S ;
             true)
                post
                (S ⊆ connectedServers)
        end,
/* Create an AE Subscription
        post: The handler is is added to the list of
                handlers.
*/
∀ S : SubscriptionElem •
```

```
    createAeSubscription(S) ≡
        if (subscription_type(S) = AE)
        then
            let
                mk_AE_details(common, aeDetails) =
                    subscription_details(S),
                handler =
                    mk_AE_subscription(
                        subscription_name(S), aeDetails)
            in
                (handlers := handlers ∪ {handler} ; true)
                    post
                    handler ∈ handlers
            end
        else true
        end,
/* Connect to DA servers
        post: The given set of servers must be a
                subset of the servers connected to.
*/
∀ S : ServerElem-set •
    connectToAeServers(S) ≡
        let
            subs =
                {subscription |
                 subscription : SubscriptionElem,
                 server : ServerElem •
                    server ∈ S ∧
                    subscription ∈
                        server_subscriptions(server) ∧
                    server_type(server) = DA ∧
                    createAeSubscription(subscription)}
        in
            (connectedServers := connectedServers ∪ S ;
```

```
                true)
                  post
                  (S ⊆ connectedServers)
            end,
        /* Create a DA Subscription
             post: The handler is is added to the list of
                   handlers.
        */
        ∀ S : SubscriptionElem •
          createAeSubscription(S) ≡
            if (subscription_type(S) = DA)
            then
              let
                mk_DA_details(common, daDetails) =
                  subscription_details(S),
                handler =
                  mk_DA_subscription(
                    subscription_name(S), daDetails)
              in
                (handlers := handlers ∪ {handler} ; true)
                  post
                  handler ∈ handlers
              end
            else true
            end,
        /* The list of connected handlers must always
           be a subset of the OPC servers list of
           connected handlers
        */
        handlers ⊆ subscribers
    end
```

In this specification it is obvious that the data structure is somewhat more complicated, comprising enough information to completely model the connection functionality. It also defines local sets (variables) to hold the servers which have an open connection, the set of han-

dlers, as well as the OPC Servers list of subscribers. The function `createSubscriptions()` from the first iteration has now been split up into several functions handling different areas of the connect routine. `connectToServers()` is a functions which from a given configuration file, creates two lists: One containing AE server elements and one containing DA server elements. These lists are then sent to the functions `connectToAeServers()` and `connectToDaServers()` respectively. These functions extract the subscriptions stored within the server element and call the functions `createAeSubscription()` and `createDaSubscription()` respectively. These functions create the actual Event Handler and adds it to the internal list of event handlers. To each of these functions is associated one or more post conditions describing what must hold after the function has been called. These post conditions are not described further, as they are textually described using comments in the specification above.

5.5.2 Persisting Alarms and Transmitting to ACS

These specifications formally specify how the back-end persists incoming alarms, and transmits them to the ACS. The first iteration of this specification is shown below:

scheme Persisting_it1 =
  class
    type Event, Storage = Event-**set**

    value
      empty : Storage,
      add : Storage × Event → Storage,
      delete : Storage × Event → Storage,
      hasWaitingEvent : Storage → **Bool**,
      retrieveEvent : Storage → Event,
      transmit : Event → **Bool**

    axiom
      empty ≡ {},
      /∗ Add an event to the storage ∗/
      ∀ E : Event, S : Storage • add(S, E) ≡ S ∪ {E},
      /∗ Delete an event from the storage ∗/
      ∀ E : Event, S : Storage • delete(S, E) ≡ S \ {E},
      /∗ Check if the storage has waiting events ∗/

```
        ∀ E : Event, S : Storage •
            hasWaitingEvent(S) ≡ (∃ E : Event • E ∈ S)
    end
```

This specification defines a very simple data structure consisting of en event, and a storage containing persisted events. Furthermore it specifies functions which manipulates the storage, as well as transmit events. In this specification there are only defined axioms for the `add()`, `delete()`, and `hasWaitingEvent()` functions at this point. The first axiom ensures that the add functions adds the given event to the given storage, the second axiom ensures that the delete function removes the given event from the given storage, and the third axiom checks whether or not there exists any events in the storage.

After a few iterations this specification evolved into a more object oriented specification, being more applicable for actual implementation in a language like C#. The final iteration of this specification is shown below:

```
object Persisting_it5 :
    class
        variable storage : Event-set, acs_storage : Event-set

        type
            Id = Int,
            Timestamp = Int,
            Code,
            UnitId,
            AreaGroups,
            Event ==
                event(
                    id : Id,
                    timestamp : Timestamp,
                    code : Code,
                    unitid : UnitId,
                    areagroups : AreaGroups,
                    text : Text)

        value
```

add : Event → **write** storage **Unit**,
delete : Event → **write** storage **Unit**,
hasWaitingEvent : **Unit** → **read** storage **Bool**,
retrieveEvent : **Unit** → **read** storage Event,
transmit : Event → **Bool**,
back_end : **Unit** → **Unit**,
run : **Unit** → **Unit**,
/∗ Reception of an alarm ∗/
alarmFromFrontEnd : **Unit** → Event

axiom
/∗ Add an event to the storage
pre: The event cannot already be in the storage
post: The event is added to the storage
∗/
∀ E : Event •
add(E) ≡
if (E ∉ storage)
then storage := storage ∪ {E}
end,
/∗ Delete an event from the storage
pre: The event is in the storage
post: The event is deleted from the storage
∗/
∀ E : Event •
delete(E) ≡
if (E ∈ storage) **then** storage := storage \ {E}
end,
/∗ Check if the storage has waiting events ∗/
hasWaitingEvent() ≡ **card** storage > 0,
/∗ Retrieve an event (the oldest) from the storage
pre: The event is in the storage
post: The event is still in the storage (use
delete to remove)

```
*/
∀ E : Event •
    retrieveEvent() as E post
        hasWaitingEvent() ∧ E ∈ storage
    pre
        hasWaitingEvent() ∧ E ∈ storage ∧
        (∀ E1 : Event •
            E1 ∈ storage ∧ E1 ≠ E ∧
            timestamp(E) < timestamp(E1)),
/* Transmit the event to ACS
        pre:  The event is in the storage
        post: The event is added to the ACS storage
*/
∀ E : Event •
    transmit(E) ≡
        (acs_storage := acs_storage ∪ {E} ; true)
    pre E ∈ storage,
/* Events have uniqe id's */
∀ E1, E2 : Event • E1 = E2 ⇒ id(E1) = id(E2),
/* The back−ends functionality is to continuously
    perform the run() function, as well as handle
    incoming alarms
*/
back_end() ≡ run() ‖ add(alarmFromFrontEnd()),
/* The body of the back−end. Continuously check
    if alarms are availble for transmission, and
    transmit them.
*/
run() ≡
    if hasWaitingEvent()
    then
        let event = retrieveEvent() in
            if transmit(event) then delete(event) end
        end
```

```
        end
    end
```

In this specification the data structure has been extended to hold all the information which events consist of. Furthermore it has defined local sets (variables) representing the back-end storage as well as the storage within the ACS. The functions remain the same as in the previous specification, however they now manipulate the local sets, and they are associated with a number of pre and post conditions. The specification now also include two functions which are actually processes although not using channels, namely `back_end()`, and `run()`, which describe how the back-end continuously listen for incoming alarms as well as transmit persisted alarms to the ACS concurrently. The specifications also define axioms for all the functions, as well as include a new axiom stating that all events are unique (have unique id's). Pre and post conditions are described textually as comments in the specification above, and are therefore not discussed further here.

One thing that might seem odd in the specification is the function `add()`. This functions actually checks whether an element is contained in a set before performing the union operation, however the mathematical rule states that sets only contain unique elements, and therefore the union would not extend the set if the element was already in the set. The reason this check is included, is to allow for implementers to easily switch to lists instead of sets if desired.

6 Design

This section describes the design of the OPC Listener in details. The design was created to be as flexible as possible in order to allow Zonith A/S to easily create alarm/event listeners for others types of networks/systems.

6.1 Design Patterns

Design Patterns are very helpful in finding solid design for a system without wasting too much time on re-designing modules. Basically Design Patters are well tested designs, created at an abstract level in order to fit into a wide range of solutions. They emerge from developers having similar problems and therefore decide to create a generic design solving the problem.

The analysis section gave rise to two areas where Design Patterns are applicable. The chosen patterns and their associated areas are described below:

Template Method pattern
The Template Method pattern is used when several classes use a similar algorithm with minor differences. The general algorithm is then implemented in an abstract class and hooks, abstract methods, are created handling the differences between the classes. The hooks are called from within the algorithm in order to retrieve the differences between the concrete implementations. A class diagram of this pattern is shown in Figure 11.

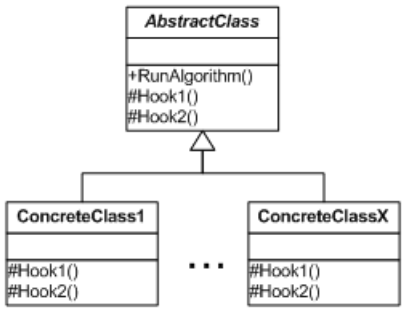


Figure 11: Template Method Design Pattern

Singleton pattern

The Singleton pattern is a pattern used when wanting to ensure that one, and only one, instance of an object is ever created. The constructor of the class is labelled as private, and thus only `GetInstance()` will be allowed to instantiate the class. This method then ensures that only one instance is ever created. A class diagram of this patterns is shown in Figure 12.

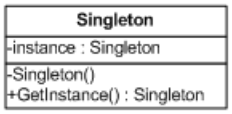


Figure 12: Singleton Design Pattern

There may indeed be more areas where Design Patterns could be applicable, however some of these might first be discovered while implementing the system.

6.2 Modular design

In order to design something which can be reused in many places, one must think abstractly and leave out specifics. Requirement 1, stating that the OPC Listener had to be a Windows Service, introduced the need for an abstract Windows Service framework designed for alarm/event listeners. This framework, although specific in the sense that it is used only for alarm/event listeners to communicate with an ACS, had to be generic enough to allow Zonith A/S to easily implement future listeners utilizing the framework.

6.2.1 Windows Service Framework

In the .NET Framework, creating a Windows Service is fairly easy. Basically all you have to do is construct a class dericed from the class `System.ServiceProcess.ServiceBase`, override the hooks `OnStart()` and `OnStop()` (and more if needed...), and give the service a unique name which is used when the service is installed on a system. In order to create a flexible Windows Service Framework, a decision was made to separate the framework into a front-end and a back-end. The front-end taking care of all incoming alarms/events, sending them to the back-end where they are stored temporarily and sent on to the ACS. A class diagram of the Windows Service Framework is shown in Figure 13. The classes and interfaces are described in more detail below:

`AbstractService`
This abstract class is the Windows Service itself. It implements the `OnStart()` and `OnStop()` methods, and defines two hooks, `GetServiceName()` and `GetListener()` which should be overridden by classes that are derived from this abstract class. The purpose of this class is merely to start and stop the listener.

`AbstractListener`
This abstract class is the middleman of the listener. It implements the methods `Start()` and `Terminate()` called by the service. Furthermore it implements a `FireEvent()` method used by the front-end to delegate the ACS Event to the back-end. It also defines two hooks, `GetBackEnd()` and `GetFrontEnd()`, which should be overridden by classes that implement this abstract class. As mentioned above, the purpose of this class is to act as the middleman between the front- and the back-end.

`IBackEnd`
This interface defines which methods any back-end should implement. These are `Initialize()` and `Terminate()` used to start and stop the back-end, as well as `FireEvent()` which is the method handling the ACS Event delegated to the back-end from the front-end.

`IFrontEnd`
This interface defines which methods any front-end should implement. These are `Initialize()`, `Start()`, and `Terminate()` used to start and stop the front-end. Unlike the back-end which should always be available and therefore started once initialized, the front-end has a dedicated `Start()` method allowing it to be started at any point in time after being initialized. Furthermore it defines the method `FireEvent()` used to delegate ACS Events to the back-end.

`AbstractFrontEnd`
This abstract class is an implementation of the front-end interface which implements the one feature used by all front-ends, namely the `FireEvent()` method. The remaining methods defined by the interface are left as virtual methods in this class, indicating that all subclasses can (in this case must) override them.

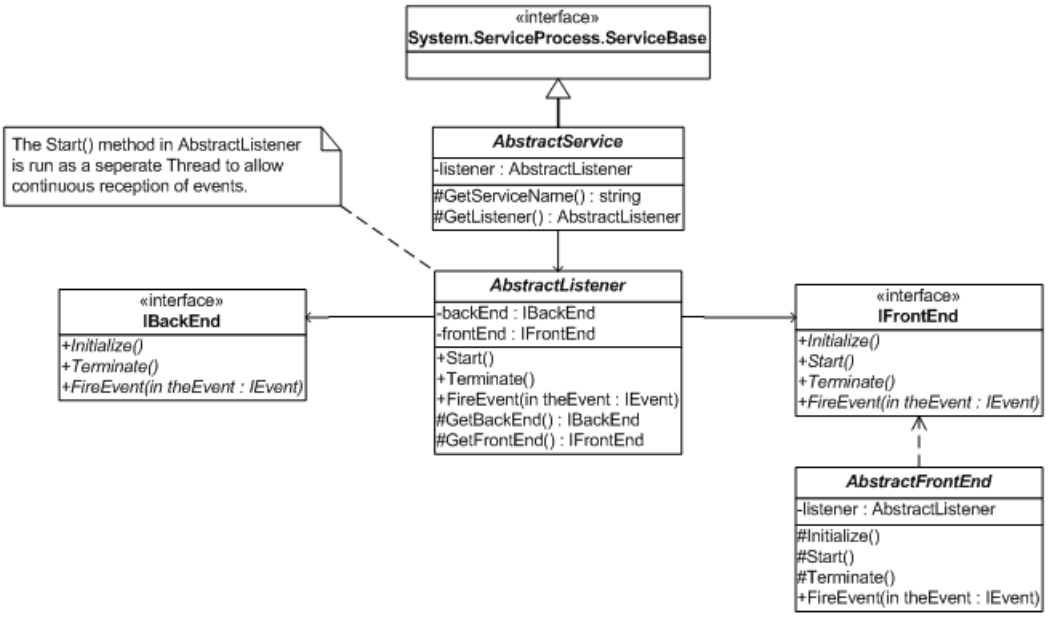


Figure 13: Abstract Windows Service Framework

6.2.2 Default back-end

Zonith A/S already has a number of listeners in their product line, all of which use the same back-end which basically stores ACS Events locally until they are able to send the Event to the ACS which is supposed to handle them. All previous listeners are written in Java, and the default back-end therefore exists in Java. In order to keep maintenance of code down to a minimal, the back-end was translated as closely as possible into C# code, thereby not introducing new features which could confuse developers of the Java back-end. The default back-end was then combined with the Windows Service Framework above, in order to create a single class library holding all the necessary classes. The introduction of the default back-end into the Windows Service Framework can be seen in Figure 14. The region marked as *Zonith's default back end* in the diagram is the implementation of the region marked *Back-end* in the Domain Model presented earlier. The classes and interfaces are described in more detail below:

`IStorageManager`

This interface defines the methods which all storage managers need to implement. A storage manager is a class which can temporarily store ACS Events in some way, until they have been

sent to the ACS. The methods in this interface are not shown in the diagram, but they are basically concerned with starting/stopping the storage manager, as well as adding/retrieving ACS Events, as well as checking for pending ACS Events.

`IWorker`
This interface defines the methods which a worker must implement. A worker is the class which continually checks for pending ACS Events. If there are any, it delivers them to the transmit proxy which sends them to the ACS, otherwise it goes to sleep, and is woken up periodically.

`ITransmitProxy`
This interface defines the single method `Transmit()` which transmit proxies need to implement with code that sends an ACS Event to the ACS.

`HSQLStorageManager`
This class is the storage manager used by default. Its storage is based on HSQL, defined in the Theory section above, thereby storing ACS Events locally in binary files, accessible via SQL commands. In other words, it contains a small DBMS where it temporarily stores files.

`CoreTransmitProxy`
This class is the transmit proxy used by default. Its only purpose is to send ACS Events through an XML/SOAP interface to the ACS. This is done using a proxy class generated from a web service definition written in WSDL (defined in the Theory section above).

`CoreListener`
This abstract class is the central class of the default back-end. It loads values from the properties file such as "listener name", "company number", and "ACS end-point", used to establish a valid connection to the ACS.

`PersistentListener`
This class is a subclass of the `CoreListener` and is the entry point to the back-end seen from the Windows Service Frameworks point-of-view. It is implemented as a singleton to ensure there cannot exists multiple storages which could confuse the worker thread.

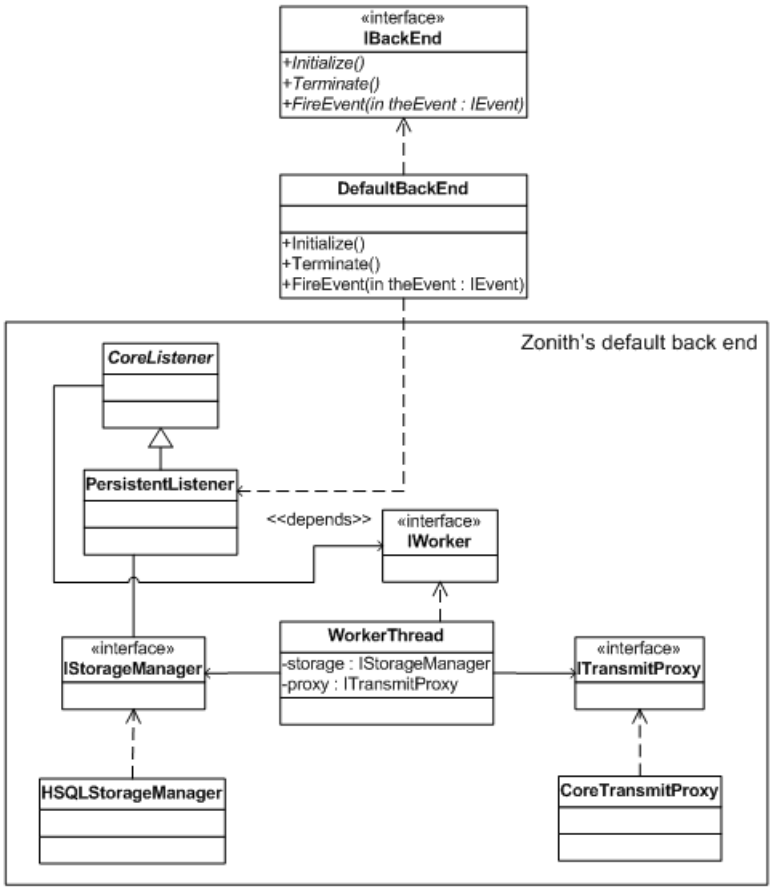


Figure 14: Default back-end combined with the Abstract Windows Service Framework

6.2.3 ACS Events

An ACS Event is the term used for an object containing information about an alarm/event from any given source. In the Framework described above, ACS Events are objects which implement the interface `IEvent`. In the existing Java back-end mentioned earlier, there was only one type of event. This event was ported to the new framework as well, in order to allow Java developers to easily port old listeners to the new framework. This event is implemented as `DefaultEvent` in the Windows Service Framework. Another event, which is aimed at OPC listeners, is the `OpcEvent`. This class contains the same information as the default event, but furthermore it contains information about the OPC server, the OPC server host, and the subscription from which the event emerged. A class diagram presenting the event types is shown in Figure 15.

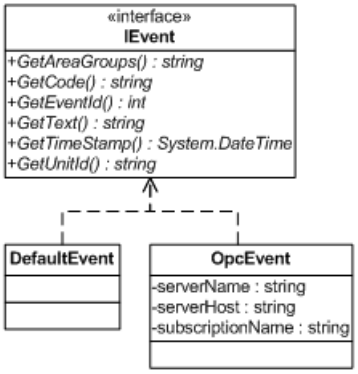


Figure 15: ACS Events

6.2.4 The front-end

The front-end has to be able to handle two different types of OPC subscriptions, namely subscriptions to an Alarms & Events (AE) OPC Server, and to a Data Access (DA) OPC Server. This separation of subscription types is maintained in the design of the front-end. Basically the front-end consists of a front-end class, `OpcFrontEnd` which extends the `AbstractFrontEnd` defined above, as well as two event handler classes that perform appropriate actions on incoming events. A class diagram of the front-end can be seen in Figure 16. The diagram shows a package called "C# OPC API", which is a third-party API purchased to ease the writing of OPC code, thus enabling the use of C# as the programming language instead of low level C++ programming. The classes shown in Figure 16 are described in more detail below:

`OpcService` and `OpcListener`
The classes are merely classes which implement the hooks defined in the `AbstractService` and `AbstractListener` classes. `GetListener()` in `OpcService` returns an `OpcListener` and `GetFrontEnd()` in `OpcListener` returns an `OpcFrontEnd`. By not overriding the `GetBackEnd()` method, `OpcListener` indicates that it wants to use the `DefaultBackEnd`.

`OpcFrontEnd`
This class is the OPC front-end itself. Its purpose is to load the configuration file and establish connections to all the servers listed in it. Furthermore it must create subscriptions to all the subscriptions listed in the configuration file, and create appropriate event handlers

to handle incoming alarms/events.

`AeEventHandler`
This class is the event handler for AE alarms/events. It contains one public method, `HandleEvent()`, which is called upon reception of an alarm/event. The `HandleEvent()` method has to create an ACS Event from the incoming alarm/event, and initialize it with the appropriate values. Once the ACS Event is created, it is sent to the back-end via the `FireEvent()` method defined in the `AbstractFrontEnd`.

`DaEventHandler`
This class is the event handler for DA alarms/events. Like the `AeEventHandler` it contains only one method, `HandleEvent()`, which is called upon reception of an alarm/event. This event handler has a little more work to do, because it has to check whether incoming alarm/event values lie within the limits defined in the subscription, and therefore qualify as an ACS Event. If this is the case, an ACS Event is created just as in the `AeEventHandler` and sent to the back-end via the `FireEvent()` method defined in the `AbstractFrontEnd`.

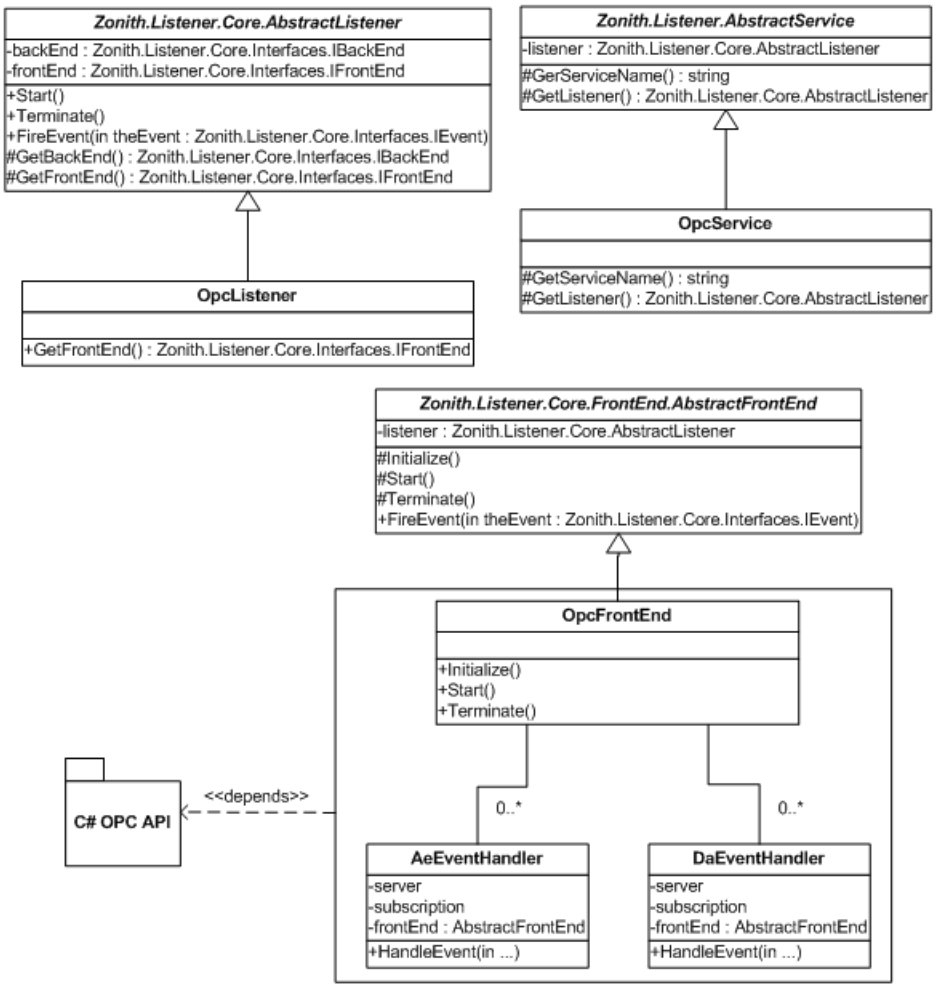


Figure 16: The front-end

6.3 Intercommunication

This section deals with the flow of information in the system designed in the previous subsection. It should give the reader a better understanding of how the different classes communicate with each other in order to make the system function correctly.

6.3.1 Sequence Diagrams

Sequence diagrams are often used in UML to clarify the flow of information. The sequence diagram shown in Figure 17 shows the the sequence of events that take place from starting

the OPC Listener until the ACS Event is delivered to the back-end. Communication with the OPC Server displayed in the diagram depends on the "C# OPC API" discussed above, but that layer is left out to simplify the diagram, and make it more understandable. Once the

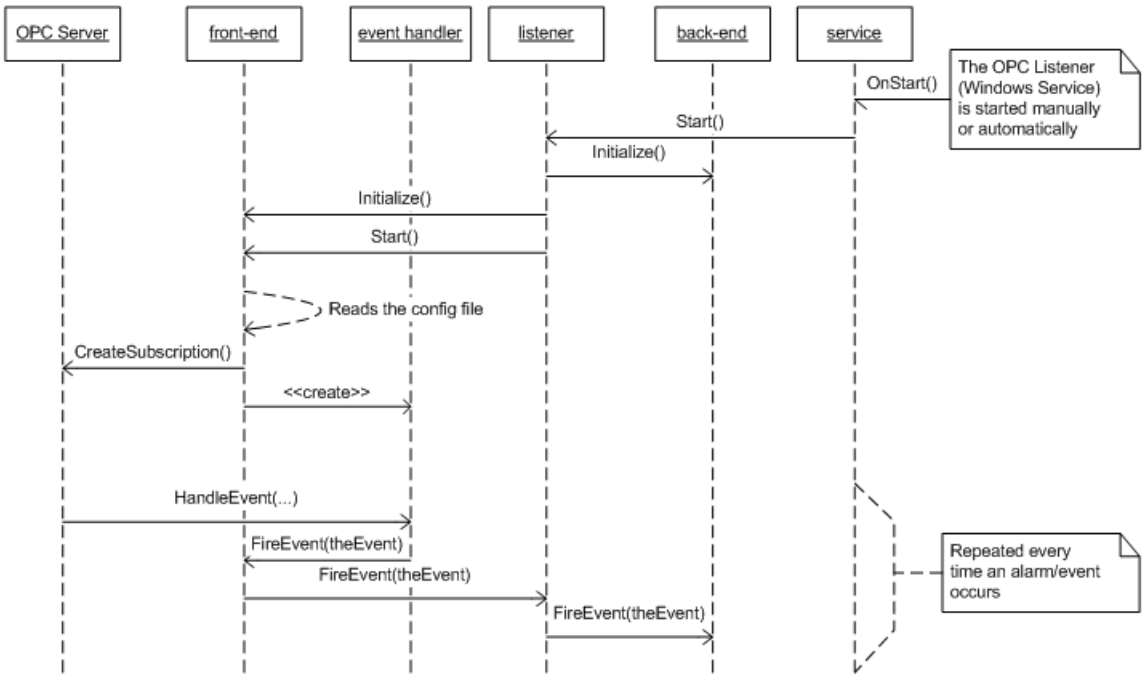


Figure 17: Sequence diagram

ACS Event, represented by *theEvent* in Figure 17, reaches the back-end. `DefaultBackEnd` fetches a reference to the `PersistentListener` singleton, and fires the event using the method `FireEvent()`. When the back-end is initialized, the `PersistentListener` instantiates a new `HSQLStorageManager` and a new `CoreTransmitProxy`. `PersistenListener`'s superclass, `CoreListener` spawns a `WorkerThread`, handing it references to the storage manager and the transmit proxy. This worker thread will then run until the OPC Listener is stopped, continuously fetching pending events from the storage manager and delivering them to the transmit proxy. The `FireEvent()` method simply adds the ACS Event to the storage manager, and wakes up the worker thread afterwards. This causes the ACS Event to be fetched from the storage manager, delivered to the transmit proxy, and finally received by the ACS. A sequence diagram of how this storing/fetching/sending ACS Events occurs is shown in Figure 18.

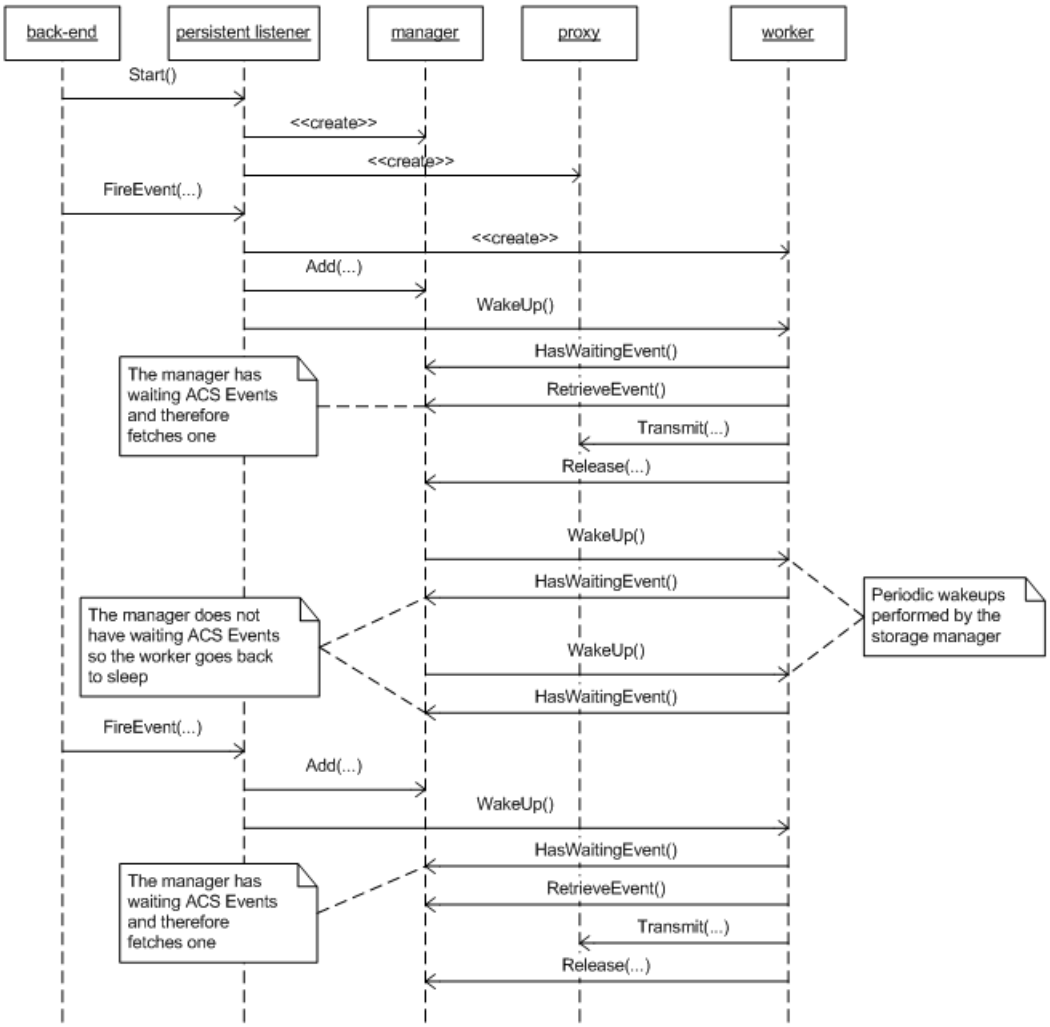


Figure 18: Sequence diagram for the back-end

6.3.2 OPC Listener to ACS

Communication from the OPC Listener to the ACS takes place using XML/SOAP based Web Services as mentioned in previous sections. Creating a Web Service Client using the .NET Framework is very simple. You simply supply the WSDL file describing the Web Service, along with the name of the programming language you are using, to a small executable utility that ships with the .NET Framework, and this will generate a client class for you. This class simply needs to be added to your project, and it will supply you with all the methods defined in the WSDL file.

7 Implementation

This section goes into details with how some of the ideas from previous sections have been implemented in the code. This is done mainly by presenting an excerpt of the source code followed by a text describing it. The complete source code can be found in Appendix H.

7.1 Coding Conventions

Following a coding convention makes it a lot easier for other programmers to read the code. This increases the level of understanding, and decreases the time required to understand the inner workings of the code. Unfortunately no standard coding convention for the C# programming language exists (that I am aware of), like the one found in Java, namely Suns Java Coding Conventions. However, surfing the net, led to a paper discussing "Guidelines and Best Practices" for the C# programming language. I adopted this paper as the coding convention of choice, and thus it is included in the report, see Appendix E. The paper can also be found at the following address:

http://www.idesign.net/idesign/download/IDesign%20CSharp%20Coding%20Standard.zip

7.2 Code Maintenance

Maintaining and extending code can be a tedious job, especially if the code is not documented/commented well. In order to easy future work on the code, it was decided to use the XML style commenting which is standard in C#. This commenting style, which hereafter will be referred to as XMLDoc, has many similarities to the JavaDoc used by Java developers. XMLDoc however has a few advantages over JavaDoc since it is XML based, and thus allows for data manipulation/representation of various types. The XML file generated can be used by the standard Windows Help function, it can generate Web Pages (like JavaDoc), or one can use XSLT to generate nearly any form of presentation desired.

In order to generate XMLDoc for a class, one needs to add the desired XML tags before the elements being documented. Elements can be classes, member variables, or methods. An example of how XMLDoc looks with a simple class is shown below:

```
namespace mycompany.myproduct;

/// <summary>
```

```
/// This is my class. It returns my string when asked to.
/// </summary>
public class MyClass {

    /// <summary>
    /// This is my private string
    /// </summary>
    private string m_myString;

    /// <summary>
    /// My private method fetching my private string and appending the given number.
    /// </summary>
    /// <param name="number">
    /// The number which is appended to my private string.
    /// </param>
    /// <returns>
    /// My private string appended with the given number.
    /// </returns>
    public string GetMyString(int number)
    {
        return m_myString + " - " + number;
    }
}
```

7.3 Interfaces and Implementations

Defining a system through interfaces is a great way of separating definitions, interfaces, from implementations. Anyone using the classes in the system must access them via the defined interfaces thereby hiding the actual implementation. When doing this, developers have free hands to code their implementation however they see fit, as long as it implements the defined interface. This technique comes in handy when changes are required in the back-end of a system which is not visible to the end user. Since the end user is accessing the system via the interfaces, the changes should not affect him/her/it. An example of this could be a class maintaining a list of some sort. One could define an interface with some methods manipulating the list. The first assumption could be that the list always has a fixed size, and therefore the implementation is made using a static array. A sudden change in the requirements now require the list to be dynamic, and thus the developer can simply change the implementation, but leave the interface as it was, and the end user will never know the difference.

7.4 Abstract Classes

If two classes contain similar information, with only minor differences, it is always a good idea to extract the common information into an abstract class, and make the two classes subclasses of this class. This allows a developer to handle the two classes in the same manner when working on the information which is common. To illustrate an example of this, the following code excerpt has been taken from `OpcFrontEnd`:

```
private void ConnectToComServers(ArrayList comServers)
{
    // Iterate through all servers
    foreach (OpcServerNode server in comServers)
    {
        // Iterate through all subscriptions in for the server
        foreach(TreeNode node in server.Nodes)
        {
            // Convert the node to the abstract subscription class
            OpcSubscriptionNode subscription = (OpcSubscriptionNode) node;

            // Test if the subscription is enabled, and call the appropriate
            // method.
            if (subscription.Enabled)
            {
                if (subscription is OpcAeSubscriptionNode)
                    CreateSubscriptionToAeServer(server,
                        (OpcAeSubscriptionNode) subscription);
                if (subscription is OpcDaSubscriptionNode)
                    CreateSubscriptionToDaServer(server,
                        (OpcDaSubscriptionNode) subscription);
            }
        }
    }
}
```

The code above expects one of two classes, namely `OpcAeSubscriptionNode` and `OpcDaSubscriptionNode`. However, information such as whether or not the subscription is enabled, could easily be extracted into a superclass. In the example, this superclass is `OpcSubscriptionNode` which contains all the common information. This allows the inner `foreach` loop to treat both subscriptions in the same manner when testing whether or not the subscription is enabled. If this is the case, the subscriptions is tested to figure out which concrete class it represents, and the appropriate method is called. The line where the

`TreeNode` is converted to an `OpcSubscriptionNode` is required because the servers and associated subscriptions are stored in a tree structure which requires the `OpcSubscriptionNode` to extend the `TreeNode` class thereby making the type cast possible.

7.5 Implemented Design Patterns

The Template Method Design Patterns is implemented with the following code:

```
/// <summary>
/// An abstract Listener performing all the necessary setups. Subclasses should
/// primarily override the hook GetFrontEnd(). The Back End can naturally be
/// changed by overriding the hook GetBackEnd(). If this is not overridden the
/// DefaultBackEnd is used.
/// </summary>
public abstract class AbstractListener
{
    ...

    /// <summary>
    /// Reference to the Back End
    /// </summary>
    IBackEnd m_backEnd;

    /// <summary>
    /// Reference to the Front End
    /// </summary>
    IFrontEnd m_frontEnd;

    /// <summary>
    /// Default constructor initializing the APP_HOME and the Front and Back
    /// ends
    /// </summary>
    public AbstractListener()
    {
        ...

        // Create Front/Back-Ends
        m_backEnd = GetBackEnd();
        m_frontEnd = GetFrontEnd();
    }

    /// <summary>
```

```
/// The body of the listener
///
/// Initilizes Front and Back End and starts the Front End
/// </summary>
public void Start()
{
    m_backEnd.Initialize();
    m_frontEnd.Initialize();
    m_frontEnd.Start();
}

/// <summary>
/// Method which terminates the listener. This termines the Front
/// and Back End.
/// </summary>
public void Terminate()
{
    m_frontEnd.Terminate();
    m_backEnd.Terminate();
    m_backEnd = null;
    m_frontEnd = null;
}

/// <summary>
/// Method used to pass an event from the Front End to the Back End
/// </summary>
/// <param name="theEvent">The event to pass to the Back End</param>
public void FireEvent(IEvent theEvent)
{
    m_backEnd.FireEvent(theEvent);
}

/// <summary>
/// Hook yielding the Back End to use
/// </summary>
/// <returns>The Back End to use</returns>
protected virtual IBackEnd GetBackEnd()
{
    return new DefaultBackEnd();
}

/// <summary>
```

```
    /// Hook yielding the Front End to use
    /// </summary>
    /// <returns>The Front End to use</returns>
    protected abstract IFrontEnd GetFrontEnd();
}
```

The code above is the code for the abstract class. To "complete" the pattern, another class, namely the concrete class should also be included. This could be the class `OpcListener` for example. This class has one sole purpose: to create an `OpcFrontEnd`, and possibly a `OpcBackEnd` if special back-end behavior is required. In this example, the algorithm of the abstract class lies in the `Start()`, `FireEvent(...)`, and `Terminate()` methods.

The Singleton Design Pattern is implemented with the following code:

```
public class PersistentListener : CoreListener
{

    ...

    /// <summary>
    /// Default constructor calling the constructor of the base class
    /// </summary>
    private PersistentListener() : base() {}

    /// <summary>
    /// Static method yielding a reference to the listener (singleton)
    /// </summary>
    /// <returns>Reference to the listener instance</returns>
    public static CoreListener GetInstance()
    {
        if (m_instance == null)
        {
            m_instance = new PersistentListener();
        }
        return m_instance;
    }

    ...

}
```

Making the constructor private ensures that only the class itself can instantiate an instance. To fetch a reference to the object, clients have to call the static method `GetInstance()` which creates an instance if it does not already exists, and returns it. It is worth noting that the variable `m_instance` is an instance variable declared in the superclass `CoreListener`.

8 Test

Testing is an important step in the process of software development. One type of test allows the developer(s) to identify which parts of the implementation do not work as expected, while another type can show a customer that the product works as required. These types of tests differ in that one is successful if and only if it finds errors in the implementation, while the other is successful if and only if no errors occur. There are several standard test types, each with its ups and downs, this section will mention a few of these types and explain the test plan for this project in details.

8.1 Test Procedures

A *Release Test* is a test where a given system is tested by feeding it some input and verifying that the generated output match the expected output. In such a test, it is verified that a given action is carried out without being concerned with the details of how it was carried out. This is also known as the *Black-box principle*. If customers are involved in the process, the test is sometimes referred to as *Acceptance Test*. The opposite of the *Black-box principle* is the *White-box principle* which does go into details with how every little action took place. When following the *White-box* principle it is common to divide a system into components, each being tested individually. This technique, known as *Component Testing* or *Unit Testing*, makes it a lot easier to locate errors, and correct them compared to *Black-box* testing. The method is commonly down to the level of testing methods/functions to verify that they return the expected values or perform the expected tasks on a given input. This technique makes it very simple to locate errors and correct them, however it is a rather time consuming procedure. Another type of test that can be carried out is *Integration Testing* which involves testing the integration of 2 or more components. When performing such a test, it is crucial to use an incremental approach, otherwise identifying where an error occurred can be hard. Incremental approach means that you take the simplest combination first (possibly a single component) and test that it works as expected. You then add a new component, and a test for that. You first run the tests which you have already seen work for the single component alone to verify that they still work before moving on to running the new tests. In this way it is easier to identify where the error was introduced.

For the OPC Listener, it was decided that a *Release Test*, combined with a few *Integration Tests* would suffice. Reasons for choosing this type of test, and not *Component Testing*

along with other types of more detailed test include: 1) Zonith A/S have a standard way of writing all their Test Specifications so it was a natural decision to maintain this standard, 2) The OPC Listener is not a product requested by a customer, it is rather a product aimed at potential customers, and thus an *Acceptance Test* is hard to carry out, and 3) The time it takes to carry out a very detailed test is considered too expensive since the product is not a critical system with regards to minor flaws. It is Zonith A/S's opinion that solving minor problems once they are found (possibly by customers) is the most cost-effective solution for their products. The *Release Test* created for the OPC Listener covers the basic aspects such as installing/uninstalling, configuration of the product, as well as a test of the User Interface.

8.2 Test Plan

As mentioned above, the OPC Listener will be tested using a *Release Test*, however the system is not a just a box which you can feed with some input and receive some output. In order to test the system thoroughly some parts are tested separately. The following sections will describe what test suites are included in the Test Plan which can be found in Appendix F.

The test are carried out using two different OPC Servers: 1) Schneider Electric OPC Factory Server (OFS) V3.1, and 2) Signalix MT-101. Schneider Electric OFS can be installed on a PC and the values manipulated using any OPC browser application. Signalix MT-101 is a physical unit which needs a GPRS modem connection in order for an application to receive values.
To manipulate values on the two servers, Softing OPC Toolbox Demo Client, available at http://www.softing.com/en/communications/downloads/demos/opc/demo_client.htm, was used.

8.2.1 Installation

This test suite verifies that the OPC Listener can be installed correctly on a Microsoft platform. It states some prerequisites, and thereafter tests the installation of the software in two ways: 1) Installing to a path which does not include white spaces, and 2) Installing to a path which does include white spaces.

8.2.2 OPC Server to OPC Listener Communication

This test consists of two suites, one for the A&E interface and one for the DA interface. The purpose of the test is to ensure that alarms/events generated in the OPC Server are actually

intercepted by the OPC Listener. In order to verify this, one is required to inspect log files, and optionally perform a few "SELECT" statements in the temporary storage database.

8.2.3 OPC Listener to ACS Communication

As above, this test consists of a test suite for each of the two interfaces. The purpose of the test is to ensure that alarms/events available in the temporary storage database, are actually delivered to the ACS. In order to verify this, one is required to login to the ACS and list incoming alarms. In case alarms are not displayed in the ACS, one can inspect the log files to see if the alarms/events indeed where received, but merely had non-existing remote unit id's or the like, which consequently causes the alarms to be "ignored". By "ignored" is meant: stored in the database and log files for reporting purposes, but not available for dispatching.

8.2.4 OPC Server to ACS Communication via OPC Listener

As above, this test consists of a test suite for each of the two interfaces. The purpose of this test, is to combine the test suites mentioned above, and verify that an alarm/event generated in the OPC Server actually appears in the ACS. The test is included in order to ensure that the OPC Listener delivers the alarm/event on the fly, and does not require a restart. To verify that the communication takes place, one follows the same verification procedures as in the two previous tests.

8.2.5 Distributed Systems

This test suite verifies that the system can be executed in a distributed environment. By distributed environment is meant: OPC Listener installed on one computer in a network (domain), the OPC Server installed on another computer in the same network (domain), and the ACS installed on any network reachable via Ethernet with a given IP address. This test suite is included due to the common fact that running OPC systems in a distributed environment is a often error prone. Basically the test consists of the test suites described above, executed in a distributed environment.

8.2.6 OPC Listener Configuration Tool

This test suite verifies that the OPC Listener Configuration Tool functions correctly. It guides the tester through creation of all the common elements available in the tool, and

verifies the correctness of the XML configuration file generated once the configuration is saved.

8.2.7 Uninstalling

This test suite verifies that the OPC Listener can be correctly uninstalled, causing the Windows Service to be removed, and files to be removed from the installation folder. Remains in the installation folder will be files such as configuration files created by the user, and log files generated by the OPC Listener.

9 Discussion

During the design and development of the OPC Listener, a number of good and not so good things surfaced. This section discusses these issues and identifies possible extensions/corrections for future implementation.

9.1 The Good Things

Initially this project was meant to cover only the Alarms&Events interface, however, after a number of discussions with several people involved in the world of OPC, it was clear that the "common way of doing things" did not involve the Alarms&Events interface. The reason for this is mainly that the Alarms&Events interface is rather new, and OPC solutions have been implemented over a period of time using the Data Access interface. Therefore existing solutions, for the most part, rely on the Data Access interface. Once this became clear to me, I decided to extend the project to comply to the Data Access interface as well as the Alarms&Events interface, thereby ensuring compatibility with most existing solutions.

While translating Zonith A/S's default back-end implementation, I discovered a small error which could cause a thread to "hang". This "bug" was not found since the original back-end is written in Java, and stopping the Windows Service simply terminates the JVM which in turn destroys all running threads, thus not leaving it hanging. This however is not the case when writing the Windows Service in C#, so the code had to be modified to fix this problem. The good thing about discovering this problem is that it is now fixed in both the Java version and the C# version thereby eliminating any future errors this might have caused.

9.2 The Bad Things

OPC has long been a technology primarily available to products running on a Microsoft platform. This however is about to change in the near future, as XML/SOAP based solutions are in the making, covering most of the interfaces defined by the OPC Foundation [6]. The OPC Listener developed in this project, is written in C# for the .NET Framework which, for now, is only available for Microsoft platforms. Although it is a bad thing to limit the product to one platform, it is not much of a drawback just yet as nearly all OPC systems are currently running on Microsoft platforms.

9.3 Possible Extensions

In the subsection above the limitation of the OPC Listener to only work with Microsoft solutions is mentioned. The OPC Listener is generically designed, allowing for easy implementation of XML/SOAP based OPC interaction. This makes the OPC Listener more generic in the sense that it can communicate with OPC Servers running on any platform, although it, itself, is still restricted to run on a Microsoft platform.

As of now, configuration files do not conform to an XML Schema or the like, they simply have a predefined structure which they must follow. The check is implemented by simply processing the XML file looking for the expected elements, and errors are generated if they do not appear. In the future the configuration files should conform to a XML Schema thereby allowing for easy validation of the XML file, and easier ways of parsing the information held in the configuration file.

The testing of the product is currently only done using a *Release Test*, however a good extension would be to introduce automatic *Unit Testing* which would minimize the amount of time spent locating minor errors. This could possibly be done using *NUnit* which is unit-testing framework for all .NET languages. *NUnit* is initially ported from JUnit known from the Java world.

10 Conclusion

This project has resulted in the creation of a new Listener, OPC Listener, which Zonith A/S can sell alongside their existing Alarm Control System (ACS). Furthermore, a report documenting the analysis, design and implementation of the OPC Listener was created, and analysis incorporating the fields of Domain Modelling and Formal Specification was included. The Domain Modelling comprised the complete domain in which the OPC Listener exists, although only the OPC Listener part itself was modelled into fine details. The Formal Specification was used to analyze and structurally design the process of connecting to OPC Servers, as well as the process of storing/persisting alarms locally until they with certainty have been received by the ACS.

In order to make the product sellable, a User's Guide was written as well (see Appendix A). This helps users through the installation process as well as configuring the OPC Listener to handle alarms/events from desired OPC Servers.

The project has taught me a great deal about the .NET Framework, and given me some insight into what capabilities it has. Working with the world of OPC has given me some insight which can be useful in many future projects/jobs. Problems arising from running distributed OPC server/client systems, have given me a fair amount of knowledge on setting up DCOM Configuration in Windows, and opening the OPC Specific ports on firewalls, etc.

Zonith A/S was very pleased with the project, and the final product, and as a result have offered me a position as a Software Engineer in the company. A position which I have gladly accepted.

As a whole, the project has been very educational and interesting. I am very pleased with the result, and look forward to new challenges as a Zonith A/S employee.

References

[1] **Larman, Craig** (2002) - Applying UML and Patterns : *an introduction to object-oriented analysis and design and the Unified Process*

[2] **Fowler, Martin** (2000) - UML Distilled : *a brief guide to the standard object modelling language*

[3] **Kruchten, Philippe** (2000) - The Rational Unified Process : *an introduction - Second Edition*

[4] **The RAISE Language Group** (1992) - The RAISE Specification Language

[5] **Hejlsberg, Anders** (2004) - The C# Programming Language

[6] **OPC Foundation** - http://www.opcfoundation.org

[7] **Numerous Authors** (2005) - UML OCL2 Specification - http://www.omg.org/docs/ptc/05-06-06.pdf

[8] **Ian Sommerville** (2004) - Software Engineering - Seventh Edition

Appendices

Appendix A - Users Guide: The Users Guide is a document meant to ease the installation process and guide end users through the different setup possibilities available in the OPC Listener.

Appendix B - Time Table: The Time Table documents how the time was spent, and what milestones etc. were included in the process.

Appendix C - Use Case Screenshots: The Use Case Screenshots are used to aid the reader get a better understanding of the Use Case Descriptions.

Appendix D - RSL Specifications: The RSL Specifications documents the developments of the Formal Specification of the selected areas of the project. These specifications each go through several iterations to get to the end result. The report only includes the first and last specification, while the iterations leading from one to the other are placed in this Appendix.

Appendix E - Coding Convention: The Coding Convention documents the style which is used in the coding of the product.

Appendix F - Test Specification: The Test Specification documents the End-to-end test performed on the system.

Appendix G - Broad Scope Diagram: The Broad scope diagram is a diagram showing how the ACS is connected to several types of networks through various listeners.

Appendix H - Source Code: The Source Code is printouts of the C# source which encompass the OPC Listener and the Windows Service Framework.

OPC Listener for ACS

User's Guide

Version 1.0.1

ZONITH
Gammel Kongevej 39E,1 DK-1610 Copenhagen V
www.zonith.com tlf +45 33 32 45 30 info@zonith.com

Version 1.0.1
(01-09-2005)

CONTENTS

1 INTRODUCTION

The OPC Listener for Alarm Control System (ACS) provides the ability to receive alarms from OPC compliant networks via the standard OPC interfaces defined by the OPC Foundation.

For the OPC Listener to function the following is required:

- A running version of the Alarm Control System (ACS) software

This document will guide you through the installation, configuration and usage of the OPC Listener.

1.1 Alarm Delivery

When the OPC Listener receives an Event/Alarm from an OPC Server (within an OPC compliant network) it will verify that the Event/Alarm is of interest, and propagate it into the ACS Server. The alarm will be visible in the list of ongoing alarms.

If the listener receives alarms while the ACS server is taken down for maintenance the alarms will not be lost. The listener collects the alarms and will try to propagate them to ACS continuously. When ACS is in service again the alarms will be delivered as normal. If the listener is stopped while trying to propagate alarms to ACS it will persists the alarms it has collected before shutting down. Of course no alarms will be delivered to ACS as long as the listener is stopped and it will not be able to collect alarms as well. When the listener is started it will load the persisted alarms and start delivering them to ACS. Thus, it is important that the listener is installed as a service and runs all the time. This provides the best alarm delivery guarantee.

1.2 System Requirements

The OPC Listener runs on the following platforms:

- Microsoft Windows 2003 Server
- Microsoft Windows 2000 Server
- Microsoft Windows XP SP1 and SP2

It is required that the .NET Compact Framework is installed, as well as OPC Core Components.

2 INSTALLATION

2.1 Prerequisites

Before installing the OPC Listener, you need to make sure you have the .NET Compact Framework as well as the OPC Core Components installed on the system.

2.1.1 Installing .NET Compact Framework

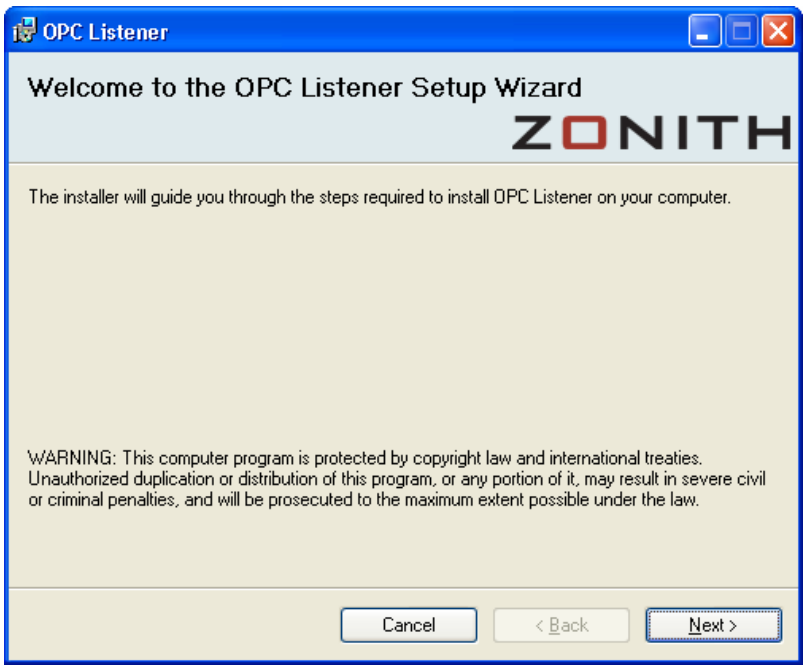
Run the 'NETCFSetup.msi' installation file available on the CD-ROM and follow the instructions.

2.1.2 Installing the OPC Core Components

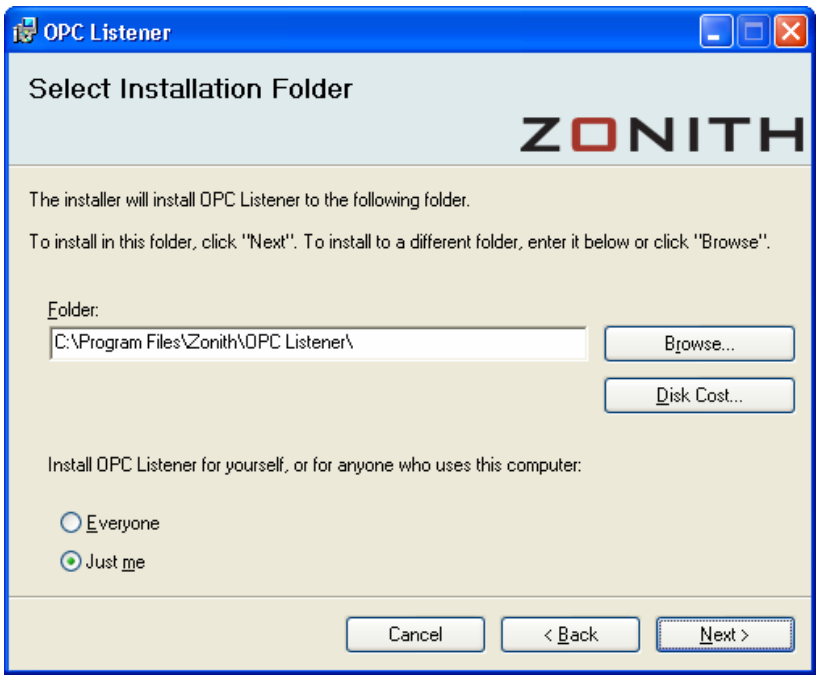
Run the 'OPC Core Components 2.00 Redistributable 2.20.msi' installation file available on the CD-ROM and follow the instructions.

2.2 Installing the OPC Listener

When running the 'Setup.exe' of the OPC Listener distribution set you are presented with a standard installer program that takes care of installing the software in an easy and user-friendly way:

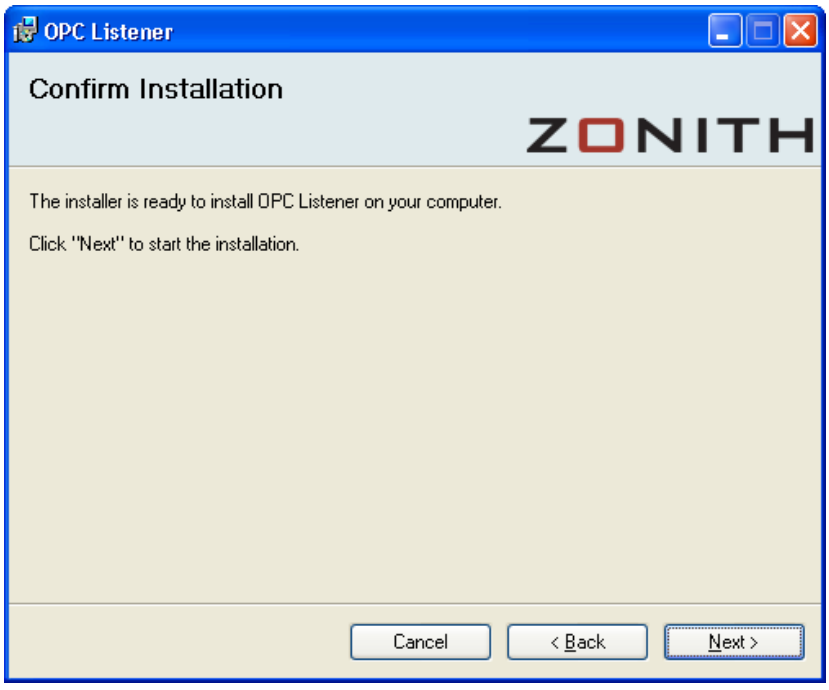


When the license agreement has been accepted you are presented with the proposed installation path for the software:



The installation-path can be modified, but please refer from using any special characters like æ, ø and å. Please only use US characters only.

When the installation directory has been selected you are presented with a screen allowing you to install the software



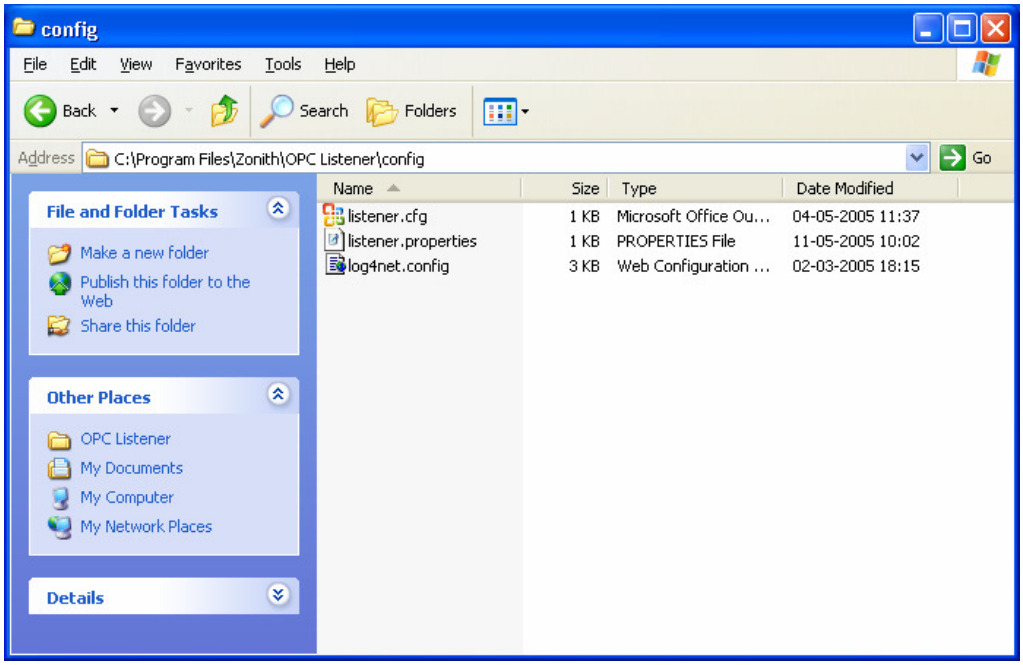
When clicking the **Next** button all required files are copied to the installation directory and you are asked for a username/password combination which the OPC Listener Windows Service will run as. This **must** be a username which is a member of the domain where the OPC Server is running. The username is appended by the domain name such that a user, XY, in the domain Z, would be written as XY@Z. Enter such a username/password combination and click **Ok**. The software is not running at this point.

When the installation program finishes, the OPC Listener directory hierarchy should contain the following directories:

Directory	Description
`config`	Contains configuration files.
`languages`	Contains database files.
`doc`	Contains User's Guide.
`logs`	Contains log files. This does not exist after a new installation; it is created once a log entry is generated.
`storage`	Contains the binary data files with persisted alarms. This does not exist after a new installation; it is created once a log entry is generated.

3 CONFIGURATION

The `<install-dir>\config\` directory contains all the OPC Listener property files:



If the listener is installed on the same server that is running ACS, the configuration of the listener works out of the box, but otherwise you will need to edit a single file in this directory, namely `listener.properties`.

The content of this file is shown in the figure below:

```
# Properties for the listener framework, configuring the settings
for receiving alarms

listener.name=ACS OPC Listener
listener.companyNo=1
listener.acs.endPoint=http://localhost:8080/acs/services/AlarmServ
ice
listener.storage=storage\my_hsql_storage
```

Here you can change different settings for the OPC Listener, but the only line that typically should be changed to configure the system is:

- **`listener.acs.endpoint`**

  Should point to the server running ACS

4 LOG FACILITY

The listener provides two logging facilities: a message log and a system log. The message log is used for logging status on sending the received alarms to ACS. The message log is file based and logs only failures by default.

The system log is used for logging program warnings and errors. The system log both appends program warnings and errors to the Windows Event Log System and writes them to file. Logging to the Windows Event Log System is convenient for monitoring purposes, whereas the file based log is convenient for offline analysis.

Both file based message and system logs are so-called rolling logs, meaning that the underlying file is rolled over at a certain frequency. A new file is created to preserve the log history. The rolling schedule for both logs is set to rollover at the beginning of each month. For example, at midnight of October 31st, 2003 `<install-dir>\logs\messagelog.log` will be copied to `<install-dir>\logs\messagelog.log.2004-10`. Logging for the month of November will be output to `<install-dir>\logs\messagelog.log` until it is also rolled over the next month.

The logging configuration file is located at:

`<install-dir>\config\log4net.config`

Note: Changes to the `log4net.config` file will only take effect after the Listener has been (re)started.

4.1 Message Log

The listener will always log to `<install-dir>\logs\messagelog.log` if it cannot deliver an alarm to ACS. It is recommended that you monitor this file.

The message log is always enabled and logs only delivery failures by default. It is also possible to log when the listener delivers alarms successfully to ACS. However, it is not recommended to use this option permanently. To enable logging of all alarm delivery attempts to ACS change the following in `log4net.config`:

```
<logger name="MessageLog">
      <level value="ERROR" />
      <appender-ref ref="MessageLogAppender" />
</logger>
```

to

```
<logger name="MessageLog">
      <level value="ALL" />
      <appender-ref ref="MessageLogAppender" />
</logger>
```

Remember that the message log will grow faster when it logs all alarm delivery attempts.

4.2 Windows Application Event Log

The listener appends program warnings and errors to the Windows Application Event Log for monitoring purposes. Entries are identified by the source name 'ACS OPC Listener'.

4.3 System Log

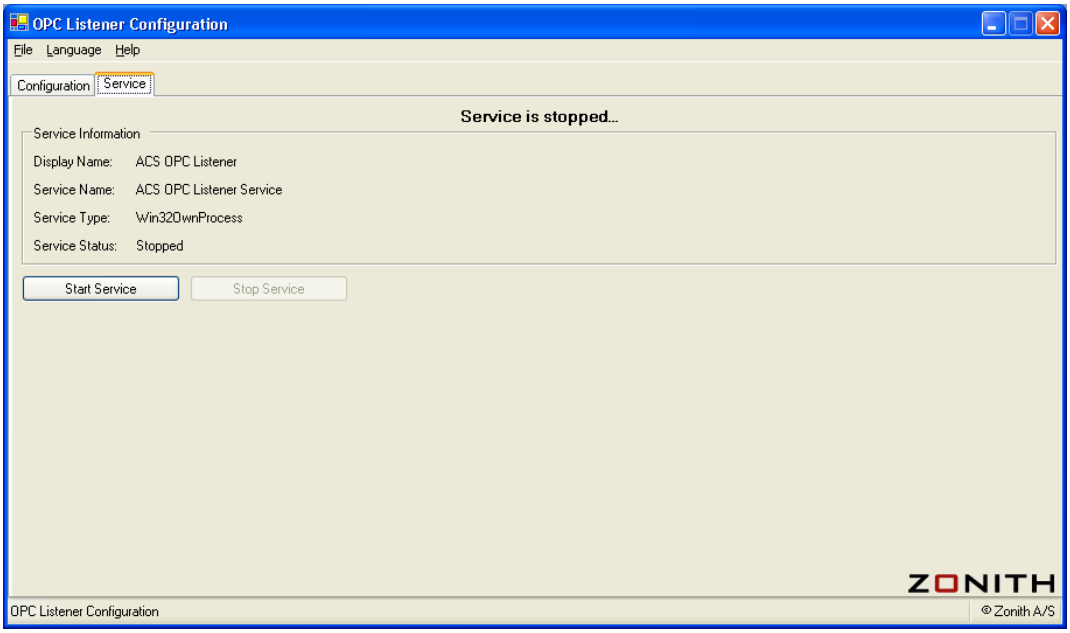
The listener will log program errors including invalid configuration and network failures to the file `<install-dir>\logs\systemlog.log`. See Appendix A for details on the log file format.

5 RUNNING THE LISTENER

The `<install-dir>` directory contains executable programs. There are 2 executables in the directory: 'OPC Configuration.exe' which is the OPC Listener Configuration Tool also reachable via the Programs Menu, and 'Zonith.Listener.OpcListener.exe' which is the executable executed as a service. The latter of the two cannot be started as a stand alone program, thus the OPC Listener **must** be executed as a Windows Service.

5.1 Running OPC Listener

The easiest way of starting the OPC Listener is via the OPC Listener Configuration Tool. You simply select the **Service** tab and click **Start Service**.



The ACS OPC Listener Service can also be started and stopped from a command line using the following commands:

```
net start "ACS OPC Listener Service"
net stop "ACS OPC Listener Service"
```

Starting the OPC Listener has no effect other than initializing the front and back ends of the system unless a configuration file is specified. This configuration file is created using the OPC Listener Configuration Tool, allowing users to select which alarms should be intercepted, etc. For a complete guide on creating configuration files, please see Section 6 of the User's Guide.

The configuration file must be named `listener.cfg` and must be located in `<install-dir>\config\`.

Note: Configuration files can be saved anywhere for later use, but only the one located at `<install-dir>\config\listener.cfg` is used by the OPC Listener. Once the OPC Listener is started (by clicking **Start Service**), the OPC Listener is running, and the OPC Listener Configuration Tool can be shut down.

6 OPC LISTENER CONFIGURATION TOOL

Configuration files created using the OPC Listener Configuration Tool are used to tell the OPC Listener which alarms should be intercepted and propagated to the ACS. This section will guide you through the setup of such a configuration file.

6.1 Menus

File Menu

The File Menu allows the user to create new configurations as well as save and open existing ones. It also has the option of exiting the OPC Listener Configuration Tool.

Language Menu

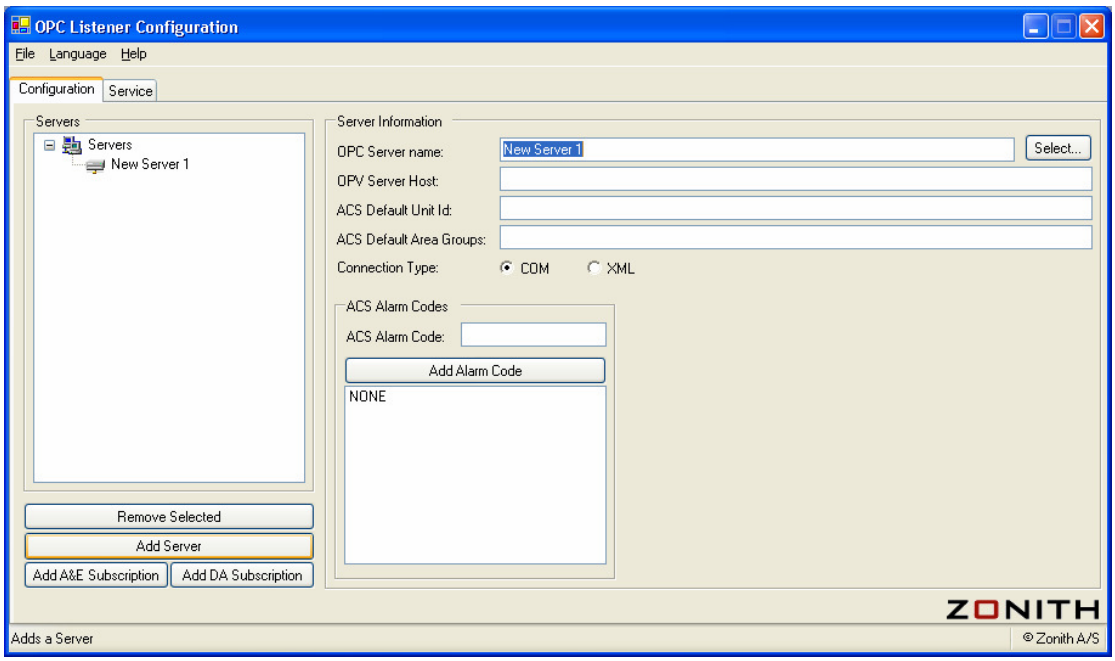
The Language menu allows the user to switch between different languages affecting the look of the OPC Listener Configuration Tool.

Help Menu

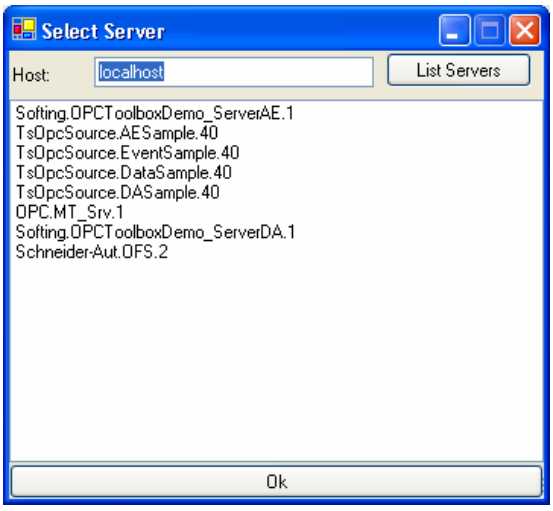
The Help menu allows the user to see an '**About**' dialog displaying information about the OPC Listener Configuration Tool.

6.2 Adding a connection to an OPC Server

To add a new connection to an OPC Server to the configuration, you click on **Add Server**. This will adds a new server element in the list of servers, and you now have the possibility of configuring a few things for the server.



Unless you know the exact name of the OPC Server you want to connect to, click **Select…** and choose the desired server from the list of servers. If you want to connect to a remote server, you must enter the DNS name of the server or its IP address.



Default Unit Id

The Default Unit Id which you want to use for this server. This Unit Id must be exactly the same as the one you have defined in the ACS (the Remote Unit Id) in order to display alarms correctly in the ACS. The reason this is called the *default* Remote Unit Id is because each subscription created on this server have the possibility of overriding this value if needed.

Default Area Groups

The Default Area Groups which like the Default Unit Id have to be exact matches of Area Groups defined in the ACS in order to function correctly.

Connection Type

The Connection Type can be set to either COM or XML, but version 1.0.1 only supports COM servers. Please contact Zonith A/S for information on when to expect XML support.

6.2.1 ACS Alarm Codes

ACS Alarm Code

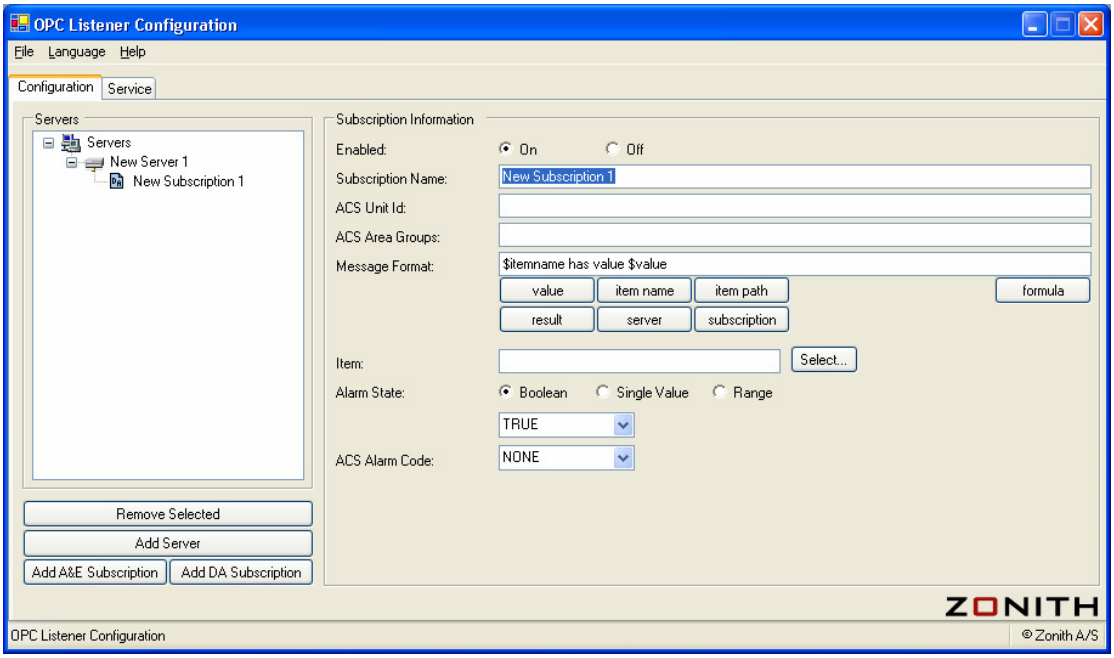
ACS Alarm Codes can be created to match any Alarm Codes (Code Mappings on Remote Units) created in the ACS. Values entered in this list should match the values named *Code* in the Code Mappings for a Remote Unit in the ACS.

Add Alarm Code

This button causes the Alarm Code entered above to be added to the list of ACS Alarm Codes. To delete an Alarm Code, select it in the list and hit the delete key.

6.3 Adding a Data Access subscription to the server

To add a Data Access subscription to a server you select the server in the list of servers, and click **Add DA Subscription**. This will add the subscription element to the list of servers, and you now have the possibility of configuring the alarms you want to intercept.



Enabled/Disabled

This option allows you to select whether the subscription should be enabled or disabled. Enabled subscriptions intercept the alarms defined while disabled subscriptions do not. The reason for having such an option is to be able to keep subscriptions in the configuration although they might not be used at the moment.

Subscription Name

The Subscription Name is an arbitrary, but unique, name of the subscription. This can be used to give the subscription a meaningful name describing which

alarms are intercepted. An example of such a name could be ‘Water level too low subscription’ or the like.

Unit Id and Area Groups

The Unit Id and Area Groups allow you to override the default Unit Id and Area Groups defined in the server. The Unit Id should only be changed if the alarms should appear as coming from a different Remote Unit, while changing the values of the Area Groups enables the user to define very specific competencies or geographic locations required for each alarm.

Message Format

The Message Format is the actual content of the alarm message sent to the ACS. The value of this field can contain static text which the user enters him-/herself, or it can contain dynamic content which is added upon reception of an alarm. The dynamic content for Data Access subscriptions can be one of the following:

- value
  - The value read from the selected item. This can either be a number or a Boolean value.
- item name
  - The name of the selected item in the OPC Server.
- item path
  - The path of the selected item in the OPC Server.
- result
  - A result indicating whether the *fetch value* operation was successfully executed.
- server
  - The name of the OPC server which the item is located on.
- subscription
  - The subscription name defined above.
- formula
  - Possibility of adding a mathematical formula. This is primarily used to scale values to stay within a given range, but could be used for other things as well. An example of how to use the formula element is shown here: ‘$formula($value * 2)’ which will calculate the value of the item multiplied by 2. It is a good idea to use parentheses often in order to obtain the correct computation. It is important to note that all results are in **radians**.
  - The mathematical functions which are available for use within the $formula(…) element are:
    - + [Addition]
      - Example: 1+2 which yields 3
    - - [Subtraction]
      - Example: 6-2 which yields 4
    - * [Multiplication]
      - Example: 2*4 which yields 8
    - / [Division]
      - Example: 14/2 which yields 7
    - ^ [Power]

        - Example: 4^2 which yields 16
      - & [Root]
        - Example: 2&4 which yields 2 (better known as the square root of 4)
        - Example: 3&4 which yields 1.587 (better known as the third/cube root of 4)
      - sin [Sine]
        - Example: sin0.5 which yields 0.479
      - cos [Cosine]
        - Example: cos0.5 which yields 0.878
      - tan [Tangent]
        - Example: tan0.5 which yields 0.546
      - Asin [Arc Sine]
        - Example: Asin0.5 which yields 0.524
      - Acos [Arc Cosine]
        - Example: Acos0.5 which yields 1.047
      - Atan [Arc Tangent]
        - Example: Atan0.5 which yields 0.464
      - sinh [Hyperbolic Sine]
        - Example: sinh0.5 which yields 0.521
      - cosh [Hyperbolic Cosine]
        - Example: cosh0.5 which yields 1.128
      - tanh [Hyperbolic Tangent]
        - Example: tanh0.5 which yields 0.462
      - log [Logarithm Base 10]
        - Example: log3 which yields 0.477
      - ln [Natural Logarithm]
        - Example: ln3 which yields 1.099
      - ! [Factorial]
        - Example: 5! Which yields 120
    - Furthermore the following constants can be used:
      - pi
        - Corresponding to the value 3.1415926…
      - e
        - Corresponding to the value 2.7182818…
    - To ease the writing of very small decimal numbers one can also use ‘Big E notation’. This means that you can write 25E-5 instead of 0.00025.

Item

The Item is the item from the OPC Server which is surveyed for changes. Unless you know the exact name of the desired item, you should click **Select…** and choose the desired item from the list.

Alarm State

The Alarm State indicates which type of alarm should be intercepted. The way an alarm is identified is by looking at the value, and checking whether it falls within the rules defined in here. There are the following three options:

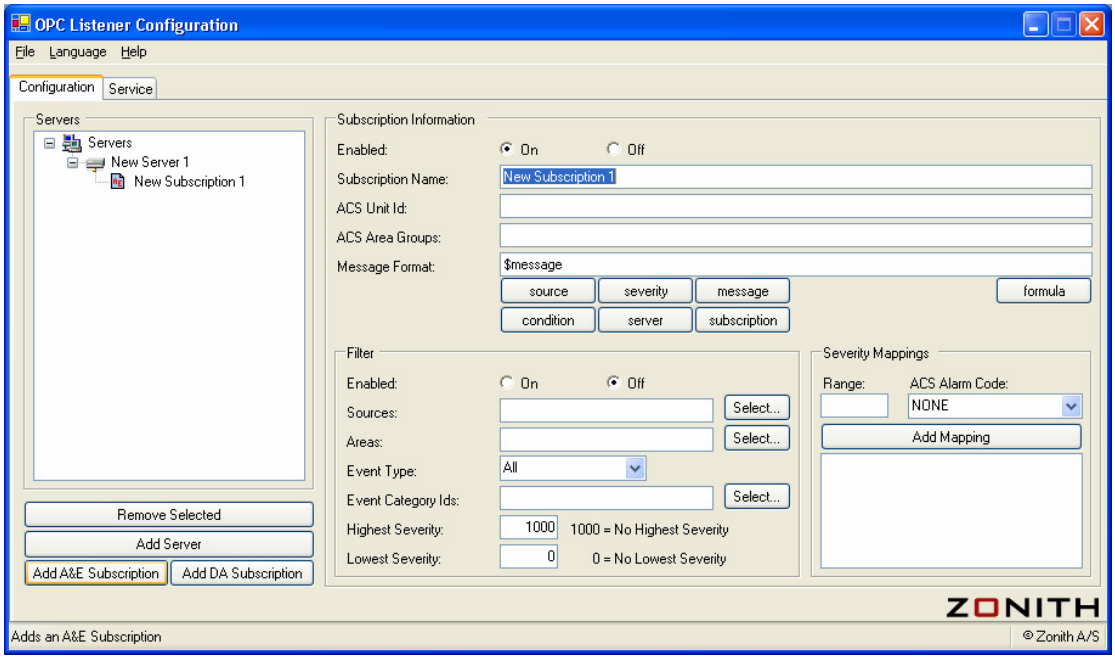
- Boolean
  - The value of the Item will be either true or false (or 1 or 0) where only one of them will indicate an alarm.
- Single Value
  - The value of the item can be any number, but only if the value is exactly the same as the number specified here is it considered an alarm.
- Range
  - The value of the item can be any number, but only if the value falls between the lower and higher limit of the range specified here is it considered an alarm.

ACS Alarm Code

The ACS Alarm Code is what corresponds to Code Mappings on Remote Units in the ACS. If you have defined a special Code Mapping in the ACS which you want to use for this alarm, you must define an ACS Alarm Code with the same Code as used in the ACS. This ACS Alarm Code needs to be specified for the server which this subscription belongs to in order for it to appear in the select box.

6.4 Adding an Alarms & Events subscription to the server

To add an Alarms & Events subscription to a server you select the server in the list of servers, and click **Add A&E Subscription**. This will add the subscription element to the list of servers, and you now have the possibility of configuring the alarms you want to intercept.



Enabled/Disabled

This is the same as for Data Access subscriptions.

Subscription Name

This is the same as for Data Access subscriptions.

Unit Id and Area Groups

This is the same as for Data Access subscriptions.

Message Format

The Message Format is the actual content of the alarm message sent to the ACS. The value of this field can contain static text which the user enters him-/herself, or it can contain dynamic content which is added upon reception of an alarm. The dynamic content for Data Access subscriptions can be one of the following:

- Source
  - The name of the source where the alarm emerged from.
- Severity
  - The severity of the alarm. This is a number between 0 and 1000.
- Message
  - The message associated with the alarm. This message originates from the OPC Server.
- Condition
  - The condition which caused this alarm to be generated.
- Server
  - The name of the OPC server which the source is located on.
- Subscription
  - The subscription name defined above.
- Formula
  - The same as for Data Access subscriptions.

6.4.1 Filter

Enabled/Disabled

This option allows you to select whether the filter should be enabled or disabled. Enabling the filter causes the subscription to merely intercept the alarm types defined in the filter, while disabling it causes the subscription to intercept all alarms.

Sources

This is the list of sources from the OPC server which should be surveyed. Leaving this field empty is the same as selecting all sources. When selecting multiple sources, these should be separated by commas. It is recommended that you use the **Select…** button to select the desired sources.

Areas

This is the list of areas within the OPC Server which should be surveyed. Surveying an area causes the subscription to intercept all alarms from sources located in that area unless otherwise specified in the sources field above. When selecting multiple areas, these should be separated by commas. It is recommended that you use the **Select…** button to select the desired areas.

Event Type

This is the type of events which should be intercepted.

Event Category Ids

This is the list of event category ids which should be surveyed. Surveying an event category id causes the subscription to only intercept the alarms which are associated with this event category id. When selecting multiple event category ids, these should be separated by commas. It is recommended that you use the **Select…** button to select the desired event category ids, as this also gives you the possibility to see their display names instead of the raw numbers.

Highest Severity

The highest severity allows you to put an upper limit on which alarms you want to intercept based on their severity values. This number should always be greater than or equal to the lowest severity. Alarms with severities higher than the value defined here will not be intercepted. 1000 is the maximum severity an alarm can have, and thus entering this value causes the subscription to intercept all alarms (with severities greater than or equal to the lowest severity).

Lowest Severity

The lowest severity allows you to put a lower limit on which alarms you want to intercept based on their severity values. This number should always be less than or equal to the highest severity. Alarms with severities lower than the value defined here will not be intercepted. 0 is the minimum severity an alarm can have, and thus entering this value cases the subscription to intercept all alarms (with severities less than or equal to the highest severity).

6.4.2 Severity Mappings

Range

Here you can define a range of severities which should be mapped to the ACS Alarm Code selected in the next field. This value can be one of the following:

- Single value
  - A single number between 0 and 1000
- Range
  - Two numbers between 0 and 1000 separated by a dash (-). The first number must be less than the second number.

ACS Alarm Code

This is the ACS Alarm code which should be mapped to. If you have an ACS Alarm Code by the name of ‘LOW’, this could possibly cover the range of severities ranging from 0 to 200. Thus you would enter ‘0-200’ as the range and select LOW as the ACS Alarm Code.

Add Mapping

This button causes the mapping defined above to be added to the list of Severity Mappings. To delete a severity mapping, select it in the list and hit the delete key.

APPENDIX A - LOG FILE FORMAT

This appendix describes the log message format used by the listener system log file.

The first line of each message begins with ##### followed by the message header. The message header provides the run-time context of the message. Each attribute of the message is contained between square brackets. Lines following the message body are only present for messages logging an exception and display the stack trace for the exception.

The general format of a log message is as follows:

```
#####[Timestamp] [Severity] [Location] [Product] [ThreadID]
[MessageText]
```

The following is an example of a log message:

```
#####[2004-06-23 15:37:41,025] [FATAL]
[Zonith.Listener.AbstractService.OnStart(:0)] [OPC Listener]
[3068] [Could not start OPC Listener Service]
```

Note: The character encoding used in writing the system log file is the default character encoding of the host system.

Message Attributes

Each log message saved in the system log file contains the attributes listed in the following table:

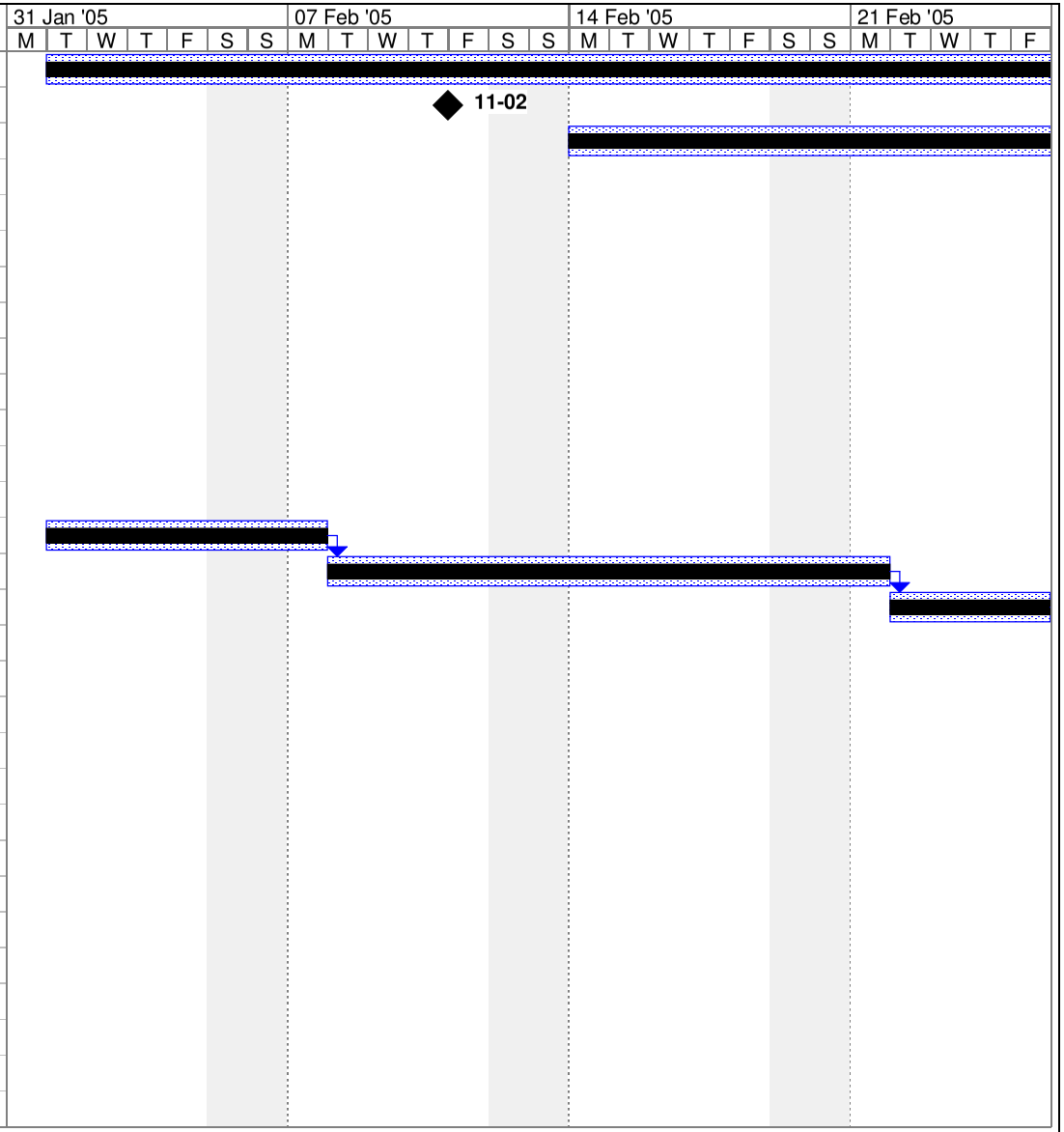
Attribute	Description
Timestamp	The time and date when the message originated, in a format that is specific to the locale.
Severity	Indicates the degree of impact or seriousness of the alarm reported by the message. See Message Severity below.
Location	Location in the code where the message originates from
Product	This attribute denotes the product that was the source of the message. In this case, OPC Listener.
ThreadID	Identify the thread that originated the message.
MessageText	The actual message text defined by the developer of the program.

Message Severity

The listener log messages have an attribute called severity that reflects the importance or potential impact on usage of the system. The defined severities are listed below in order of severity with Fatal being the highest severity:

Severity	Meaning
Fatal	The server is in an unusable state. This severity indicates a severe system failure and restarting the server is the only solution.
Error	A system or service error has occurred. The system may be able to recover but there might be a momentary loss, or permanent degradation, of service.
Warn	A suspicious operation or configuration has occurred but it may not have an impact on normal operation.
Info	Used for reporting normal operations and behaviour.

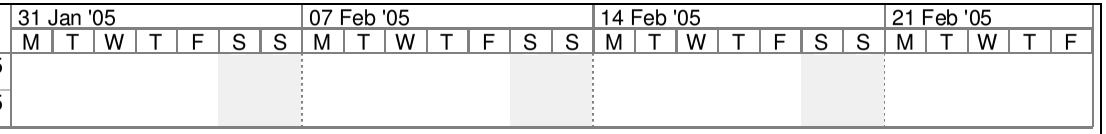
ID	Task Name	Duration	Start
1	Write report	28,1 wks	Tue 01-02-05
2	Milestone: Outline of report (table of contents)	1 day	Fri 11-02-05
3	Domain modelling	30 days	Mon 14-02-05
4	Milestone: Review of Domain Model	1 hr	Thu 07-04-05
5	Domain model theory and application	101,38 days	Thu 07-04-05
6	Find areas where RAISE is appropriate	17 days	Mon 07-03-05
7	Milestone: Review of RAISE choice	1 day	Mon 11-04-05
8	RAISE theory and application	98,5 days	Tue 12-04-05
9	Deadline: Turn in report	1 hr	Mon 05-09-05
10			
11	Easter break	8 days	Mon 21-03-05
12	Vacation in Greece	6 days	Mon 13-06-05
13			
14	Setup of tools	5 days	Tue 01-02-05
15	Technology testing	10 days	Tue 08-02-05
16	Planning	4 days	Tue 22-02-05
17	Analysis of Service Framework	2 days	Mon 28-02-05
18	Design of Service Framework	3 days	Wed 02-03-05
19	Milestone: Design Review of Service Framework	1 hr	Mon 07-03-05
20	Implementation of Service Framework	9 days	Tue 08-03-05
21	Milestone: Code Review of Service Framework	1 hr	Thu 31-03-05
22	Acquire information on OPC APIs etc.	5 days	Thu 31-03-05
23	Milestone: Choose OPC implementation	1 hr	Thu 07-04-05
24	Get to know the OPC API	3 days	Thu 07-04-05
25	Analysis of OPC Front End	4 days	Tue 12-04-05
26	Design of OPC Front End	5 days	Mon 18-04-05
27	Milestone: Design Review of OPC Front End	1 hr	Mon 25-04-05
28	Implementation of OPC Front End	15 days	Tue 26-04-05
29	Milestone: Code Review of OPC Front End	1 hr	Tue 17-05-05
30	Analysis of Test Plan	2 days	Wed 18-05-05



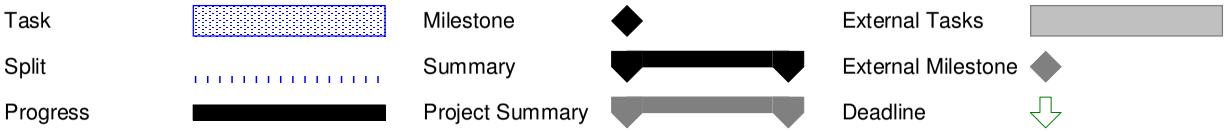
Project: Project.mpp
Date: Thu 01-09-05

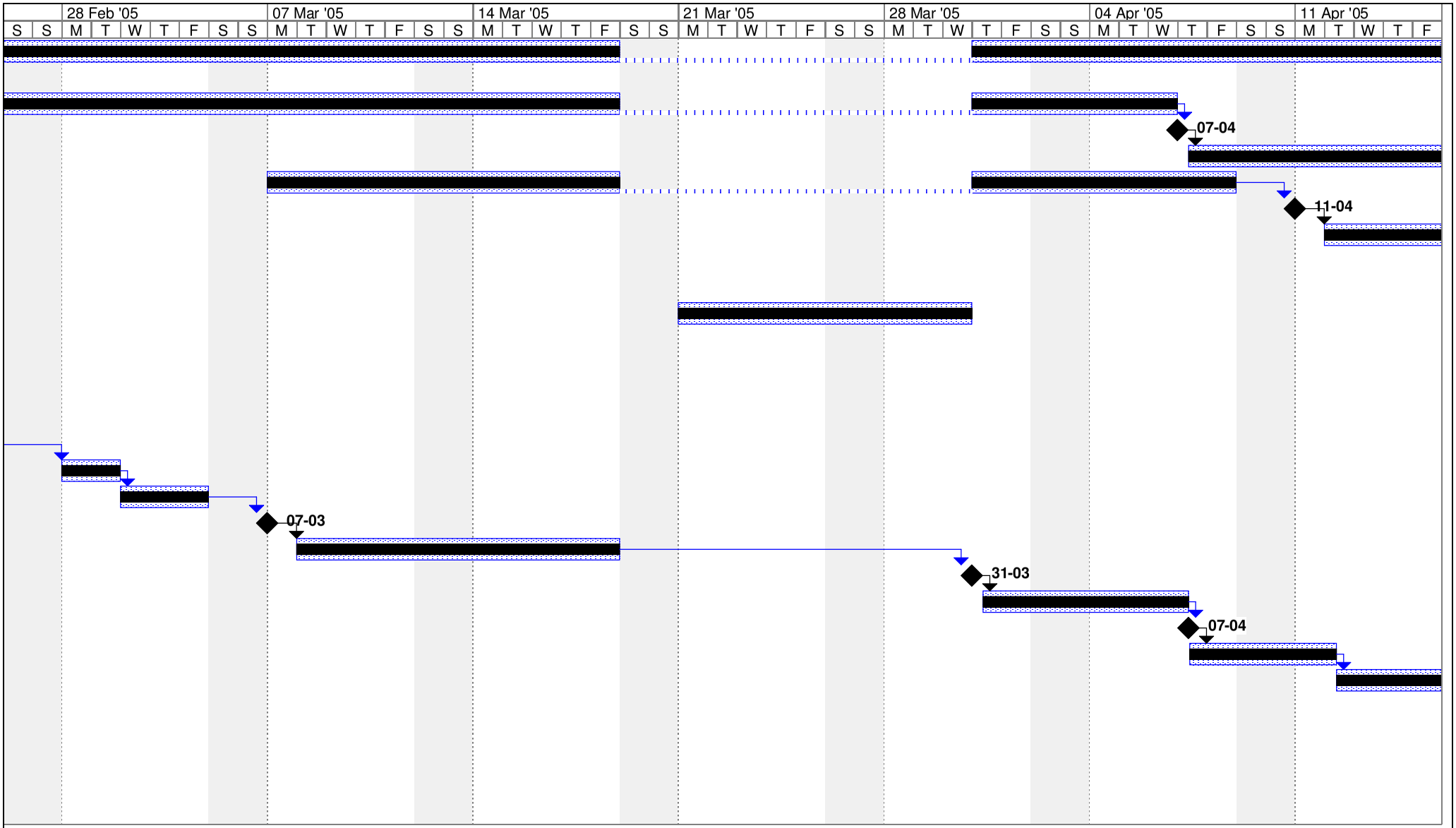
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

ID	Task Name	Duration	Start
31	Construction of Test Plan	4 days	Fri 20-05-05
32	Milestone: Review of Test Plan	1 hr	Thu 26-05-05



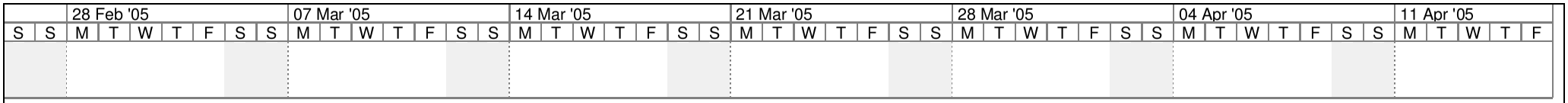
Project: Project.mpp
Date: Thu 01-09-05





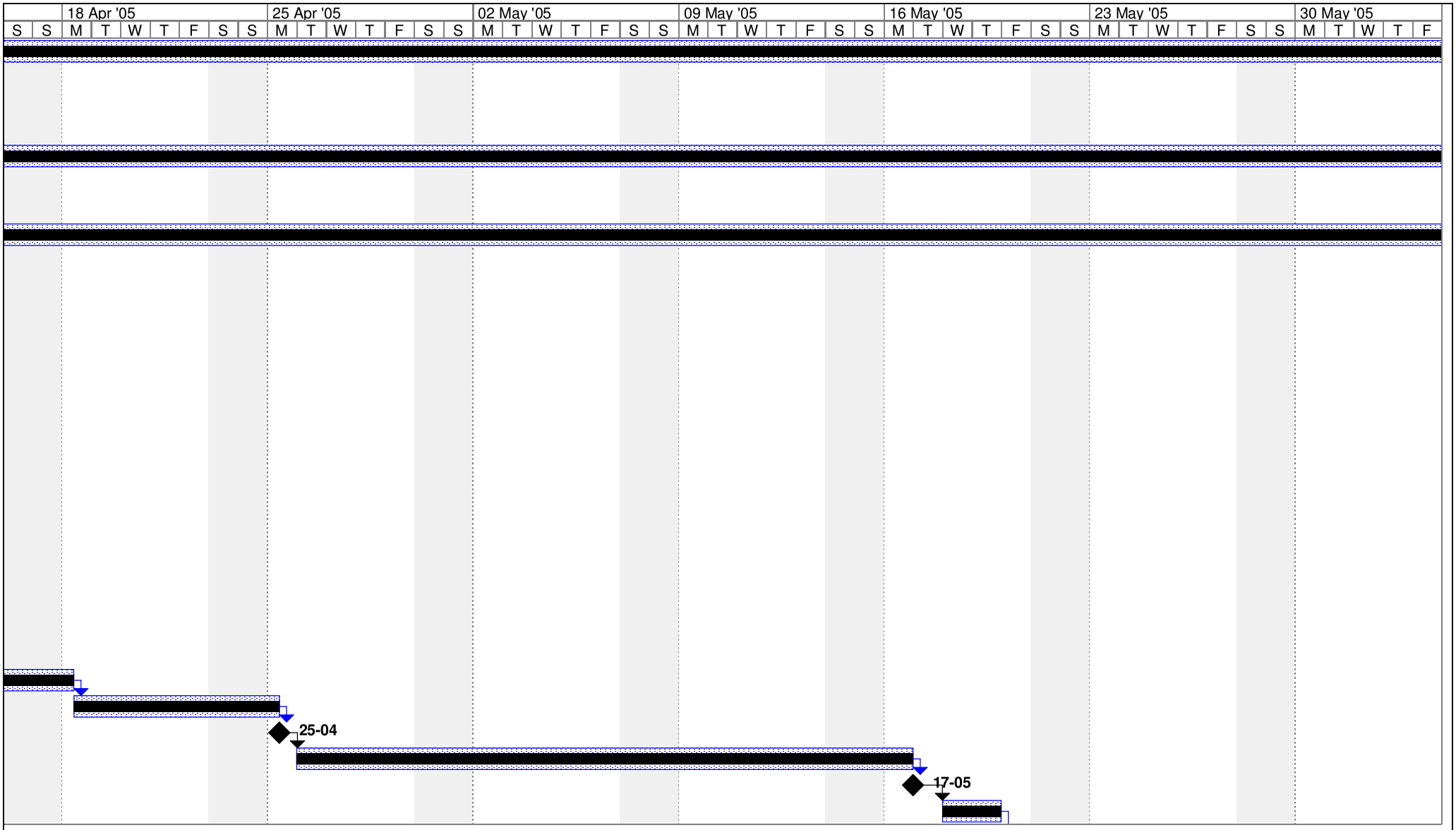
Project: Project.mpp
Date: Thu 01-09-05

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	



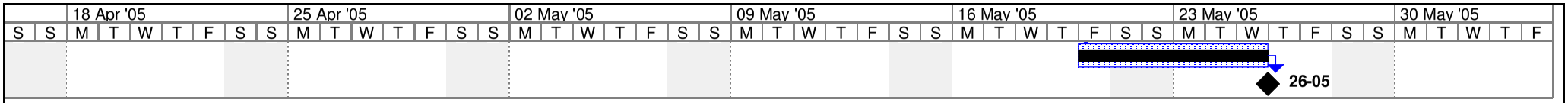
Project: Project.mpp
Date: Thu 01-09-05

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline



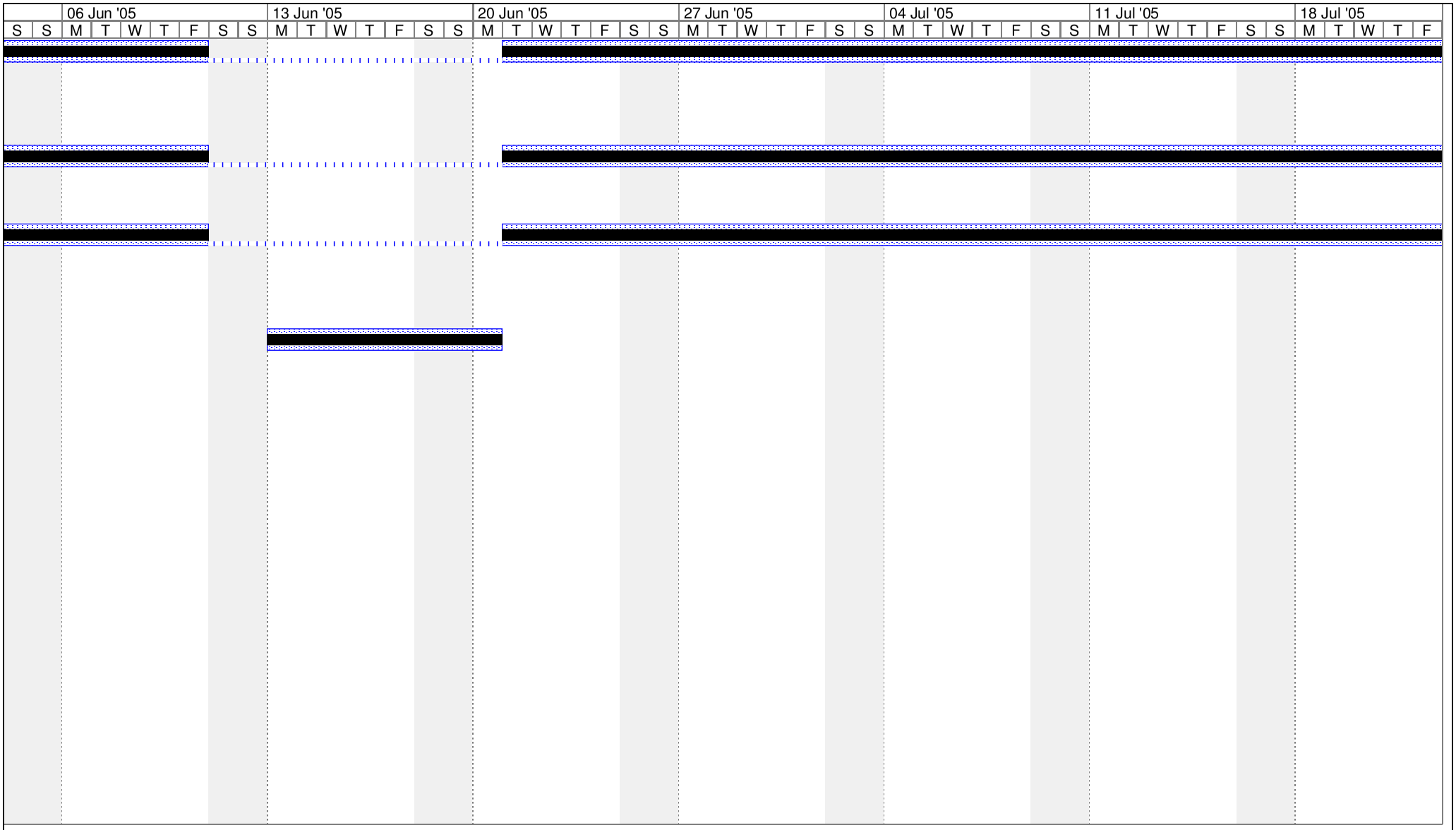
Project: Project.mpp
Date: Thu 01-09-05

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	

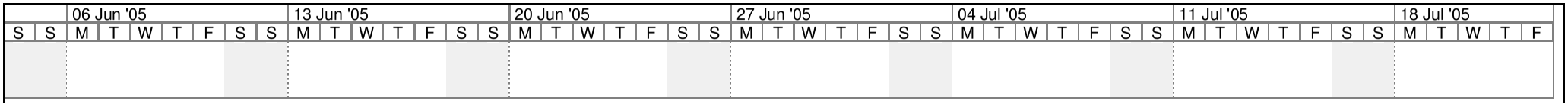


Project: Project.mpp
Date: Thu 01-09-05

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

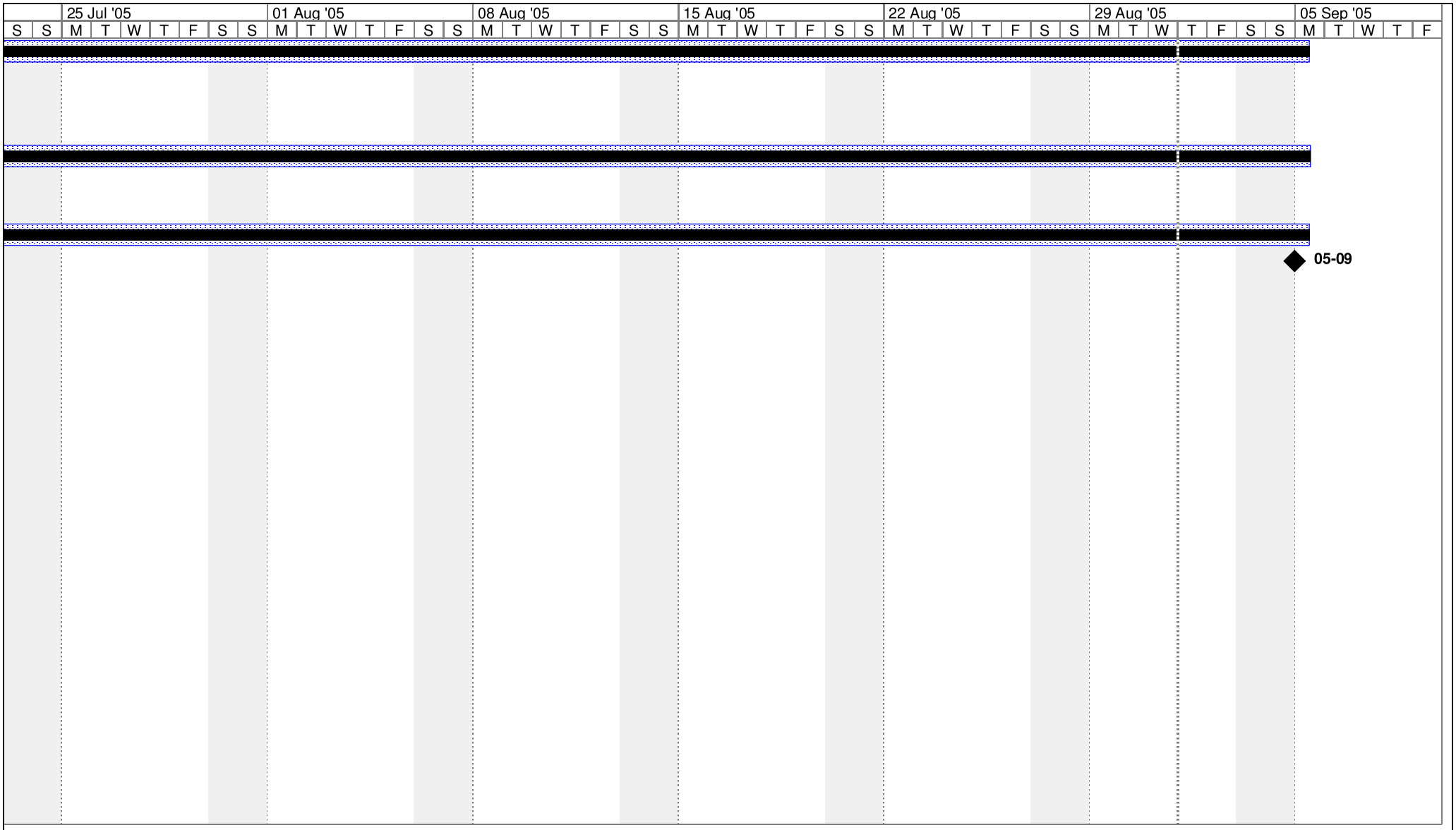


Project: Project.mpp
Date: Thu 01-09-05



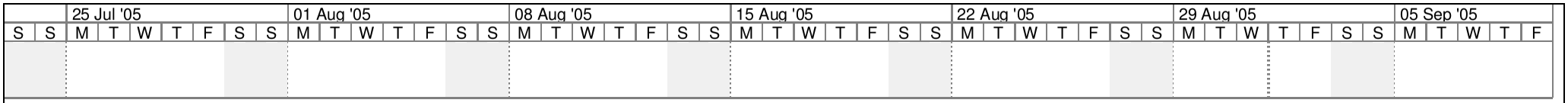
Project: Project.mpp
Date: Thu 01-09-05

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline



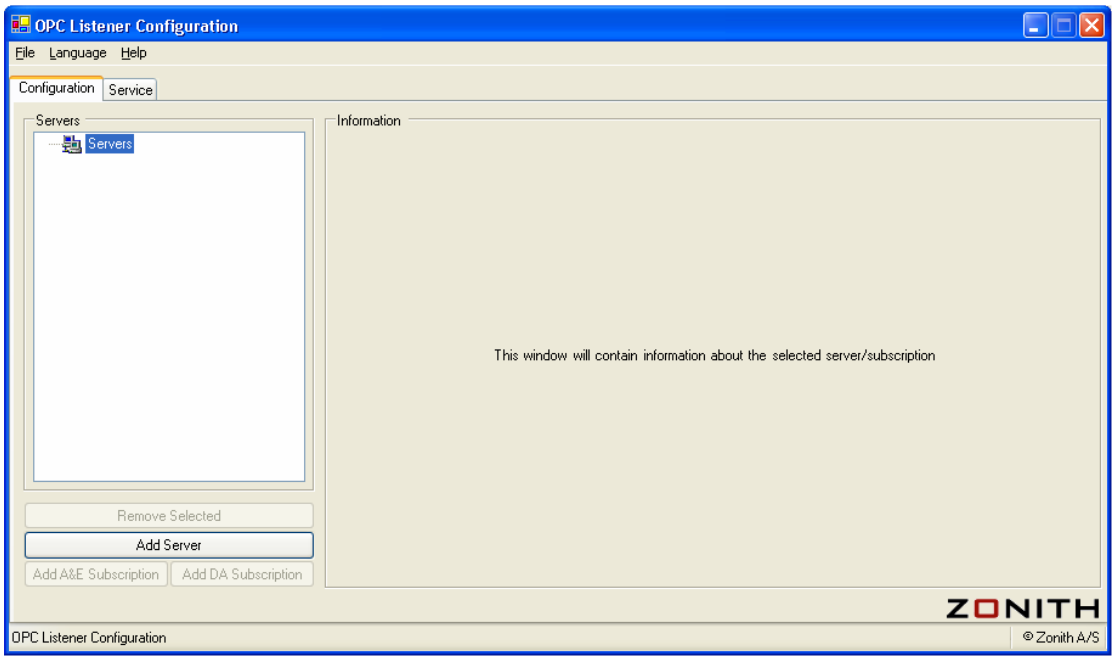
Project: Project.mpp
Date: Thu 01-09-05

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

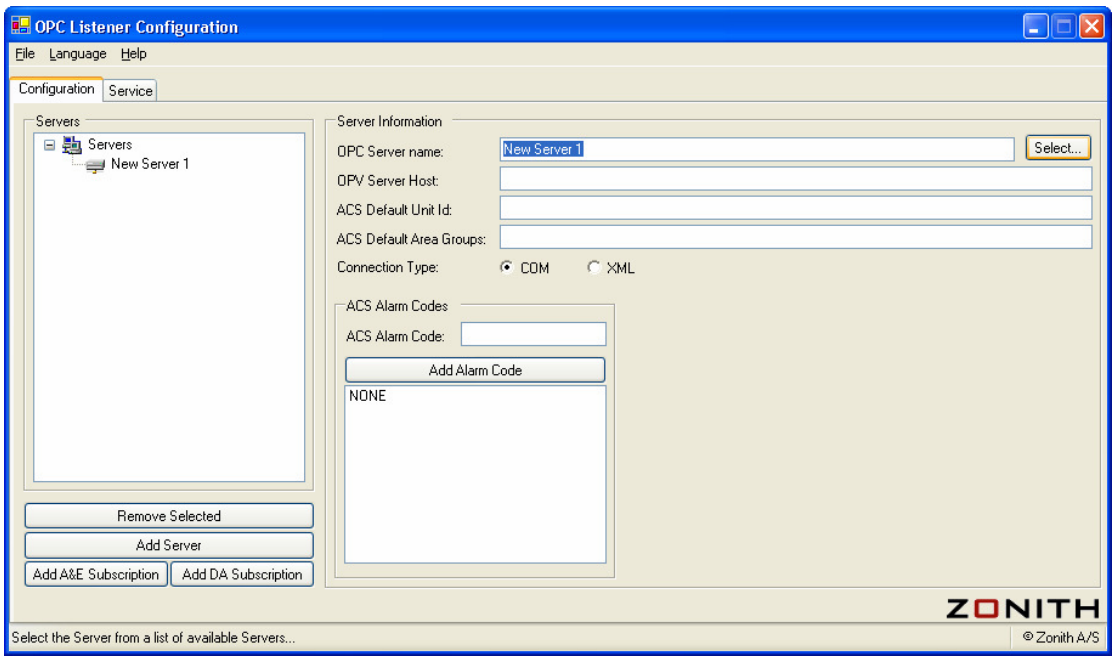


Project: Project.mpp
Date: Thu 01-09-05

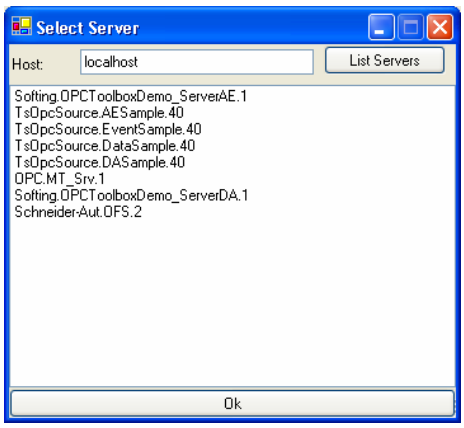
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline



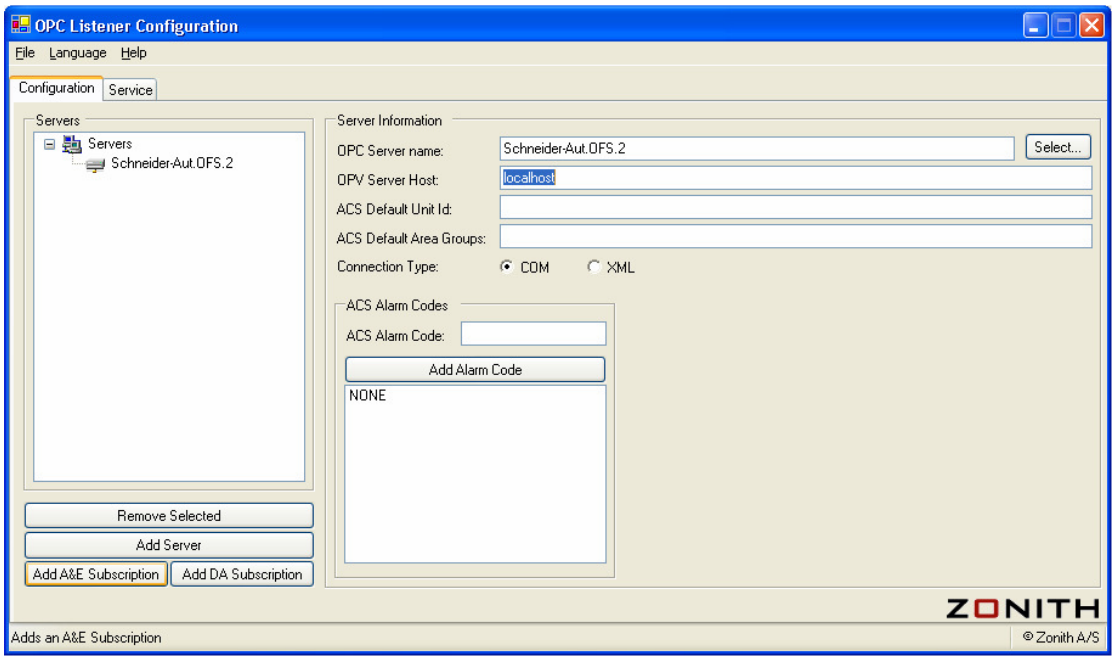
Screenshot 1



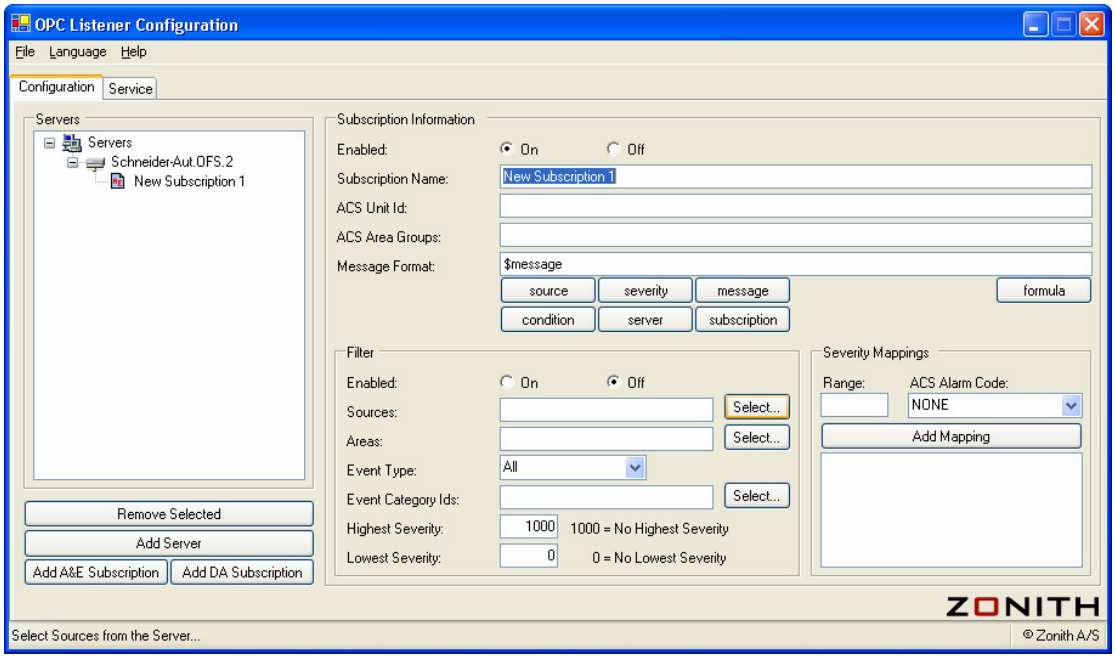
Screenshot 2



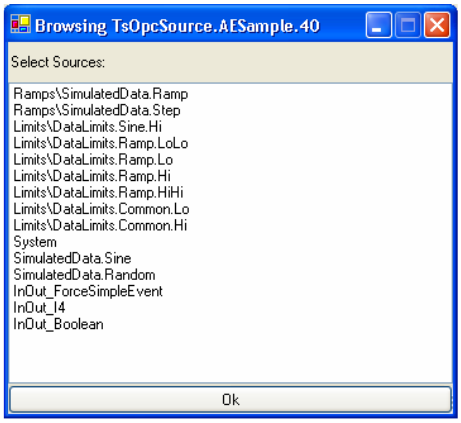
Screenshot 3



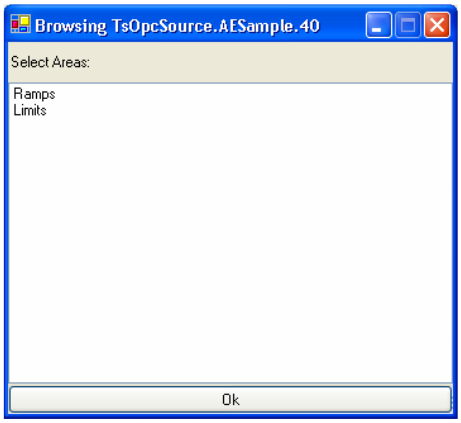
Screenshot 4



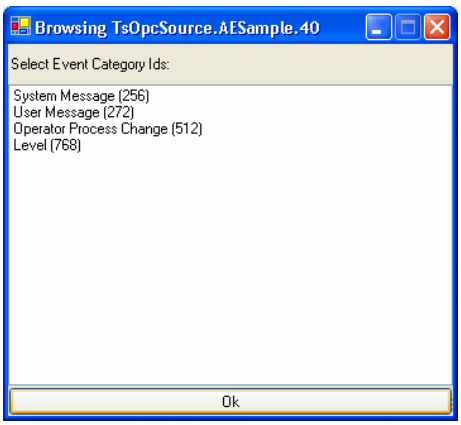
Screenshot 5



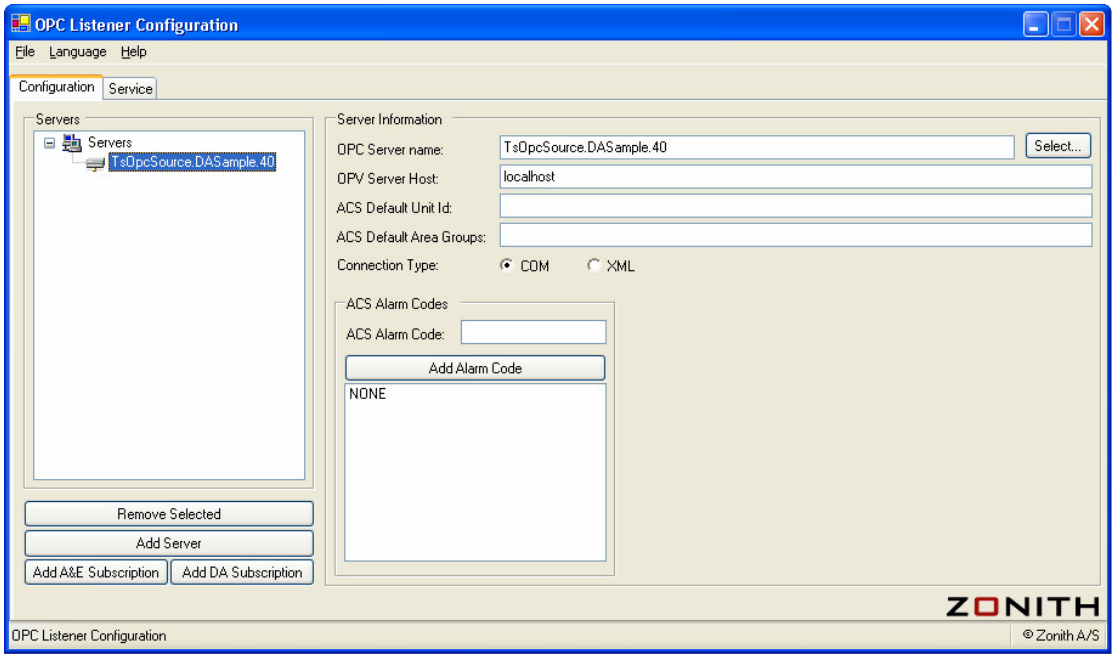
Screenshot 6



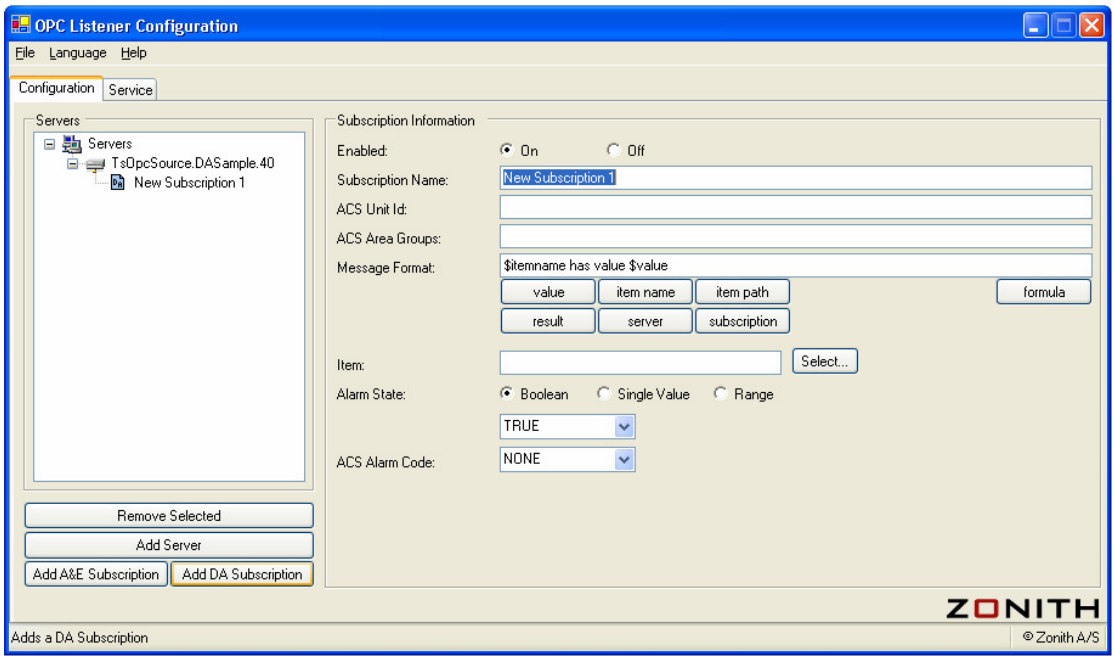
Screenshot 7



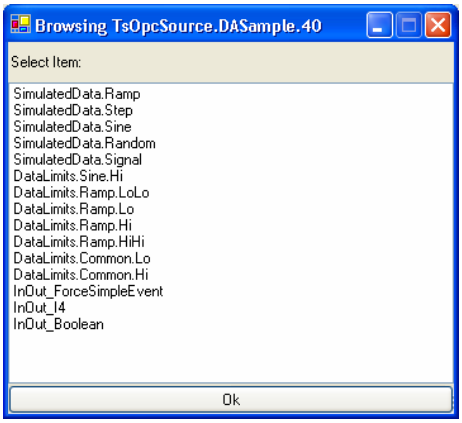
Screenshot 8



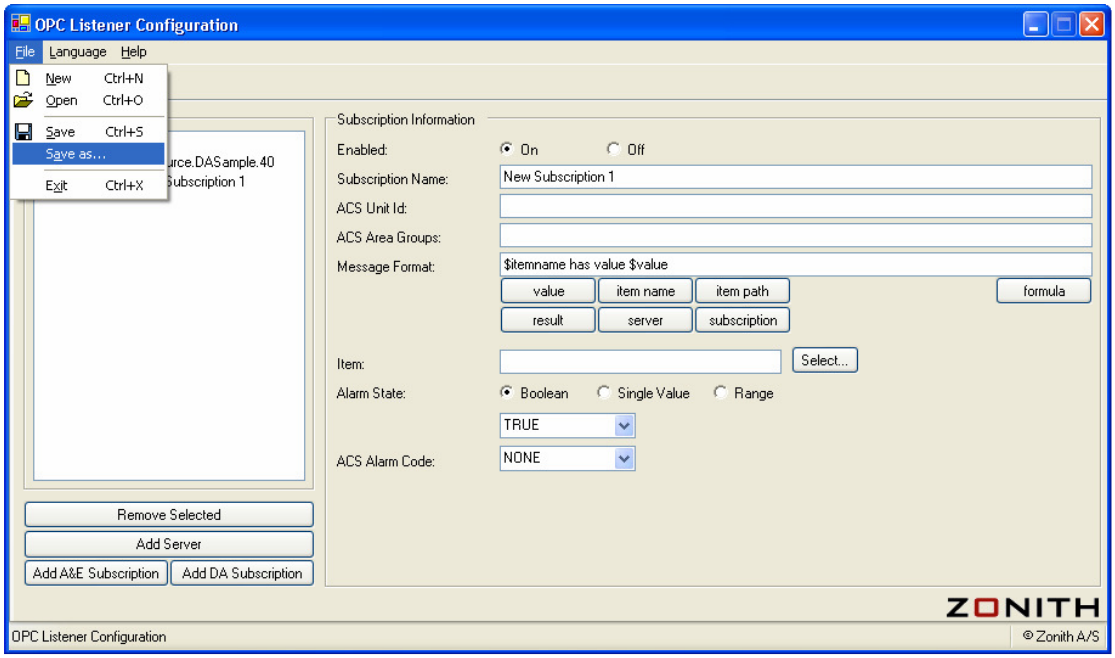
Screenshot 9



Screenshot 10



Screenshot 11



Screenshot 12

```
scheme Connecting_it1 =
  class
    type
      Details,
      SubscriptionElem = Text × Details,
      ServerElem = Text × SubscriptionElem-set,
      ConfigFile = ServerElem-set,
      Handler = Text × SubscriptionElem

    value
      empty : ConfigFile,
      createSubscriptions : ConfigFile → Handler-set

    axiom
      empty ≡ {},
      ∀ C : ConfigFile •
        createSubscriptions(C) ≡
          {(T1, S1) |
           (T1, S1) : Handler •
             ∃
               (T2, S2) : ServerElem,
               (T3, D) : SubscriptionElem
             •
               (T2, S2) ∈ C ∧ (T3, D) ∈ S2 ∧
               (T3, D) = S1 ∧ T1 = T2}
  end
```

```
scheme Connecting_it2a =
  class
    type
      Details,
      SubscriptionElem = Text × Details,
      ServerElem = Text × SubscriptionElem-set,
      ConfigFile = ServerElem-set,
      Handler = Text × SubscriptionElem,
      Handlers = Handler-set,
      Input ==
        mk_createHandler(ServerElem, SubscriptionElem) |
        mk_createHandlers(ConfigFile),
      Output == create_succeeded | create_failed

    value
      evaluate : Input × Handlers → Handlers × Output

    axiom
      /∗ Create a single handler (result is added to
          the list) ∗/
      ∀
        H : Handlers, S1 : ServerElem, S2 : SubscriptionElem
      •
        evaluate(mk_createHandler(S1, S2), H) ≡
          let (server_name, subscriptions) = S1 in
            if (S2 ∈ subscriptions)
            then
              (H ∪ {(server_name, S2)}, create_succeeded)
            else (H, create_failed)
            end
          end,
      /∗ Create a handlers for server elements in the
          config file (result is added to the list) ∗/
      ∀ H : Handlers, C : ConfigFile •
        evaluate(mk_createHandlers(C), H) ≡
          let
            outputs =
              {O |
               O : (Handlers × Output) •
                 (∀ (server, subscriptions) : ServerElem •
                     (server, subscriptions) ∈ C ∧
                     (∀ S1 : SubscriptionElem •
                         S1 ∈ subscriptions ∧
                         O =
                           evaluate(
                             mk_createHandler(
                               (server, subscriptions), S1),
                             H)))},
            handlerSets =
              {H1 |
               (H1, O) : (Handlers × Output) •
```

```
                (H1, O) ∈ outputs ∧
                O = create_succeeded},
            handlers =
              {H1 |
               H1 : Handler •
                 (∀ HS : Handlers •
                     HS ∈ handlerSets ∧ H1 ∈ HS)}
        in
          (H ∪ handlers, create_succeeded)
        end
end
```

```
scheme Connecting_it2b =
  class
    type
      Details,
      SubscriptionElem = Text × Details,
      ServerElem = Text × SubscriptionElem-set,
      ConfigFile = ServerElem-set,
      Handler = Text × SubscriptionElem

    value
      createSubscriptions : ConfigFile → Handler-set

    axiom
      /∗ Create handlers for server elements in the
          config file ∗/
      ∀ C : ConfigFile •
        createSubscriptions(C) as H post
          H =
            {(T1, S1) |
             (T1, S1) : Handler •
               ∃
                 (T2, S2) : ServerElem,
                 (T3, D) : SubscriptionElem
               •
                 (T2, S2) ∈ C ∧ (T3, D) ∈ S2 ∧
                 (T3, D) = S1 ∧ T1 = T2} ∧ card H = card C
  end
```

```
object Connecting_it3 :
  class
    variable
      connectedServers : ServerElem-set,
      handlers : Handler-set

    type
      Details,
      SubscriptionType == AE | DA,
      ServerType == AE | DA | BOTH,
      SubscriptionElem = Text × SubscriptionType × Details,
      ServerElem =
        Text × ServerType × SubscriptionElem-set,
      ConfigFile = ServerElem-set,
      AeDetails,
      DaDetails,
      Handler ==
        mk_AE_subscription(
          ae_name : Text, details : AeDetails) |
        mk_DA_subscription(
          da_name : Text, details : DaDetails)

    value
      connectToServers : ConfigFile → Bool,
      connectToAeServers : ServerElem-set → Bool,
      createAeSubscription : SubscriptionElem → Bool,
      /∗ Gets the AE details from an AE subscription
         element ∗/
      getAeDetails : SubscriptionElem → AeDetails,
      connectToDaServers : ServerElem-set → Bool,
      createDaSubscription : SubscriptionElem → Bool,
      /∗ Gets the DA details from a DA subscription
         element ∗/
      getDaDetails : SubscriptionElem → DaDetails

    axiom
      /∗ Connect to servers in config file ∗/
      ∀ C : ConfigFile •
        connectToServers(C) ≡
          let
            servers = {S | S : ServerElem • S ∈ C},
            aeServers =
              {(Sname, Stype, Ssubs) |
               (Sname, Stype, Ssubs) : ServerElem •
                 (Sname, Stype, Ssubs) ∈ servers ∧
                 (Stype = AE ∨ Stype = BOTH)},
            daServers =
              {(Sname, Stype, Ssubs) |
               (Sname, Stype, Ssubs) : ServerElem •
                 (Sname, Stype, Ssubs) ∈ servers ∧
                 (Stype = DA ∨ Stype = BOTH)}
```

```
        in
            connectToAeServers(aeServers) ∧
            connectToDaServers(daServers)
        end,
/∗ Connect to AE servers ∗/
∀ S : ServerElem-set •
    connectToAeServers(S) ≡
        let
            subs =
                {subscription |
                 subscription : SubscriptionElem,
                 (serverName, serverType, subscriptions) :
                     ServerElem •
                     (serverName, serverType, subscriptions) ∈
                         S ∧ subscription ∈ subscriptions ∧
                     serverType = AE ∧
                     createAeSubscription(subscription)}
        in
            connectedServers := connectedServers ∪ S ;
            true
        end,
/∗ Create an AE Subscription ∗/
∀ S : SubscriptionElem •
    createAeSubscription(S) ≡
        let (subName, subType, subDetails) = S in
            if (subType = AE)
            then
                let
                    handler =
                        mk_AE_subscription(
                            subName, getAeDetails(S))
                in
                    handlers := handlers ∪ {handler} ; true
                end
            else true
            end
        end,
/∗ Connect to DA servers ∗/
∀ S : ServerElem-set •
    connectToAeServers(S) ≡
        let
            subs =
                {subscription |
                 subscription : SubscriptionElem,
                 (serverName, serverType, subscriptions) :
                     ServerElem •
                     (serverName, serverType, subscriptions) ∈
                         S ∧ subscription ∈ subscriptions ∧
                     serverType = DA ∧
                     createAeSubscription(subscription)}
        in
```

```
            connectedServers := connectedServers ∪ S ;
            true
        end,
    /* Create a DA Subscription */
    ∀ S : SubscriptionElem •
      createAeSubscription(S) ≡
        let (subName, subType, subDetails) = S in
        if (subType = DA)
        then
          let
            handler =
              mk_DA_subscription(
                subName, getDaDetails(S))
          in
            handlers := handlers ∪ {handler} ; true
          end
        else true
        end
    end
end
```

```
object Connecting_it4 :
  class
    variable
      connectedServers : ServerElem-set,
      handlers : Handler-set,
      /∗ The list within the OPC server of all the connected
          subscribers ∗/
      subscribers : Handler-set

    type
      AeSpecifics,
      DaSpecifics,
      RemoteUnit,
      AreaGroups,
      MessageFormat,
      CommonDetails =
        RemoteUnit × AreaGroups × MessageFormat,
      Details ==
        mk_AE_details(CommonDetails, AeSpecifics) |
        mk_DA_details(CommonDetails, DaSpecifics),
      SubscriptionType == AE | DA,
      ServerType == AE | DA | BOTH,
      SubscriptionElem ==
        mk_subscription(
          subscription_name : Text,
          subscription_type : SubscriptionType,
          subscription_details : Details),
      ServerElem ==
        mk_server(
          server_name : Text,
          server_type : ServerType,
          server_subscriptions : SubscriptionElem-set),
      ConfigFile = ServerElem-set,
      Handler ==
        mk_AE_subscription(
          ae_name : Text, details : AeSpecifics) |
        mk_DA_subscription(
          da_name : Text, details : DaSpecifics)

    value
      connectToServers : ConfigFile → Bool,
      connectToAeServers : ServerElem-set → Bool,
      createAeSubscription : SubscriptionElem → Bool,
      connectToDaServers : ServerElem-set → Bool,
      createDaSubscription : SubscriptionElem → Bool

    axiom
      /∗ Connect to servers in config file
          post: There must be created the same number
                of handlers as there are subscriptions
                in the config file.
```

```
*/
∀ C : ConfigFile •
   connectToServers(C) ≡
      let
         servers = {S | S : ServerElem • S ∈ C},
         aeServers =
            {server |
             server : ServerElem •
                server ∈ servers ∧
                (server_type(server) = AE ∨
                  server_type(server) = BOTH)},
         daServers =
            {server |
             server : ServerElem •
                server ∈ servers ∧
                (server_type(server) = DA ∨
                  server_type(server) = BOTH)}
      in
         (connectToAeServers(aeServers) ∧
          connectToDaServers(daServers))
            post
            (card handlers =
               card {s |
                        s : SubscriptionElem •
                           (∀ serv : ServerElem •
                             serv ∈ C ∧
                             s ∈
                               server_subscriptions(serv))})
      end,
/* Connect to AE servers
       post: The given set of servers must be a
             subset of the servers connected to.
*/
∀ S : ServerElem-set •
   connectToAeServers(S) ≡
      let
         subs =
            {subscription |
             subscription : SubscriptionElem,
             server : ServerElem •
                server ∈ S ∧
                subscription ∈
                   server_subscriptions(server) ∧
                server_type(server) = AE ∧
                createAeSubscription(subscription)}
      in
         (connectedServers := connectedServers ∪ S ;
          true)
            post
            (S ⊆ connectedServers)
      end,
```

```
/* Create an AE Subscription
        post: The handler is is added to the list of
                handlers.
*/
∀ S : SubscriptionElem •
   createAeSubscription(S) ≡
      if (subscription_type(S) = AE)
      then
         let
            mk_AE_details(common, aeDetails) =
               subscription_details(S),
            handler =
               mk_AE_subscription(
                  subscription_name(S), aeDetails)
         in
            (handlers := handlers ∪ {handler} ; true)
               post
               handler ∈ handlers
         end
      else true
      end,
/* Connect to DA servers
        post: The given set of servers must be a
                subset of the servers connected to.
*/
∀ S : ServerElem-set •
   connectToAeServers(S) ≡
      let
         subs =
            {subscription |
             subscription : SubscriptionElem,
             server : ServerElem •
                server ∈ S ∧
                subscription ∈
                   server_subscriptions(server) ∧
                server_type(server) = DA ∧
                createAeSubscription(subscription)}
      in
         (connectedServers := connectedServers ∪ S ;
          true)
            post
            (S ⊆ connectedServers)
      end,
/* Create a DA Subscription
        post: The handler is is added to the list of
                handlers.
*/
∀ S : SubscriptionElem •
   createAeSubscription(S) ≡
      if (subscription_type(S) = DA)
      then
```

```
                let
                    mk_DA_details(common, daDetails) =
                        subscription_details(S),
                    handler =
                        mk_DA_subscription(
                            subscription_name(S), daDetails)
                in
                    (handlers := handlers ∪ {handler} ; true)
                        post
                        handler ∈ handlers
                end
            else true
            end,
        /* The list of connected handlers must always
            be a subset of the OPC servers list of
            connected handlers
        */
        handlers ⊆ subscribers
end
```

```
scheme Persisting_it1 =
  class
    type Event, Storage = Event-set

    value
      empty : Storage,
      add : Storage × Event → Storage,
      delete : Storage × Event → Storage,
      hasWaitingEvent : Storage → Bool,
      retrieveEvent : Storage → Event,
      transmit : Event → Bool

    axiom
      empty ≡ {},
      /∗ Add an event to the storage ∗/
      ∀ E : Event, S : Storage • add(S, E) ≡ S ∪ {E},
      /∗ Delete an event from the storage ∗/
      ∀ E : Event, S : Storage • delete(S, E) ≡ S \ {E},
      /∗ Check if the storage has waiting events ∗/
      ∀ E : Event, S : Storage •
        hasWaitingEvent(S) ≡ (∃ E : Event • E ∈ S)
  end
```

```
scheme Persisting_it2a =
  class
    type
      Id = Int,
      Details,
      Event = Id × Details,
      Storage = Event-set,
      Input ==
        mk_empty |
        mk_add(Event) |
        mk_delete(Event) |
        mk_hasWaitingEvent |
        mk_retrieveEvent,
      Output ==
        empty_storage |
        event_added |
        event_deleted |
        no_events_waiting |
        events_waiting(Nat) |
        event_retrieved(Event) |
        error_retrieving_event

    value
      evaluate : Input × Storage → Storage × Output,
      transmit : Event → Bool

    axiom
      ∀ S : Storage •
        evaluate(mk_empty, S) ≡ ({}, empty_storage),
      /∗ Add an event to the storage
            post: The event is added to the storage
      ∗/
      ∀ E : Event, S : Storage •
        evaluate(mk_add(E), S) ≡ (S ∪ {E}, event_added),
      /∗ Delete an event from the storage
            post: The event is deleted from the storage
      ∗/
      ∀ E : Event, S : Storage •
        evaluate(mk_delete(E), S) ≡ (S \ {E}, event_deleted),
      /∗ Check if the storage has waiting events ∗/
      ∀ S : Storage •
        evaluate(mk_hasWaitingEvent, S) ≡
          if (∃ E : Event • E ∈ S)
          then (S, events_waiting(card S))
          else (S, no_events_waiting)
          end,
      /∗ Retrieve an event from the storage
            pre:  The event is in the storage
            post: The event is still in the storage (use delete to remove)
      ∗/
      ∀ E : Event, S : Storage •
```

```
        evaluate(mk_retrieveEvent, S) ≡
            if (card (S) > 0 ∧ E ∈ S)
            then (S, event_retrieved(E))
            else (S, error_retrieving_event)
            end,
    /∗ Events are unique ∗/
    ∀ E1, E2 : Event •
        E1 = E2 ⇒
            let (id1, details1) = E1, (id2, details2) = E2 in
                id1 = id2 ∧ details1 = details2
            end
end
```

```
scheme Persisting_it2b =
  class
    type
      Id = Int,
      Details,
      Event = Id × Details,
      Storage = Event-set

    value
      add : Storage × Event → Storage,
      delete : Storage × Event → Storage,
      hasWaitingEvent : Storage → Bool,
      retrieveEvent : Storage → Event,
      transmit : Event → Bool

    axiom
      /∗ Add an event to the storage
            post: The event is added to the storage
      ∗/
      ∀ E : Event, S : Storage •
        add(S, E) as S1 post S1 = S ∪ {E},
      /∗ Delete an event from the storage
            post: The event is deleted from the storage
      ∗/
      ∀ E : Event, S : Storage •
        delete(S, E) as S1 post S1 = S \ {E},
      /∗ Check if the storage has waiting events ∗/
      ∀ S : Storage • hasWaitingEvent(S) ≡ card S > 0,
      /∗ Retrieve an event from the storage
            post: The event is still in the storage (use delete to remove)
      ∗/
      ∀ S : Storage • retrieveEvent(S) as E post E ∈ S,
      /∗ Events are unique ∗/
      ∀ E1, E2 : Event •
        E1 = E2 ⇒
          let (id1, details1) = E1, (id2, details2) = E2 in
            id1 = id2
          end
  end
```

```
scheme Persisting_it3 =
  class
    type
      Id = Int,
      Timestamp,
      Code,
      UnitId,
      AreaGroups,
      Event ==
        event(
          id : Id,
          timestamp : Timestamp,
          code : Code,
          unitid : UnitId,
          areagroups : AreaGroups,
          text : Text),
      Storage = Event-set

    value
      add : Storage × Event → Storage,
      delete : Storage × Event → Storage,
      hasWaitingEvent : Storage → Bool,
      retrieveEvent : Storage → Event,
      transmit : Event → Bool

    axiom
      /∗ Add an event to the storage
           pre:  The event cannot already be in the storage
           post: The event is added to the storage
      ∗/
      ∀ E : Event, S : Storage •
        add(S, E) as S1 post S1 = S ∪ {E} pre E ∉ S,
      /∗ Delete an event from the storage
           pre:  The event is in the storage
           post: The event is deleted from the storage
      ∗/
      ∀ E : Event, S : Storage •
        delete(S, E) as S1 post S1 = S \ {E} pre E ∈ S,
      /∗ Check if the storage has waiting events ∗/
      ∀ S : Storage • hasWaitingEvent(S) ≡ card S > 0,
      /∗ Retrieve an event from the storage
           pre:  The event is in the storage
           post: The event is still in the storage (use
                   delete to remove)
      ∗/
      ∀ E : Event, S : Storage •
        retrieveEvent(S) as E post E ∈ S pre E ∈ S,
      /∗ Transmit the event to ACS
           pre: The event is in the storage (not implemented
                  here)
      ∗/
```

$\forall$ E : Event • transmit(E) $\equiv$ **true**,
/$*$ Events have uniqe id$'$s $*$/
$\forall$ E1, E2 : Event • E1 = E2 $\Rightarrow$ id(E1) = id(E2)

end

```
object Persisting_it4 :
  class
    variable
      storage : Event-set
    type
      Id = Int,
      Timestamp, Code, UnitId, AreaGroups,
      Event == event(id : Id, timestamp: Timestamp, code : Code, unitid
      : UnitId, areagroups : AreaGroups, text : Text)

    value
      add : Event → write storage Unit,
      delete : Event → write storage Unit,
      hasWaitingEvent : Unit → read storage Bool,
      retrieveEvent : Unit → read storage Event,
      transmit : Event → Bool

    axiom
      /∗ Add an event to the storage
            pre:  The event cannot already be in the storage
            post: The event is added to the storage
      ∗/
      ∀ E : Event • add(E) ≡ if (E ∉ storage) then storage :=
      storage ∪ {E} end,
      /∗ Delete an event from the storage
            pre:  The event is in the storage
            post: The event is deleted from the storage
      ∗/
      ∀ E : Event • delete(E) ≡ if (E ∈ storage) then storage
      := storage \ {E} end,
      /∗ Check if the storage has waiting events ∗/
      hasWaitingEvent() ≡ card storage > 0,
      /∗ Retrieve an event from the storage
            pre:  The event is in the storage
            post: The event is still in the storage (use delete to remove)
      ∗/
      ∀ E : Event • retrieveEvent() as E post E ∈ storage pre E ∈ storage,
      /∗ Transmit the event to ACS
            pre: The event is in the storage
      ∗/
      ∀ E : Event • transmit(E) ≡ true pre E ∈ storage,

      /∗ Events have uniqe id′s ∗/
      ∀ E1, E2 : Event • E1 = E2 ⇒ id(E1) = id(E2)
  end
```

```
object Persisting_it5 :
  class
    variable storage : Event-set, acs_storage : Event-set

    type
      Id = Int,
      Timestamp = Int,
      Code,
      UnitId,
      AreaGroups,
      Event ==
        event(
          id : Id,
          timestamp : Timestamp,
          code : Code,
          unitid : UnitId,
          areagroups : AreaGroups,
          text : Text)

    value
      add : Event → write storage Unit,
      delete : Event → write storage Unit,
      hasWaitingEvent : Unit → read storage Bool,
      retrieveEvent : Unit → read storage Event,
      transmit : Event → Bool,
      back_end : Unit → Unit,
      run : Unit → Unit,
      /∗ Reception of an alarm ∗/
      alarmFromFrontEnd : Unit → Event

    axiom
      /∗ Add an event to the storage
           pre:  The event cannot already be in the storage
           post: The event is added to the storage
      ∗/
      ∀ E : Event •
        add(E) ≡
          if (E ∉ storage)
          then storage := storage ∪ {E}
          end,
      /∗ Delete an event from the storage
           pre:  The event is in the storage
           post: The event is deleted from the storage
      ∗/
      ∀ E : Event •
        delete(E) ≡
          if (E ∈ storage) then storage := storage \ {E}
          end,
      /∗ Check if the storage has waiting events ∗/
      hasWaitingEvent() ≡ card storage > 0,
      /∗ Retrieve an event (the oldest) from the storage
```

```
            pre:  The event is in the storage
            post: The event is still in the storage (use
                  delete to remove)
        */
        ∀ E : Event •
          retrieveEvent() as E post
            hasWaitingEvent() ∧ E ∈ storage
          pre
            hasWaitingEvent() ∧ E ∈ storage ∧
            (∀ E1 : Event •
                E1 ∈ storage ∧ E1 ≠ E ∧
                timestamp(E) < timestamp(E1)),
        /* Transmit the event to ACS
            pre:  The event is in the storage
            post: The event is added to the ACS storage
        */
        ∀ E : Event •
          transmit(E) ≡
            (acs_storage := acs_storage ∪ {E} ; true)
          pre E ∈ storage,
        /* Events have uniqe id's */
        ∀ E1, E2 : Event • E1 = E2 ⇒ id(E1) = id(E2),
        /* The back−ends functionality is to continuously
            perform the run() function, as well as handle
            incoming alarms
        */
        back_end() ≡ run() || add(alarmFromFrontEnd()),
        /* The body of the back−end. Continuously check
            if alarms are availble for transmission, and
            transmit them.
        */
        run() ≡
          if hasWaitingEvent()
          then
            let event = retrieveEvent() in
              if transmit(event) then delete(event) end
            end
          end
end
```

Test Specification

OPC Listener

Release

1.0.1

Test Specification by:
ZONITH A/S
Gammel Kongevej 39E, 1
1610 København V
CVR no. 25807766

Zonith A/S:
E-mail: support@zonith.com
Phone: +45 3332 4530

Document History

Version	Revisions	Initials
Version 0.1	Initial version	NH

Reference Documents

Version	Document Title and Info
Released	Using OPC via DCOM with Microsoft Windows XP Service Pack 2

Reference Applications

Application	Location
Softing Demo Client	http://www.softing.com/en/communications/products/opc/tools.htm

CONTENTS

1 INTRODUCTION

This specification is for internal test of OPC Listener release 1.0.1.

2 LISTENER TEST

2.1 Test Suite: Installation

#	Test Case	Comment	Conducted by
1	Install .NET Compact Framework (if not already part of the OS)		
2	Install OPC Core Components		
3	Install the OPC Listener in a directory **without** white spaces in path. • Can the OPC Configuration be started from the Programs menu? • Is the OPC Listener Service installed? Check the Service Tab of the OPC Configuration. • Can the service be started? Start from the Service Tab of the OPC Configuration. Check log files to verify correct start up.		
4	Uninstall the OPC Listener using "Add/Remove Programs" from Control Panel. • Is the OPC Listener directory empty apart from log files, storage files, language files, and configuration files? • Delete the OPC Listener directory completely.		
5	Install the OPC Listener in a directory **with** white spaces in path. • Can the OPC Configuration be started from the Programs menu? • Is the OPC Listener Service installed? Check the Service Tab of the OPC Configuration. • Can the service be started? Start from the Service Tab of the OPC Configuration. Check log files to verify correct start up.		

2.2 Test Suite: Alarms & Events interface (from OPC Server to OPC Listener)

#	Test Case	Comment	Conducted by
1	Install an OPC Server which supports the Alarms & Events interface (if not present already) • E.g. Demo Server by Technosoftware A/G		
2	Verify that the OPC Server can be executed, and events/alarms captured • Use a third party client to connect to the server (Demo Client by Softing for example) • Select the Installed Server, create a subscription, and verify that events/alarms are captured. 1. In Softing's Demo Client you select the Server from the 'OPC Servers' tab which automatically creates a subscription 2. Select the 'AE Events' and 'AE Conditions' tab and verify that some events/condition changes occur.		
3	Start the OPC Configuration from the Programs Menu, and create an Alarms & Events subscription to the installed server. • Click 'New Server' • Click 'Select…' next to the OPC Server Name input box and select the installed Alarms & Events compliant OPC Server • Select COM as the connection type • Click 'Add A&E Subscription' and enter an arbitrary subscription name • Add a remote unit ID either on the server or the subscription • Add all the dynamic elements to the Message Format • Save the configuration		
4	Disable the connection to ACS. • Edit %OPC Listener home%/config/listener.properties, and set 'listener.acs.endPoint' to a non-existent address. This will cause an NetException when trying to transmit to the ACS (ignore these exceptions)		
5	Ensure logging occurs at INFO level. • Edit %OPC Listener home%/config/log4net.config, and set the value of the level element of the two loggers to "ALL". The loggers are found at the bottom of the XML file		

6	Start the service and verify that events are captured. • Select the 'Service tab' • Click 'Start Service' • Wait a short while until some events/alarms have been generated (This is almost instantly with Technosoftware's Demo Server) • Click 'Stop Service' and wait until the OPC Configuration indicates the service is stopped • Check the log file %OPC Listener home%/logs/messagelog.log, and verify that events are received. An indication of a received event looks something like this: "… - Received OPC event: EVENT[<date>]{…}" • Ensure that all the dynamic elements of the Message Format were included.		

2.3 Test Suite: Alarms & Events interface (from OPC Listener to ACS)

#	Test Case	Comment	Conducted by
1	Make sure the ACS accepts (and shows) all incoming alarms/events having the remote unit specified in the previous test • Create a remote unit • Create a customer • Add a location to the customer, select the remote unit, and apply the same remote unit id as the one used in the test above		
2	Make sure the OPC Listener does not listen for incoming alarms/events • Open (if not already open) OPC Configuration • Select the menu item "New" from the "File" menu • Save the empty configuration, overriding the old configuration		
3	Enable the connection to ACS. • Edit %OPC Listener home%/config/listener.properties, and set 'listener.acs.endPoint' to an existent address where an ACS is located • E.g. http://localhost:8080/acs/services/AlarmService		
4	Start the OPC Listener and verify that the existing alarms/events are transmitted to the ACS • Select the "Service" tab in the OPC Configuration • Click "Start Service" • Wait a short while until the existing alarms/events have been transmitted to the ACS. If it is a limited number of alarms (<100), this will not take much longer than a minute or so • Click "Stop Service" • Check the log file %OPC Listener home%/logs/messagelog.log, and verify that the events were transmitted successfully • Check the ACS web interface to verify that the events were received correctly		

2.4 Test Suite: Alarms & Events interface (from OPC Server to ACS via OPC Listener)

#	Test Case	Comment	Conducted by
1	Cancel alarms already present in ACS web interface. • Select all the alarms received in the previous test, and click "Cancel"		
2	Start the OPC Configuration from the Programs Menu, and create an Alarms & Events subscription to the installed server like previous. • Click 'New Server' • Click 'Select…' next to the OPC Server Name input box and select the installed Alarms & Events compliant OPC Server • Select COM as the connection type • Click 'Add A&E Subscription' and enter an arbitrary subscription name • Add a remote unit ID either on the server or the subscription • Save the configuration		
3	Start the OPC Listener and verify that the captured events/alarms are transmitted to the ACS. • Select the "Service" tab in the OPC Configuration • Click "Start Service" • Wait a short while until some events/alarms have been generated (This is almost instantly with Technosoftware's Demo Server) • Click "Stop Service" • Check the log file %OPC Listener home%/logs/messagelog.log, and verify that the events were transmitted successfully • Check the ACS web interface to verify that the events were received correctly		

2.5 Test Suite: Alarms & Events interface – Filter and Severity Mappings

#	Test Case	Comment	Conducted by
1	Add a single Source to the filter of the subscription from above. • Open OPC Configuration and edit the subscription • Enable the filter and select a single source • Save the configuration • Select the "Service" tab • Click "Start Service" • Wait a little while until an event/alarm has been generated • Check log files to ensure the filter was added successfully, and one or more events/alarms were transmitted (if generated)		
2	Do the same as above, but select several Sources. • If entered manually, Sources are separated by a comma (,)		
3	Remove the filtering of Sources and add one Area instead. • Save the configuration • Start and stop the service like above • Check the log files to ensure the filter was added successfully, and one or more events/alarms were transmitted (if generated)		
4	Do the same as above, but select several Areas. • If entered manually, Areas are separated by a comma (,)		
5	Remove the filtering of Sources and change the Event Type (do this for all types, one at a time). • Save the configuration • Start and stop the service like above • Check the log files to ensure the filter was added successfully, and one or more events/alarms were transmitted (if generated)		
6	Set the Event Type to "All" and add a single Event Category Id. • Save the configuration • Start and stop the service like above • Check the log files to ensure the filter was added successfully, and one or more events/alarms were transmitted (if generated)		

7	Do the same as above, but select several Event Category Ids. • If entered manually, Event Category Ids are separated by a comma (,)		
8	Remove the filtering of Event Category Ids and change the Highest Severity to something between 1 and 999. • Save the configuration • Start and stop the service like above • Check the log files to ensure the filter was added successfully, and one or more events/alarms were transmitted (if generated). These events naturally have to have a severity between 0 and the value entered to be transmitted		
9	Change the Highest Severity back to 1000 and change the Lowest Severity to something between 1 and 999. • Save the configuration • Start and stop the service like above • Check the log files to ensure the filter was added successfully, and one or more events/alarms were transmitted (if generated). These events naturally have to have a severity between value entered and 1000 to be transmitted		
10	Disable the filter and at some Severity Mappings. Add at least 1 single value mapping, and one range. • Severity Mappings map an OPC severity (0-1000) to an ACS Code Mapping defined in the ACS web interface • Save the configuration • Start and stop the service like above • Check the log files and ACS web interface to ensure the mappings were used (if any events/alarms fall within their range/value)		

2.6 Test Suite: Data Access interface (from OPC Server to OPC Listener)

#	Test Case	Comment	Conducted by
1	Install an OPC Server which supports the Data Access interface (if not present already) • E.g. Demo Server by Technosoftware A/G		
2	Verify that the OPC Server can be executed, and data changes captured • Use a third party client to connect to the server (Demo Client by Softing for example) • Select the Installed Server, create a group, and verify that data changes are captured. 1. In Softing's Demo Client you select the Server from the 'OPC Servers' tab which automatically creates a subscription 2. Select the 'DA Browse' tab and double click on an item (not an area) 3. Select the 'DA Items' tab and verify that data changes are captured		
3	Start the OPC Configuration from the Programs Menu, and create a Data Access subscription to the installed server. • Click 'New Server' • Click 'Select…' next to the OPC Server Name input box and select the installed Data Access compliant OPC Server • Select COM as the connection type • Click 'Add DA Subscription' and enter an arbitrary subscription name • Add a remote unit ID either on the server or the subscription • Add all the dynamic elements to the Message Format • Click "Select…" next to the Item input box and select the desired item from the OPC Server • Add an appropriate Alarm State according to the Item Type enabling creation of an ACS event (can be seen using Softings Demo Client) • Save the configuration		

4	Disable the connection to ACS. • Edit %OPC Listener home%/config/listener.properties, and set 'listener.acs.endPoint' to a non-existent address. This will cause an NetException when trying to transmit to the ACS (ignore these exceptions)		
5	Ensure logging occurs at INFO level. • Edit %OPC Listener home%/config/log4net.config, and set the value of the level element of the two loggers to "ALL". The loggers are found at the bottom of the XML file		
6	Start the service and verify that events are captured. • Select the 'Service tab' • Click 'Start Service' • Wait a short while until some events/alarms have been generated (This is almost instantly with Technosoftware's Demo Server) • Click 'Stop Service' and wait until the OPC Configuration indicates the service is stopped • Check the log file %OPC Listener home%/logs/messagelog.log, and verify that events are received. An indication of a received event looks something like this: "… - Received OPC event: EVENT[<date>]{…}" • Ensure that all the dynamic elements of the Message Format were included.		

2.7 Test Suite: Data Access interface (from OPC Listener to ACS)

#	Test Case	Comment	Conducted by
1	Make sure the ACS accepts (and shows) all incoming alarms/events having the remote unit specified in the previous test • Create a remote unit • Create a customer • Add a location to the customer, select the remote unit, and apply the same remote unit id as the one used in the test above		
2	Make sure the OPC Listener does not listen for incoming alarms/events • Open (if not already open) OPC Configuration • Select the menu item "New" from the "File" menu • Save the empty configuration, overriding the old configuration		
3	Enable the connection to ACS. • Edit %OPC Listener home%/config/listener.properties, and set 'listener.acs.endPoint' to an existent address where an ACS is located • E.g. http://localhost:8080/acs/services/AlarmService		
4	Start the OPC Listener and verify that the existing alarms/events are transmitted to the ACS • Select the "Service" tab in the OPC Configuration • Click "Start Service" • Wait a short while until the existing alarms/events have been transmitted to the ACS. If it is a limited number of alarms (<100), this will not take much longer than a minute or so • Click "Stop Service" • Check the log file %OPC Listener home%/logs/messagelog.log, and verify that the events were transmitted successfully • Check the ACS web interface to verify that the events were received correctly		

2.8 Test Suite: Data Access interface (from OPC Server to ACS via OPC Listener)

#	Test Case	Comment	Conducted by
1	Cancel alarms already present in ACS web interface. • Select all the alarms received in the previous test, and click "Cancel"		
2	Start the OPC Configuration from the Programs Menu, and create a Data Access subscription to the installed server like previous. • Click 'New Server' • Click 'Select…' next to the OPC Server Name input box and select the installed Data Access compliant OPC Server • Select COM as the connection type • Click 'Add DA Subscription' and enter an arbitrary subscription name • Add a remote unit ID either on the server or the subscription • Click "Select…" next to the Item input box and select the desired item from the OPC Server • Add an appropriate Alarm State according to the Item Type enabling creation of an ACS event (can be seen using Softings Demo Client) • Save the configuration		
3	Start the OPC Listener and verify that the captured events/alarms are transmitted to the ACS. • Select the "Service" tab in the OPC Configuration • Click "Start Service" • Wait a short while until some events/alarms have been generated (This is almost instantly with Technosoftware's Demo Server) • Click "Stop Service" • Check the log file %OPC Listener home%/logs/messagelog.log, and verify that the events were transmitted successfully • Check the ACS web interface to verify that the events were received correctly		

2.9 Test Suite: Configuration tool

#	Test Case	Comment	Conducted by
1	Start OPC Listener Configuration Tool • Select the OPC Configuration from the OPC Listener folder in the Programs menu.		
2	Test menu items • Click 'Help'/'About OPC Listener Configuration…' and verify that an About Box dialog appears • Click on the different languages in the 'Language' menu an verify that the correct languages appear • Add a server by clicking on the 'Add Server' button. Click 'File'/'New' and verify that a new, empty, configuration is shown in the Configuration tab. • Add a server by clicking on the 'Add Server' button. Save the configuration using 'File'/'Save As…'. Verify that the file was saved by checking the file system. • Change a few things in the configuration and save the changes using 'File'/'Save'. Check the file system to verify that the changes indeed were saved. • Create a new, empty configuration by clicking 'File'/'New'. Open the just saved configuration by clicking 'File'/'Open' and selecting the file saved above. • Exit the OPC Listener Configuration by selecting 'File'/'Exit'.		
3	Test add/remove buttons • Click 'Add Server' and verify that a server is added. • Click 'Add A&E Subscription' and verify that an AE Subscription is added. • Click 'Add DA Subscription' and verify that a DA Subscription is added. • Click 'Remove Selected' and verify that the selected server/subscription is removed.		

4	Test start/stop service buttons • Select the 'Service' tab • Click 'Start Service' and verify that the service is started in 'My Computer'/'Manage'/'Services'/'ACS OPC Listener'. • Click 'Stop Service' and verify that the service is stopped in 'My Computer'/'Manage'/'Services'/'ACS OPC Listener'.		
5	Test Server buttons • Add a server to the configuration (in the Configuration tab) • Click 'Select…' next to OPC Server Name and verify that a new window is displayed listing all the OPC Servers available on the specified address. Click 'Ok'. • Enter an ACS Alarm Code and hit 'Add Alarm Code' and verify that a list element shows up in the list of Alarm Codes with the entered value.		
6	Test A&E Subscription buttons • Click on 'source', 'severity', 'message', 'condition', 'server', and 'subscription' and verify that the corresponding elements are added to the Message Format input field. • Click on 'formula' and verify that a formula field is added to the Message Format input field, and the cursor is placed between the parentheses. • Enter a range under the Severity Mappings group box, select a mapping, and click 'Add Mapping'. Verify that the mapping is added to the list in the format 'Range → Alarm Code'. • Try entering a range which is not allowed, and verify that an error is displayed. Valid ranges are single numbers between 0 and 1000 and ranges between 0 and 1000 where it ranges from a low to a high number. Range limits are separated by a dash (-). • Click on 'Select…' next to 'Sources', 'Areas' and 'Event Category Ids' and verify that a new windows is displayed listing all the appropriate elements (sources, areas and event category id's).		
7	Test DA Subscription buttons • Verify that the 'source', 'severity', 'message', 'condition', 'server', 'subscription', and 'formula' buttons work as specified above.		

	• Click on 'Select…' next to Item and verify that a new window is displayed listing all the items available on the selected server. • Switch Alarm States between 'Boolean', 'Single Value' and 'Range' and verify that the element right below changes appropriately (select box, input field, input fields with a dash between them).		
8	Test that no duplicates can be created in server / subscription names • Create a new Configuration and add a new Server. Select a server either by entering the name, or using the 'Select…' button. • Add another server and enter / select the same server. Once the input field containing the server name loses focus, verify that an error message is displayed. • For any one of the servers, perform the same task by adding two subscriptions and giving them the same subscription name. This should result in a similar error message being displayed.		
9	Test that the 'NONE' Alarm Code can not be deleted • For any Server, select the 'NONE' element in the list of Alarm Codes and hit the 'delete' key on the keyboard. Verify that nothing happens. Try deleting one of the other Alarm Codes and verify that they disappear from the list.		
10	Test that Severity Mappings can be deleted • Select any A&E Subscription and create a Severity Mapping. Select the mapping just created in the list, and hit the 'delete' key on the keyboard. Verify that the element is deleted from the list.		
11	Test that the 'dirty bit' is set once changes are made in the configuration • Create a new configuration, add at least one server and one subscription and save the configuration using 'File'/'Save As…'. Verify that the 'File'/'Save' menu item is now disabled. Change anything in the configuration and verify that the 'File'/'Save' menu item now becomes enabled. Do this test with all possible changes (checkboxes, radio buttons, input fields, etc.).		

2.10 Test Suite: Simultaneous access across user spaces

#	Test Case	Comment	Conducted by
1	Start OPC Server • Log in using the desired domain username (preferably a dedicated OPC user) • Start the server (as a service or as regular application)		
2	Configure OPC Listener (accessed from another user space) • Log in as another user than the one used above • Start OPC Configuration • Click "Add Server" • Click "Select…" next to the OPC Server Name input box and select the server from above • Create an appropriate subscription enabling capturing of events • Save configuration		
3	Ensure the OPC Listener is running as a different username than the OPC Server. • Right-click on 'My Computer', Select 'Manage'. Select 'Services' under 'Services and Applications' and locate the OPC Listener 'ACS OPC Listener' Service. • Right-click on 'ACS OPC Listener' and select 'Properties'. Select the 'Log On' tab and enter the username and password for a different user than the one used for the OPC Server.		
4	Start the OPC Listener, and verify that it started correctly (accessed from another user space) • Select the 'Service' tab in the OPC Listener Configuration Tool • Click 'Start Service' and wait until the progress bar disappears. • Open the log file 'systemlog.log' from '%OPC_LISTENER_HOME%/logs' • Ensure that no errors are present indicating that a connection to the server could not be established.		

2.11 Test Suite: Remote access

#	Test Case	Comment	Conducted by
1	Allow remote connections on server machine • Follow the instructions in the document "Using OPC via DCOM with Microsoft Windows XP Service Pack 2"		
2	Start OPC Server • Log in using the desired domain user name (preferably a dedicated OPC user) • Start the server (as a service or as regular application)		
3	Install OPC Listener on a remote machine • Set the Service User Name to be the same domain user as used on server machine.		
4	Configure OPC Listener • Start OPC Configuration • Click "Add Server" • Click "Select…" next to the OPC Server Name input box, enter the ip/dns-name of the remote machine hosting the OPC Server, and select the desired server • Create an appropriate subscription enabling capturing of events • Save configuration		
5	Start the OPC Listener and verify the remote access works • Select the "Service" tab • Click "Start Service" • Wait a short while until some events have been generated • Click "Stop Service" • Check the log file %OPC Listener home%/logs/messagelog.log, and verify that the events were transmitted successfully • Optionally check the ACS web interface to verify that the events were received correctly (if setup correctly in 'listener.properties')		

2.12 Test Suite: Uninstalling

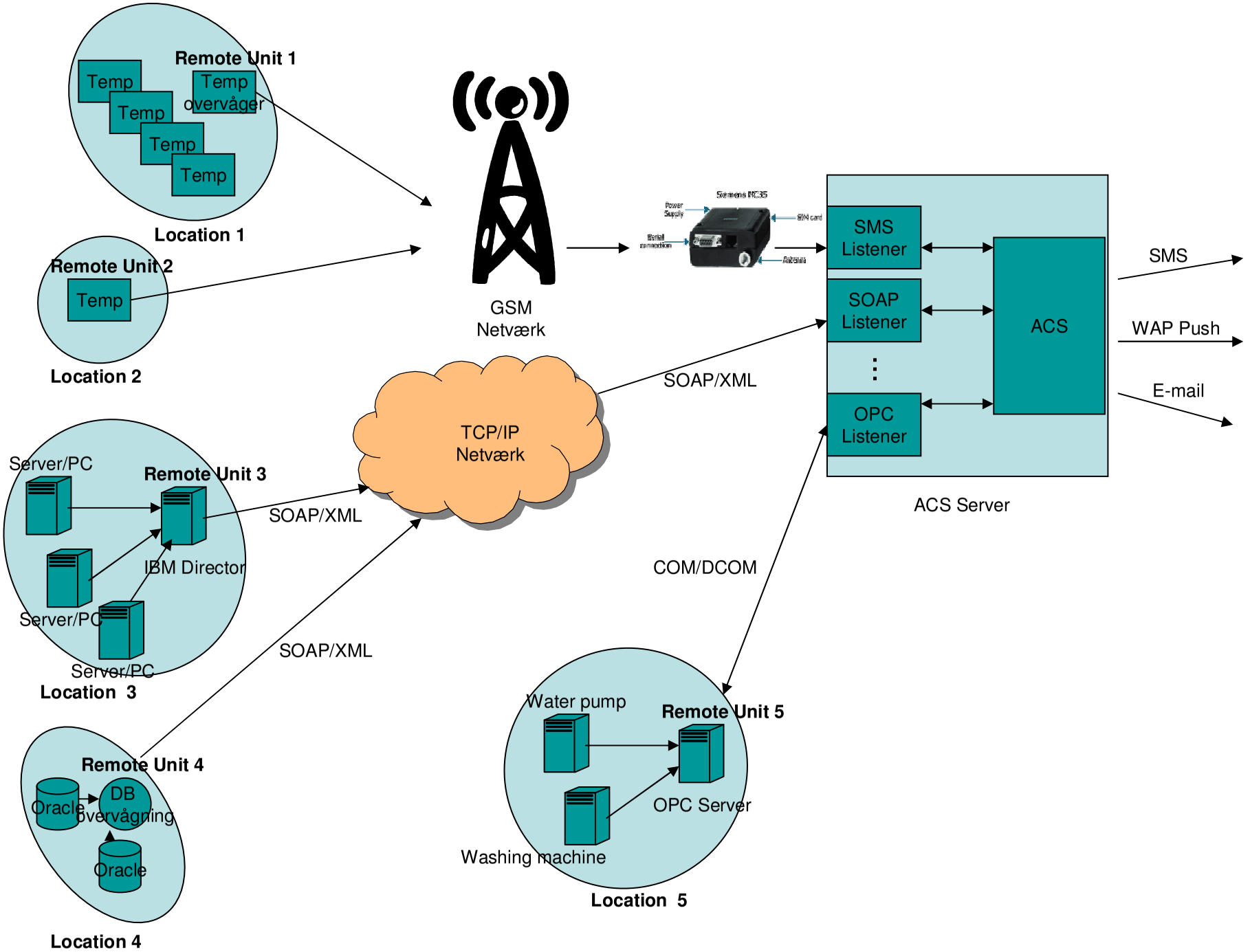
#	Test Case	Comment	Conducted by
1	Stop the OPC Listener • Open the OPC Listener Configuration Tool • Select the 'Service" tab • Click "Stop Service" if the Service is running		
2	Uninstall the OPC Listener • Open the Control Panel • Select "Add/Remove Programs" • Locate "OPC Listener" and select "Remove"		
3	Verify that the program files were removed • Open "%OPC Listener home%" and verify that only log files and configuration files remain. • Delete the directory if desired.		

3 TERMS AND ABBREVIATIONS

The following terms and abbreviations apply to this document:

Term	Description
DCOMCNFG	DCOM (Distributed Component Object Model) Configuration. This is used to set launch/access permissions for installed COM components. The file DCOMCNFG.EXE is usually located in C:\WINDOWS\SYSTEM32\ enabling it to be called anywhere.

Abbreviation	Definition
ACS	Alarm Control System
OPC	OLE (Object Linking and Embedding) for Process Control


Remote Unit 1
Temp
Temp
Temp
Temp
Temp overvåger
Location 1
Remote Unit 2
Temp
Location 2
GSM
Netværk
SMS
Listener
SOAP
Listener
OPC
Listener
ACS
ACS Server
SMS
WAP Push
E-mail
SOAP/XML
TCP/IP
Netværk
Server/PC
Remote Unit 3
IBM Director
Server/PC
Server/PC
Location 3
SOAP/XML
COM/DCOM
Remote Unit 4
Oracle
DB
overvågning
Oracle
Location 4
Water pump
Remote Unit 5
OPC Server
Washing machine
Location 5

```
using System;

namespace Zonith.Configuration.OpcListener
{
  /// <summary>
  /// Alarms & Events subscription
  /// </summary>
  public class OpcAeSubscriptionNode : OpcSubscriptionNode
  {
    private String messageFormat;
    public String MessageFormat { get { return messageFormat; } set { messageFormat = value; } }

    private bool filterEnabled = false;
    public bool FilterEnabled { get { return filterEnabled; } set { filterEnabled = value; } }

    private String sources;
    public String Sources { get { return sources; } set { sources = value; } }

    private String areas;
    public String Areas { get { return areas; } set { areas = value; } }

    private String eventType;
    public String EventType { get { return eventType; } set { eventType = value; } }

    private String categoryIds;
    public String CategoryIds { get { return categoryIds; } set { categoryIds = value; } }

    private String lowestSeverity;
    public String LowestSeverity { get { return lowestSeverity; } set { lowestSeverity = value; } }

    private String highestSeverity;
    public String HighestSeverity { get { return highestSeverity; } set { highestSeverity = value; } }

    private OpcSeverityMapping[] severityMappings = {};
    public OpcSeverityMapping[] SeverityMappings { get { return severityMappings; } set { severityMappings = value; } }

    public OpcAeSubscriptionNode(String name) : base(name) {}
    public OpcAeSubscriptionNode() {}
  }
}

```

```
using System;

namespace Zonith.Configuration.OpcListener
{
  /// <summary>
  /// Data Access subscription
  /// </summary>
  public class OpcDaSubscriptionNode : OpcSubscriptionNode
  {
    private String messageFormat;
    public String MessageFormat { get { return messageFormat; } set { messageFormat = value; } }

    private String item;
    public String Item { get { return item; } set { item = value; } }

    private AlarmStateTypes alarmStateType = AlarmStateTypes.BOOLEAN;
    public AlarmStateTypes AlarmStateType { get { return alarmStateType; } set { alarmStateType = value; } }

    private String alarmState;
    public String AlarmState { get { return alarmState; } set { alarmState = value; } }

    private String acsAlarmCode;
    public String AcsAlarmCode { get { return acsAlarmCode; } set { acsAlarmCode = value; } }

    public OpcDaSubscriptionNode(String name) : base(name) {}
    public OpcDaSubscriptionNode() {}
  }
}

```

```
using System;

namespace Zonith.Configuration.OpcListener
{
  /// <summary>
  /// Summary description for OPCServerNode.
  /// </summary>
  public class OpcServerNode : System.Windows.Forms.TreeNode
  {
    private String opcServerName;
    public String OpcServerName { get { return opcServerName; } set { opcServerName = value; base.Text = value; } }

    private String opcServerHost;
    public String OpcServerHost { get { return opcServerHost; } set { opcServerHost = value; } }

    private String acsDefaultUnitId;
    public String AcsDefaultUnitId { get { return acsDefaultUnitId; } set { acsDefaultUnitId = value; } }

    private String acsDefaultAreaGroups;
    public String AcsDefaultAreaGroups { get { return acsDefaultAreaGroups; } set { acsDefaultAreaGroups = value; } }

    private bool useCom = true;
    public bool UseCom { get { return useCom; } set { useCom = value; } }

    private String[] acsAlarmCodes = {};
    public String[] AcsAlarmCodes { get { return acsAlarmCodes; } set { acsAlarmCodes = value; } }

    public OpcServerNode(String name) : base(name) { OpcServerName = name; }
    public OpcServerNode() {}
  }
}

```

```
using System;

namespace Zonith.Configuration.OpcListener
{
  /// <summary>
  /// Summary description for OPCSeveritMapping.
  /// </summary>
  public class OpcSeverityMapping
  {
    private String opcRange = null;
    private String acsName = null;

    public String OpcRange { get  { return opcRange; } set  { this.opcRange = value; } }
    public String AcsName { get { return acsName; } set { this.acsName = value; } }

    public OpcSeverityMapping() {}

    public OpcSeverityMapping(String range, String name)
    {
      this.OpcRange = range;
      this.AcsName = name;
    }

    public override string ToString()
    {
      return OpcRange + " --> " + AcsName;
    }
  }
}

```

```
using System;

namespace Zonith.Configuration.OpcListener
{
  /// <summary>
  /// Abstract subscription class
  /// </summary>
  public abstract class OpcSubscriptionNode : System.Windows.Forms.TreeNode
  {
    private bool enabled = true;
    public bool Enabled { get { return enabled; } set { enabled = value; } }

    private String subscriptionName;
    public String SubscriptionName { get { return subscriptionName; } set { subscriptionName = value; base.Text = value; } }

    private String acsUnitId;
    public String AcsUnitId { get { return acsUnitId; } set { acsUnitId = value; } }

    private String acsAreaGroups;
    public String AcsAreaGroups { get { return acsAreaGroups; } set { acsAreaGroups = value; } }

    protected OpcSubscriptionNode(String name) : base(name) { SubscriptionName = name; }
    protected OpcSubscriptionNode() {}
  }
}

```

```
using System;
using System.Xml;
using System.Windows.Forms;
using System.Collections;

namespace Zonith.Configuration.OpcListener
{
  public enum OpcSubscriptionTypes { AE, DA };
  public enum AlarmStateTypes { BOOLEAN, SINGLE, RANGE };

  /// <summary>
  /// Summary description for XmlConfiguration.
  /// </summary>
  public class XmlConfiguration
  {
    private static bool DEBUG = false;

    private const String ELEM_CONFIGURATION = "configuration";
    private const String ELEM_SERVER = "server";
    private const String ELEM_SERVER_NAME = "name";
    private const String ELEM_SERVER_HOST = "host";
    private const String ELEM_SERVER_USE_COM_METHOD = "use_com";
    private const String ELEM_SERVER_ACS_DEFAULT_UNIT_ID = "acs_default_unit_id";
    private const String ELEM_SERVER_ACS_DEFAULT_AREA_GROUPS = "acs_default_areagroups";
    private const String ELEM_SERVER_ACS_ALARM_CODES = "acs_alarm_codes";
    private const String ELEM_SERVER_ACS_ALARM_CODE = "alarm_code";
    private const String ELEM_SERVER_SUBSCRIPTIONS = "subscriptions";
    private const String ELEM_SUBSCRIPTION = "subscription";
    private const String ATTR_SUBSCRIPTION_TYPE = "type";
    private const String ATTR_SUBSCRIPTION_ENABLED = "enabled";
    private const String ELEM_SUBSCRIPTION_NAME = "name";
    private const String ELEM_SUBSCRIPTION_ACS_UNIT_ID = "unit_id";
    private const String ELEM_SUBSCRIPTION_ACS_AREA_GROUPS = "areagroups";
    private const String ELEM_SUBSCRIPTION_MESSAGE_FORMAT = "message_format";
    private const String ELEM_SUBSCRIPTION_FILTER = "filter";
    private const String ELEM_SUBSCRIPTION_FILTER_ENABLED = "enabled";
    private const String ELEM_SUBSCRIPTION_FILTER_SOURCES = "sources";
    private const String ELEM_SUBSCRIPTION_FILTER_AREAS = "areas";
    private const String ELEM_SUBSCRIPTION_FILTER_EVENT_TYPE = "event_type";
    private const String ELEM_SUBSCRIPTION_FILTER_CATEGORY_IDS = "category_ids";
    private const String ELEM_SUBSCRIPTION_FILTER_LOWEST_SEVERITY = "lowest_severity";
    private const String ELEM_SUBSCRIPTION_FILTER_HIGHEST_SEVERITY = "highest_severity";
    private const String ELEM_SUBSCRIPTION_SEVERITY_MAPPINGS = "severity_mappings";
    private const String ELEM_SUBSCRIPTION_SEVERITY_MAPPING = "mapping";
    private const String ELEM_SUBSCRIPTION_MAPPING_OPC_RANGE = "opc_range";
    private const String ELEM_SUBSCRIPTION_MAPPING_ACS_NAME = "acs_name";
    private const String ELEM_SUBSCRIPTION_ITEM = "item";
```

```
48:     private const String ELEM_SUBSCRIPTION_ALARM_STATE = "alarm_state";
49:     private const String ATTR_SUBSCRIPTION_ALARM_STATE_TYPE = "type";
50:     private const String ELEM_SUBSCRIPTION_ACS_ALARM_CODE = "acs_alarm_code";
51:
52:     public static void WriteConfigurationToFile(String path, object[] servers)
53:     {
54:       XmlTextWriter writer = null;
55:       try
56:       {
57:         writer = new XmlTextWriter(path, System.Text.Encoding.UTF8);
58:         writer.Formatting = Formatting.Indented;
59:         writer.WriteStartDocument();
60:         writer.WriteStartElement(ELEM_CONFIGURATION);
61:
62:         foreach (OpcServerNode sNode in servers)
63:         {
64:           writer.WriteStartElement(ELEM_SERVER);
65:
66:           writer.WriteStartElement(ELEM_SERVER_NAME);
67:           writer.WriteString(sNode.OpcServerName);
68:           writer.WriteEndElement();
69:
70:           writer.WriteStartElement(ELEM_SERVER_HOST);
71:           writer.WriteString(sNode.OpcServerHost);
72:           writer.WriteEndElement();
73:
74:           writer.WriteStartElement(ELEM_SERVER_USE_COM_METHOD);
75:           writer.WriteString(sNode.UseCom.ToString());
76:           writer.WriteEndElement();
77:
78:           writer.WriteStartElement(ELEM_SERVER_ACS_DEFAULT_UNIT_ID);
79:           writer.WriteString(sNode.AcsDefaultUnitId);
80:           writer.WriteEndElement();
81:
82:           writer.WriteStartElement(ELEM_SERVER_ACS_DEFAULT_AREA_GROUPS);
83:           writer.WriteString(sNode.AcsDefaultAreaGroups);
84:           writer.WriteEndElement();
85:
86:           writer.WriteStartElement(ELEM_SERVER_ACS_ALARM_CODES);
87:           foreach (String code in sNode.AcsAlarmCodes)
88:           {
89:             writer.WriteStartElement(ELEM_SERVER_ACS_ALARM_CODE);
90:             writer.WriteString(code);
91:             writer.WriteEndElement();
92:           }
93:           writer.WriteEndElement();
94:
```

```
 95:           writer.WriteStartElement(ELEM_SERVER_SUBSCRIPTIONS);
 96:           IEnumerator e2 = sNode.Nodes.GetEnumerator();
 97:           while (e2.MoveNext())
 98:           {
 99:             OpcSubscriptionNode s = (OpcSubscriptionNode) e2.Current;
100:             if (s is OpcAeSubscriptionNode)
101:               WriteAeSubscription(writer, (OpcAeSubscriptionNode) s);
102:             else if (s is OpcDaSubscriptionNode)
103:               WriteDaSubscription(writer, (OpcDaSubscriptionNode) s);
104:           }
105:           writer.WriteEndElement();
106:
107:           writer.WriteEndElement();
108:         }
109:
110:         writer.WriteEndElement();
111:         writer.WriteEndDocument();
112:         writer.Flush();
113:         writer.Close();
114:       }
115:       catch (XmlException e)
116:       {
117:         throw e;
118:       }
119:       finally
120:       {
121:         if(writer != null)
122:           writer.Close();
123:       }
124:     }
125:
126:     private static void WriteAeSubscription(XmlTextWriter writer, OpcAeSubscriptionNode node)
127:     {
128:       writer.WriteStartElement(ELEM_SUBSCRIPTION);
129:       writer.WriteAttributeString(ATTR_SUBSCRIPTION_TYPE, OpcSubscriptionTypes.AE.ToString());
130:       writer.WriteAttributeString(ATTR_SUBSCRIPTION_ENABLED, node.Enabled.ToString());
131:
132:       writer.WriteStartElement(ELEM_SUBSCRIPTION_NAME);
133:       writer.WriteString(node.SubscriptionName);
134:       writer.WriteEndElement();
135:
136:       writer.WriteStartElement(ELEM_SUBSCRIPTION_ACS_UNIT_ID);
137:       writer.WriteString(node.AcsUnitId);
138:       writer.WriteEndElement();
139:
140:       writer.WriteStartElement(ELEM_SUBSCRIPTION_ACS_AREA_GROUPS);
141:       writer.WriteString(node.AcsAreaGroups);
```

```
142:       writer.WriteEndElement();
143:
144:       writer.WriteStartElement(ELEM_SUBSCRIPTION_MESSAGE_FORMAT);
145:       writer.WriteString(node.MessageFormat);
146:       writer.WriteEndElement();
147:
148:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER);
149:
150:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER_ENABLED);
151:       writer.WriteString(node.FilterEnabled.ToString());
152:       writer.WriteEndElement();
153:
154:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER_SOURCES);
155:       writer.WriteString(node.Sources);
156:       writer.WriteEndElement();
157:
158:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER_AREAS);
159:       writer.WriteString(node.Areas);
160:       writer.WriteEndElement();
161:
162:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER_EVENT_TYPE);
163:       writer.WriteString(node.EventType);
164:       writer.WriteEndElement();
165:
166:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER_CATEGORY_IDS);
167:       writer.WriteString(node.CategoryIds);
168:       writer.WriteEndElement();
169:
170:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER_LOWEST_SEVERITY);
171:       writer.WriteString(node.LowestSeverity);
172:       writer.WriteEndElement();
173:
174:       writer.WriteStartElement(ELEM_SUBSCRIPTION_FILTER_HIGHEST_SEVERITY);
175:       writer.WriteString(node.HighestSeverity);
176:       writer.WriteEndElement();
177:
178:       writer.WriteEndElement();
179:
180:       writer.WriteStartElement(ELEM_SUBSCRIPTION_SEVERITY_MAPPINGS);
181:       foreach (OpcSeverityMapping m in node.SeverityMappings)
182:       {
183:         writer.WriteStartElement(ELEM_SUBSCRIPTION_SEVERITY_MAPPING);
184:
185:         writer.WriteStartElement(ELEM_SUBSCRIPTION_MAPPING_OPC_RANGE);
186:         writer.WriteString(m.OpcRange);
187:         writer.WriteEndElement();
188:
```

```
189:           writer.WriteStartElement(ELEM_SUBSCRIPTION_MAPPING_ACS_NAME);
190:           writer.WriteString(m.AcsName);
191:           writer.WriteEndElement();
192:
193:           writer.WriteEndElement();
194:         }
195:         writer.WriteEndElement();
196:
197:         writer.WriteEndElement();
198:       }
199:
200:       private static void WriteDaSubscription(XmlTextWriter writer, OpcDaSubscriptionNode node)
201:       {
202:         writer.WriteStartElement(ELEM_SUBSCRIPTION);
203:         writer.WriteAttributeString(ATTR_SUBSCRIPTION_TYPE, OpcSubscriptionTypes.DA.ToString());
204:         writer.WriteAttributeString(ATTR_SUBSCRIPTION_ENABLED, node.Enabled.ToString());
205:
206:         writer.WriteStartElement(ELEM_SUBSCRIPTION_NAME);
207:         writer.WriteString(node.SubscriptionName);
208:         writer.WriteEndElement();
209:
210:         writer.WriteStartElement(ELEM_SUBSCRIPTION_ACS_UNIT_ID);
211:         writer.WriteString(node.AcsUnitId);
212:         writer.WriteEndElement();
213:
214:         writer.WriteStartElement(ELEM_SUBSCRIPTION_ACS_AREA_GROUPS);
215:         writer.WriteString(node.AcsAreaGroups);
216:         writer.WriteEndElement();
217:
218:         writer.WriteStartElement(ELEM_SUBSCRIPTION_MESSAGE_FORMAT);
219:         writer.WriteString(node.MessageFormat);
220:         writer.WriteEndElement();
221:
222:         writer.WriteStartElement(ELEM_SUBSCRIPTION_ITEM);
223:         writer.WriteString(node.Item);
224:         writer.WriteEndElement();
225:
226:         writer.WriteStartElement(ELEM_SUBSCRIPTION_ALARM_STATE);
227:         writer.WriteAttributeString(ATTR_SUBSCRIPTION_ALARM_STATE_TYPE, node.AlarmStateType.ToString());
228:         writer.WriteString(node.AlarmState);
229:         writer.WriteEndElement();
230:
231:         writer.WriteStartElement(ELEM_SUBSCRIPTION_ACS_ALARM_CODE);
232:         writer.WriteString(node.AcsAlarmCode);
233:         writer.WriteEndElement();
234:
235:         writer.WriteEndElement();
```

```
236:     }
237:
238:     public static OpcServerNode[] ReadConfigurationFromFile(String path)
239:     {
240:       if (DEBUG)
241:         Console.WriteLine("Reading configuration...");
242:
243:       ArrayList serverList = new ArrayList();
244:       OpcServerNode serverNode;
245:       XmlTextReader reader = null;
246:       try
247:       {
248:         reader = new XmlTextReader(path);
249:         reader.XmlResolver = null;  // ignore DTD
250:         reader.WhitespaceHandling = WhitespaceHandling.None;
251:         reader.MoveToContent();
252:         if (!(reader.Name == ELEM_CONFIGURATION))
253:           throw new ECorruptedConfigFileException("No <" + ELEM_CONFIGURATION + "> node (root element) defined");
254:
255:         while (reader.Read() && reader.Name == ELEM_SERVER)
256:         {
257:           serverNode = new OpcServerNode();
258:
259:           // Read the next element
260:           reader.Read();
261:
262:           while (reader.Name != ELEM_SERVER)
263:           {
264:             switch (reader.Name)
265:             {
266:               case ELEM_SERVER_NAME:
267:                 serverNode.OpcServerName = reader.ReadElementString();
268:                 break;
269:               case ELEM_SERVER_HOST:
270:                 serverNode.OpcServerHost = reader.ReadElementString();
271:                 break;
272:               case ELEM_SERVER_USE_COM_METHOD:
273:                 serverNode.UseCom = Convert.ToBoolean(reader.ReadElementString());
274:                 break;
275:               case ELEM_SERVER_ACS_DEFAULT_UNIT_ID:
276:                 serverNode.AcsDefaultUnitId = reader.ReadElementString();
277:                 break;
278:               case ELEM_SERVER_ACS_DEFAULT_AREA_GROUPS:
279:                 serverNode.AcsDefaultAreaGroups = reader.ReadElementString();
280:                 break;
281:               case ELEM_SERVER_ACS_ALARM_CODES:
282:                 GetAcsAlarmCodes(reader, serverNode);
```

```
283:                 break;
284:               case ELEM_SERVER_SUBSCRIPTIONS:
285:                 GetSubscriptions(reader, serverNode);
286:                 break;
287:               default:
288:                 throw new ECorruptedConfigFileException("Unknown element <" + reader.Name + ">");
289:             }
290:           }
291:
292:           if (IsInCollection(serverList.GetEnumerator(), serverNode.OpcServerName))
293:             throw new ECorruptedConfigFileException("Multiple servers with same name '" + serverNode.OpcServerName + "'
                 defined");
294:
295:           if (DEBUG)
296:             Console.WriteLine("Server " + serverNode.Text + " has " + serverNode.AcsAlarmCodes.Length + " acs alarm codes");
297:           serverList.Add(serverNode);
298:         }
299:         OpcServerNode[] sList1 = new OpcServerNode[serverList.Count];
300:         serverList.CopyTo(sList1);
301:         return sList1;
302:       }
303:       catch (XmlException)
304:       {
305:         throw new ECorruptedConfigFileException("Invalid XML format!");
306:       }
307:       catch (FormatException)
308:       {
309:         throw new ECorruptedConfigFileException("Illegal truth value in <" + ELEM_SERVER_USE_COM_METHOD + "> or <" +
               ELEM_SUBSCRIPTION_FILTER_ENABLED + ">");
310:       }
311:       finally
312:       {
313:         if (reader != null)
314:           reader.Close();
315:       }
316:     }
317:
318:     private static void GetAcsAlarmCodes(XmlTextReader reader, OpcServerNode serverNode)
319:     {
320:       if (DEBUG)
321:         Console.WriteLine("Reading acs alarm codes : type is: " + reader.NodeType + " name is: " + reader.Name);
322:       ArrayList acsCodes;
323:
324:       acsCodes = new ArrayList();
325:
326:       reader.Read();
327:       while (reader.Name == ELEM_SERVER_ACS_ALARM_CODE)
```

```
328:           acsCodes.Add(reader.ReadElementString());
329:
330:         if (DEBUG)
331:           Console.WriteLine("Read " + acsCodes.Count + " alarm codes : type is: " + reader.NodeType + " : name is: " + reader.
                 Name);
332:
333:         reader.Read();
334:         serverNode.AcsAlarmCodes = new String[acsCodes.Count];
335:         acsCodes.CopyTo(serverNode.AcsAlarmCodes);
336:       }
337:
338:       private static void GetSeverityMappings(XmlTextReader reader, OpcAeSubscriptionNode subscriptionNode)
339:       {
340:         if (DEBUG)
341:           Console.WriteLine("Reading severity mappings : type is: " + reader.NodeType);
342:         ArrayList mappingList;
343:         OpcSeverityMapping mapping;
344:
345:         bool addMappings = false;
346:
347:         mappingList = new ArrayList();
348:         while (reader.Read() && reader.Name == ELEM_SUBSCRIPTION_SEVERITY_MAPPING && reader.NodeType == XmlNodeType.Element)
349:         {
350:           if (DEBUG)
351:             Console.WriteLine("Reading mapping : type is: " + reader.NodeType);
352:           mapping = new OpcSeverityMapping();
353:
354:           // read the next element
355:           reader.Read();
356:
357:           while (reader.Name != ELEM_SUBSCRIPTION_SEVERITY_MAPPING)
358:           {
359:             if (DEBUG)
360:               Console.WriteLine("Reading mapping element (" + reader.Name + ") : type is: " + reader.NodeType);
361:             switch (reader.Name)
362:             {
363:               case ELEM_SUBSCRIPTION_MAPPING_OPC_RANGE:
364:                 mapping.OpcRange = reader.ReadElementString();
365:                 break;
366:               case ELEM_SUBSCRIPTION_MAPPING_ACS_NAME:
367:                 mapping.AcsName = reader.ReadElementString();
368:                 break;
369:               default:
370:                 throw new ECorruptedConfigFileException("Unknown element <" + reader.Name + ">");
371:             }
372:           }
373:           if (DEBUG)
```

```
374:           Console.WriteLine("Reading mapping element (" + reader.Name + ") : type is: " + reader.NodeType);
375:         if (DEBUG)
376:           Console.WriteLine("Adding mapping (" + mapping + ")");
377:         mappingList.Add(mapping);
378:         addMappings = true;
379:       }
380:       if (addMappings)
381:       {
382:         if (DEBUG)
383:           Console.WriteLine("Copying " + mappingList.Count + " mappings to server");
384:         subscriptionNode.SeverityMappings = new OpcSeverityMapping[mappingList.Count];
385:         mappingList.CopyTo(subscriptionNode.SeverityMappings);
386:       }
387:     }
388:
389:     private static void GetSubscriptions(XmlTextReader reader, OpcServerNode serverNode)
390:     {
391:       if (DEBUG)
392:         Console.WriteLine("Reading subscriptions... : type is: " + reader.NodeType);
393:       ArrayList subscriptionList;
394:       OpcSubscriptionNode subscriptionNode;
395:
396:       subscriptionList = new ArrayList();
397:       while (reader.Read() && reader.Name == ELEM_SUBSCRIPTION)
398:       {
399:         String type = reader.GetAttribute(ATTR_SUBSCRIPTION_TYPE);
400:         String enabled = reader.GetAttribute(ATTR_SUBSCRIPTION_ENABLED);
401:
402:         if (DEBUG)
403:           Console.WriteLine("Read subscription type: " + type);
404:
405:         if (type.ToLower() == OpcSubscriptionTypes.AE.ToString().ToLower())
406:         {
407:           subscriptionNode = new OpcAeSubscriptionNode();
408:           ReadAeSubscription(reader, (OpcAeSubscriptionNode) subscriptionNode);
409:         }
410:         else if (type.ToLower() == OpcSubscriptionTypes.DA.ToString().ToLower())
411:         {
412:           subscriptionNode = new OpcDaSubscriptionNode();
413:           ReadDaSubscription(reader, (OpcDaSubscriptionNode) subscriptionNode);
414:         }
415:         else
416:           throw new ECorruptedConfigFileException("Wrong subscription type [" + type + "] specified");
417:
418:         try
419:         {
420:           subscriptionNode.Enabled = Convert.ToBoolean(enabled);
```

```
421:         }
422:         catch (FormatException)
423:         {
424:           throw new ECorruptedConfigFileException("Illegal truth value in attribute '" + ATTR_SUBSCRIPTION_ENABLED + "'");
425:         }
426:
427:         if (IsInCollection(subscriptionList.GetEnumerator(), subscriptionNode.SubscriptionName))
428:           throw new ECorruptedConfigFileException("Multiple subscriptions with same name [" + subscriptionNode.
                SubscriptionName + "] defined for the same server [" + serverNode.OpcServerName + "]");
429:
430:         subscriptionList.Add(subscriptionNode);
431:       }
432:
433:       OpcSubscriptionNode[] sList = new OpcSubscriptionNode[subscriptionList.Count];
434:       subscriptionList.CopyTo(sList);
435:       serverNode.Nodes.AddRange(sList);
436:     }
437:
438:     private static void ReadAeSubscription(XmlTextReader reader, OpcAeSubscriptionNode subscriptionNode)
439:     {
440:       // read the next element
441:       reader.Read();
442:
443:       while (reader.Name != ELEM_SUBSCRIPTION)
444:       {
445:         switch (reader.Name)
446:         {
447:           case ELEM_SUBSCRIPTION_NAME:
448:             subscriptionNode.SubscriptionName = reader.ReadElementString();
449:             break;
450:           case ELEM_SUBSCRIPTION_ACS_UNIT_ID:
451:             subscriptionNode.AcsUnitId = reader.ReadElementString();
452:             break;
453:           case ELEM_SUBSCRIPTION_ACS_AREA_GROUPS:
454:             subscriptionNode.AcsAreaGroups = reader.ReadElementString();
455:             break;
456:           case ELEM_SUBSCRIPTION_MESSAGE_FORMAT:
457:             subscriptionNode.MessageFormat = reader.ReadElementString();
458:             break;
459:           case ELEM_SUBSCRIPTION_FILTER:
460:             GetFilter(reader, subscriptionNode);
461:             break;
462:           case ELEM_SUBSCRIPTION_SEVERITY_MAPPINGS:
463:             GetSeverityMappings(reader, subscriptionNode);
464:             break;
465:           default:
466:             throw new ECorruptedConfigFileException("Unknown element <" + reader.Name + "> in subscription");
```

```
467:         }
468:       }
469:     }
470:
471:     private static void ReadDaSubscription(XmlTextReader reader, OpcDaSubscriptionNode subscriptionNode)
472:     {
473:       // read the next element
474:       reader.Read();
475:
476:       while (reader.Name != ELEM_SUBSCRIPTION)
477:       {
478:         switch (reader.Name)
479:         {
480:           case ELEM_SUBSCRIPTION_NAME:
481:             subscriptionNode.SubscriptionName = reader.ReadElementString();
482:             break;
483:           case ELEM_SUBSCRIPTION_ACS_UNIT_ID:
484:             subscriptionNode.AcsUnitId = reader.ReadElementString();
485:             break;
486:           case ELEM_SUBSCRIPTION_ACS_AREA_GROUPS:
487:             subscriptionNode.AcsAreaGroups = reader.ReadElementString();
488:             break;
489:           case ELEM_SUBSCRIPTION_MESSAGE_FORMAT:
490:             subscriptionNode.MessageFormat = reader.ReadElementString();
491:             break;
492:           case ELEM_SUBSCRIPTION_ITEM:
493:             subscriptionNode.Item = reader.ReadElementString();
494:             break;
495:           case ELEM_SUBSCRIPTION_ALARM_STATE:
496:             String type = reader.GetAttribute(ATTR_SUBSCRIPTION_ALARM_STATE_TYPE);
497:             if (type.ToLower() == AlarmStateTypes.BOOLEAN.ToString().ToLower())
498:               subscriptionNode.AlarmStateType = AlarmStateTypes.BOOLEAN;
499:             else if (type.ToLower() == AlarmStateTypes.RANGE.ToString().ToLower())
500:               subscriptionNode.AlarmStateType = AlarmStateTypes.RANGE;
501:             else if (type.ToLower() == AlarmStateTypes.SINGLE.ToString().ToLower())
502:               subscriptionNode.AlarmStateType = AlarmStateTypes.SINGLE;
503:             subscriptionNode.AlarmState = reader.ReadElementString();
504:             break;
505:           case ELEM_SUBSCRIPTION_ACS_ALARM_CODE:
506:             subscriptionNode.AcsAlarmCode = reader.ReadElementString();
507:             break;
508:           default:
509:             throw new ECorruptedConfigFileException("Unknown element <" + reader.Name + "> in subscription");
510:         }
511:       }
512:     }
513:
```

```
514:     private static void GetFilter(XmlTextReader reader, OpcAeSubscriptionNode subscriptionNode)
515:     {
516:       if (DEBUG)
517:         Console.WriteLine("Reading filter : type is: " + reader.NodeType);
518:       // read next element
519:       reader.Read();
520:
521:       while (reader.Name != ELEM_SUBSCRIPTION_FILTER)
522:       {
523:         switch (reader.Name)
524:         {
525:           case ELEM_SUBSCRIPTION_FILTER_ENABLED:
526:             subscriptionNode.FilterEnabled = Convert.ToBoolean(reader.ReadElementString());
527:             break;
528:           case ELEM_SUBSCRIPTION_FILTER_SOURCES:
529:             subscriptionNode.Sources = reader.ReadElementString();
530:             break;
531:           case ELEM_SUBSCRIPTION_FILTER_AREAS:
532:             subscriptionNode.Areas = reader.ReadElementString();
533:             break;
534:           case ELEM_SUBSCRIPTION_FILTER_EVENT_TYPE:
535:             subscriptionNode.EventType = reader.ReadElementString();
536:             break;
537:           case ELEM_SUBSCRIPTION_FILTER_CATEGORY_IDS:
538:             subscriptionNode.CategoryIds = reader.ReadElementString();
539:             break;
540:           case ELEM_SUBSCRIPTION_FILTER_LOWEST_SEVERITY:
541:             subscriptionNode.LowestSeverity = reader.ReadElementString();
542:             break;
543:           case ELEM_SUBSCRIPTION_FILTER_HIGHEST_SEVERITY:
544:             subscriptionNode.HighestSeverity = reader.ReadElementString();
545:             break;
546:           default:
547:             throw new ECorruptedConfigFileException("Unknown element <" + reader.Name + ">");
548:         }
549:       }
550:
551:       // read "</filter>"
552:       reader.Read();
553:     }
554:
555:     public static bool IsInCollection(TreeNodeCollection collection, String name)
556:     {
557:       IEnumerator enumerator = collection.GetEnumerator();
558:       while (enumerator.MoveNext())
559:       {
560:         TreeNode node = (TreeNode) enumerator.Current;
```

```
561:         if (node.Text.Equals(name))
562:           return true;
563:       }
564:       return false;
565:     }
566:
567:     public static bool IsInCollection(TreeNodeCollection collection, String name, int number)
568:     {
569:       int count = 0;
570:       IEnumerator enumerator = collection.GetEnumerator();
571:       while (enumerator.MoveNext())
572:       {
573:         TreeNode node = (TreeNode) enumerator.Current;
574:         if (node.Text.Equals(name))
575:           count++;
576:         if (count == number)
577:           return true;
578:       }
579:       return false;
580:     }
581:
582:
583:     public static bool IsInCollection(IEnumerator enumerator, String name)
584:     {
585:       while (enumerator.MoveNext())
586:       {
587:         TreeNode node = (TreeNode) enumerator.Current;
588:         if (node.Text.Equals(name))
589:           return true;
590:       }
591:       return false;
592:     }
593:
594:     public static bool IsInCollection(IEnumerator enumerator, String name, int number)
595:     {
596:       int count = 0;
597:       while (enumerator.MoveNext())
598:       {
599:         TreeNode node = (TreeNode) enumerator.Current;
600:         if (node.Text.Equals(name))
601:           count++;
602:         if (count == number)
603:           return true;
604:       }
605:       return false;
606:     }
607:   }
```

```
608:
609:   public class ECorruptedConfigFileException : Exception
610:   {
611:     public ECorruptedConfigFileException(String msg) : base(msg) {}
612:   }
613:
614: }
615:
```

```
using System;
using System.Collections;
using System.Threading;
using Zonith.Listener.Core.Util;
using Zonith.Listener.Core.Interfaces;
using log4net;

namespace Zonith.Listener.Core.FrontEnd
{
  /// <summary>
  /// An abstract Front End implementing the default action of the
  /// FireEvent method.
  /// </summary>
  public abstract class AbstractFrontEnd : IFrontEnd
  {
    /// <summary>
    /// References to the message log
    /// </summary>
    private ILog m_messageLog = LogManager.GetLogger(Constants.MESSAGE_LOG);

    /// <summary>
    /// Reference to the listener
    /// </summary>
    private AbstractListener m_listener;

    /// <summary>
    /// Constructor taking a reference to the listener creating the
    /// front/back-ends as parameter
    /// </summary>
    /// <param name="listener">Reference to the listener</param>
    public AbstractFrontEnd(AbstractListener listener)
    {
      this.m_listener = listener;
    }

    #region IFrontEnd Members

    /// <summary>
    /// <see cref="Zonith.Listener.Core.Interfaces.IFrontEnd.Initialize()"/>
    /// </summary>
    public virtual void Initialize() {}

    /// <summary>
    /// <see cref="Zonith.Listener.Core.Interfaces.IFrontEnd.Terminate()"/>
    /// </summary>
    public virtual void Terminate() {}

```

```
48:     /// <summary>
49:     /// <see cref="Zonith.Listener.Core.Interfaces.IFrontEnd.Start()"/>
50:     /// </summary>
51:     public virtual void Start() {}
52:
53:     /// <summary>
54:     /// <see cref="Zonith.Listener.Core.Interfaces.IFrontEnd.FireEvent(IEvent theEvent)"/>
55:     /// </summary>
56:     public void FireEvent(IEvent theEvent)
57:     {
58:       lock(this)
59:       {
60:         // If the event is an OPC event, log the reception of the event at INFO level
61:         if (theEvent is IOpcEvent)
62:         {
63:           IOpcEvent opcEvent = (IOpcEvent) theEvent;
64:           m_messageLog.Info("Received OPC event: " + opcEvent);
65:         }
66:         m_listener.FireEvent(theEvent);
67:       }
68:     }
69:
70:     #endregion
71:   }
72: }
73:
```

```
1: using System;
2: using Zonith.Listener.Core.BackEnd;
3: using Zonith.Listener.Core.Interfaces;
4: using Zonith.Listener.Core.Util;
5: using log4net;
6:
7: namespace Zonith.Listener.Core
8: {
9:   /// <summary>
10:   /// An abstract Listener performing all the necessary setups. Subclasses should only
11:   /// override the hook GetFrontEnd(). The Back End can naturally be changed by
12:   /// overriding the hook GetBackEnd(). If this is not overridden the DefaultBackEnd is
13:   /// used.
14:   /// </summary>
15:   public abstract class AbstractListener
16:   {
17:     /// <summary>
18:     /// Reference to the Back End
19:     /// </summary>
20:     IBackEnd m_backEnd;
21:
22:     /// <summary>
23:     /// Reference to the Front End
24:     /// </summary>
25:     IFrontEnd m_frontEnd;
26:
27:     /// <summary>
28:     /// Default constructor initializing the APP_HOME and the Front and Back ends
29:     /// </summary>
30:     public AbstractListener()
31:     {
32:       // Fetch the home directory of the listener
33:       Constants.APP_HOME = AppDomain.CurrentDomain.BaseDirectory;
34:
35:       // Create Front/Back-Ends
36:       m_backEnd = GetBackEnd();
37:       m_frontEnd = GetFrontEnd();
38:     }
39:
40:     /// <summary>
41:     /// The body of the listener
42:     ///
43:     /// Initilizes Front and Back End and starts the Front End
44:     /// </summary>
45:     public void Start()
46:     {
47:       m_backEnd.Initialize();
```

```
48:       m_frontEnd.Initialize();
49:       m_frontEnd.Start();
50:     }
51:
52:     /// <summary>
53:     /// Method which terminates the listener. This termines the Front
54:     /// and Back End.
55:     /// </summary>
56:     public void Terminate()
57:     {
58:       m_frontEnd.Terminate();
59:       m_backEnd.Terminate();
60:       m_backEnd = null;
61:       m_frontEnd = null;
62:     }
63:
64:     /// <summary>
65:     /// Method used to pass an event from the Front End to the Back End
66:     /// </summary>
67:     /// <param name="theEvent">The event to pass to the Back End</param>
68:     public void FireEvent(IEvent theEvent)
69:     {
70:       m_backEnd.FireEvent(theEvent);
71:     }
72:
73:     /// <summary>
74:     /// Hook yielding the Back End to use
75:     /// </summary>
76:     /// <returns>The Back End to use</returns>
77:     protected virtual IBackEnd GetBackEnd()
78:     {
79:       return new DefaultBackEnd();
80:     }
81:
82:     /// <summary>
83:     /// Hook yielding the Front End to use
84:     /// </summary>
85:     /// <returns>The Front End to use</returns>
86:     protected abstract IFrontEnd GetFrontEnd();
87:   }
88: }
89:
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics;
using System.ServiceProcess;
using System.Threading;
using Zonith.Listener.Core;
using Zonith.Listener.Core.Util;
using log4net;

[assembly: log4net.Config.DOMConfigurator(ConfigFile="config/log4net.config", Watch=false)]

namespace Zonith.Listener
{
  /// <summary>
  /// Abstract service providing the default implementation of a listener service.
  /// Subclasses should override the hooks GetServiceName() and GetListener()
  /// </summary>
  public abstract class AbstractService : System.ServiceProcess.ServiceBase
  {
    /// <summary>
    /// References to the system log
    /// </summary>
    private ILog m_systemLog = LogManager.GetLogger(Constants.SYSTEM_LOG);

    /// <summary>
    /// Reference to the listener
    /// </summary>
    private AbstractListener listener;

    public AbstractService()
    {
      this.ServiceName = GetServiceName();
    }

    /// <summary>
    /// Set things in motion so your service can do its work.
    /// </summary>
    protected override void OnStart(string[] args)
    {
      listener = GetListener();
      m_systemLog.Info("Service started");
      Thread thread = new Thread(new ThreadStart(listener.Start));
      thread.Start();
    }

    /// <summary>
```

```
48:     /// Stop this service.
49:     /// </summary>
50:     protected override void OnStop()
51:     {
52:       Thread thread = new Thread(new ThreadStart(listener.Terminate));
53:       thread.Start();
54:       thread.Join();
55:       m_systemLog.Info("Service stopped");
56:     }
57:
58:     /// <summary>
59:     /// Hook yielding the service name. This is most likely overridden by the
60:     /// install utility, but it is implemented none the less.
61:     /// </summary>
62:     /// <returns>The name of the service</returns>
63:     protected abstract String GetServiceName();
64:
65:     /// <summary>
66:     /// Hook yielding the Listener to use.
67:     /// </summary>
68:     /// <returns>The Listener to use</returns>
69:     protected abstract AbstractListener GetListener();
70:   }
71: }
72:
```

```
1: //------------------------------------------------------------------------------
2: // <autogenerated>
3: //     This code was generated by a tool.
4: //     Runtime Version: 1.1.4322.573
5: //
6: //     Changes to this file may cause incorrect behavior and will be lost if
7: //     the code is regenerated.
8: // </autogenerated>
9: //------------------------------------------------------------------------------
10:
11: //
12: // This source code was auto-generated by wsdl, Version=1.1.4322.573.
13: //
14: using System.Diagnostics;
15: using System.Xml.Serialization;
16: using System;
17: using System.Web.Services.Protocols;
18: using System.ComponentModel;
19: using System.Web.Services;
20: using Zonith.Listener.Core.Util;
21:
22: /// <remarks/>
23: [System.Diagnostics.DebuggerStepThroughAttribute()]
24: [System.ComponentModel.DesignerCategoryAttribute("code")]
25: [System.Web.Services.WebServiceBindingAttribute(Name="AlarmServiceSoapBinding", Namespace="http://localhost:8080/acs/
      services/AlarmService")]
26: public class AlarmService : System.Web.Services.Protocols.SoapHttpClientProtocol {
27:
28:     /// <remarks/>
29:     public AlarmService() {
30:         this.Url = "http://localhost:8080/acs/services/AlarmService";
31:     }
32:
33:     /// <remarks/>
34:     [System.Web.Services.Protocols.SoapRpcMethodAttribute("", RequestNamespace="http://service.acs.zonith.com",
        ResponseNamespace="http://localhost:8080/acs/services/AlarmService")]
35:     [return: System.Xml.Serialization.SoapElementAttribute("raiseAlarmReturn")]
36:     public int raiseAlarm(string originatingKey, int scfNumber, string listenerId, string timeStamp, string alarmCode,
        string ud, string areagroups) {
37:         object[] results = this.Invoke("raiseAlarm", new object[] {
38:                     originatingKey,
39:                     scfNumber,
40:                     listenerId,
41:                     timeStamp,
42:                     alarmCode,
43:                     ud,
44:                     areagroups});
```

```
45:         return ((int)(results[0]));
46:     }
47:
48:     /// <remarks/>
49:     public System.IAsyncResult BeginraiseAlarm(string originatingKey, int scfNumber, string listenerId, string timeStamp,
          string alarmCode, string ud, string areagroups, System.AsyncCallback callback, object asyncState) {
50:         return this.BeginInvoke("raiseAlarm", new object[] {
51:                     originatingKey,
52:                     scfNumber,
53:                     listenerId,
54:                     timeStamp,
55:                     alarmCode,
56:                     ud,
57:                     areagroups}, callback, asyncState);
58:     }
59:
60:     /// <remarks/>
61:     public int EndraiseAlarm(System.IAsyncResult asyncResult) {
62:         object[] results = this.EndInvoke(asyncResult);
63:         return ((int)(results[0]));
64:     }
65: }
66:
```

```
using System;

namespace Zonith.Listener.Core.Util
{
  /// <summary>
  /// Enumeration used to represent success and error return values
  /// </summary>
  public enum ReturnCode {OK, ERROR};

  /// <summary>
  /// Constants used throughout the abstract listener service framework
  /// </summary>
  public class Constants
  {
    /// <summary>
    /// Location of the property file (relative to APP_HOME)
    /// </summary>
    public static String PROPERTY_FILE = @"config\listener.properties";

    /// <summary>
    /// Listener name property to be read from the property file
    /// </summary>
    public static String PROP_LISTENER_NAME = "listener.name";

    /// <summary>
    /// Company number property to be read from the property file
    /// </summary>
    public static String PROP_COMPANY_NO = "listener.companyNo";

    /// <summary>
    /// ACS end point property to be read from the property file
    /// </summary>
    public static String PROP_ACS_ENDPOINT = "listener.acs.endPoint";

    /// <summary>
    /// Storage file property to be read from the property file
    /// </summary>
    public static String PROP_STORAGE_FILE = "listener.storage";

    /// <summary>
    /// The absolute path to where the service is located
    /// </summary>
    public static String APP_HOME = @"c:\";       // Set during runtime

    /// <summary>
    /// Mask used to standardize the timestamps used throughout the framework
    /// </summary>
```

```
48:     public static String INTERNAL_TIMESTAMP_MASK = "yyyy-MM-dd HH:mm:ss";
49:
50:     /// <summary>
51:     /// The of the logger used for the system log. This name should also be found in
52:     /// the "/config/log4net.config" file (at the bottom usually)
53:     /// </summary>
54:     public static String SYSTEM_LOG = "SystemLog";
55:
56:     /// <summary>
57:     /// The of the logger used for the message log. This name should also be found in
58:     /// the "/config/log4net.config" file (at the bottom usually)
59:     /// </summary>
60:     public static String MESSAGE_LOG = "MessageLog";
61:   }
62: }
```

```
using System;
using Zonith.Listener.Core.Util;
using Zonith.Listener.Core.Interfaces;
using Zonith.Listener.Core.Exceptions;
using log4net;
using System.Threading;

namespace Zonith.Listener.Core.BackEnd
{
  /// <summary>
  ///  This class is the engine for handling incoming event.
  ///  The Class provides two hooks which can be implemented by a
  ///  subclass to change the Listener behaviour.
  ///
  ///  The hooks are:
  ///  <list type="bullet">
  ///  <item><see cref="GetStorageManager()"/>
  ///     - Here a subclass should provide the Storage Strategy for incoming events.
  ///       A default database implementation is provided
  ///       <see cref="Listener.Core.BackEnd.HSQLStorageManager"/>
  ///       until it is able to send these to the ACS.
  ///  </item>
  ///  <item><see cref="GetTransmitProxy()"/>
  ///     - Here a subclass should provide the Proxying Strategy for sending events
  ///       fetched from the Storage Manager.
  ///       A default SOAP proxy implementation is provided
  ///       <see cref="Listener.Core.BackEnd.CoreTransmitProxy"/>
  ///  </item>
  ///  </list>
  /// </summary>
  /// <remarks>
  ///  Author: Nicolai Henriksen
  ///  </remarks>
  public abstract class CoreListener
  {
    /// <summary>
    /// References to the system log
    /// </summary>
    private static ILog systemLog = LogManager.GetLogger(Constants.SYSTEM_LOG);

    /// <summary>
    /// References to the message log
    /// </summary>
    private static ILog messageLog = LogManager.GetLogger(Constants.MESSAGE_LOG);

    /// <summary>
    /// Worker thread retrieving events and delivering them to the ACS
```

```
48:     /// </summary>
49:     private IWorker worker;
50:
51:     /// <summary>
52:     /// Yields whether or not the listener is started
53:     /// </summary>
54:     private bool listenerStart = false;
55:
56:     /// <summary>
57:     /// The listener name (read from property file)
58:     /// </summary>
59:     private String listenerName = "AbstractListenerName";
60:
61:     /// <summary>
62:     /// The company number (read from property file)
63:     /// </summary>
64:     private String companyNo = "1";
65:
66:     /// <summary>
67:     /// The ACS end point (read from property file)
68:     /// </summary>
69:     private String acsEndPoint;
70:
71:     /// <summary>
72:     /// Helper class used to load properties from property file
73:     /// </summary>
74:     private PropertyFileHelper properties;
75:
76:     /// <summary>
77:     /// The singleton instance variable used
78:     /// </summary>
79:     protected static CoreListener instance = null;
80:
81:     private Thread thread = null;
82:
83:     /// <summary>
84:     /// Default constructor
85:     ///
86:     /// Initializes listener details
87:     /// </summary>
88:     public CoreListener()
89:     {
90:       try
91:       {
92:         properties = new PropertyFileHelper(Constants.APP_HOME + Constants.PROPERTY_FILE);
93:         listenerName = properties.GetProperty(Constants.PROP_LISTENER_NAME, listenerName);
94:         companyNo = properties.GetProperty(Constants.PROP_COMPANY_NO, companyNo);
```

```
 95:         acsEndPoint = properties.GetProperty(Constants.PROP_ACS_ENDPOINT);
 96:       }
 97:       catch (EListenerException e)
 98:       {
 99:         systemLog.Error(e.Message, e);
100:       }
101:     }
102:
103:     /// <summary>
104:     /// Starts the listener if not already started
105:     ///
106:     /// This means spawning a worker thread
107:     /// </summary>
108:     public void Start()
109:     {
110:       if (!listenerStart) {
111:         try
112:         {
113:           systemLog.Info(GetListenerName() + " started");
114:           listenerStart = true;
115:           worker = new WorkerThread(GetTransmitProxy(), GetStorageManager());
116:           if (worker.Init())
117:           {
118:             thread = new Thread(new ThreadStart(this.ThreadMain));
119:             thread.Start();
120:           }
121:         }
122:         catch (Exception e)
123:         {
124:           throw new EListenerException(e.Message, e);
125:         }
126:           }
127:     }
128:
129:     /// <summary>
130:     /// Main method of the worker thread (ie. worker.Run() is the main method)
131:     /// </summary>
132:     public void ThreadMain()
133:     {
134:       worker.Run();
135:     }
136:
137:     /// <summary>
138:     /// Terminates the listener (ie. terminates the worker thread)
139:     /// </summary>
140:     public void Terminate()
141:     {
```

```
142:       if (listenerStart) {
143:         worker.Terminate();
144:       }
145:       if (thread != null)
146:       {
147:         thread.Join();
148:       }
149:       systemLog.Info(GetListenerName() + " terminated");
150:     }
151:
152:     /// <summary>
153:     /// Method used to store an event in the Storage Manager to be delivered
154:     /// to the ACS.
155:     /// </summary>
156:     /// <param name="theEvent">The event to store</param>
157:     /// <returns>ReturnCode.OK if it succeeds, ReturnCode.ERROR otherwise</returns>
158:     public ReturnCode FireEvent(IEvent theEvent)
159:     {
160:       try {
161:         Start();
162:       } catch (EListenerException e1) {
163:         systemLog.Error(e1.Message, e1);
164:         return ReturnCode.ERROR;
165:       }
166:
167:       try {
168:         GetStorageManager().Add(theEvent);
169:       } catch (EStorageManagerException e) {
170:         systemLog.Error(e.Message, e);
171:         return ReturnCode.ERROR;
172:       }
173:
174:       WakeUpService();
175:       return ReturnCode.OK;
176:     }
177:
178:     /// <summary>
179:     /// Method used to wakeup a worker thread which is / could be sleeping
180:     /// </summary>
181:     private void WakeUpService()
182:     {
183:       worker.WakeUp();
184:     }
185:
186:     /// <summary>
187:     /// Hook yielding the TransmitProxy
188:     /// </summary>
```

```
189:     /// <returns>The TransmitProxy to be used</returns>
190:     protected abstract ITransmitProxy GetTransmitProxy();
191:
192:     /// <summary>
193:     /// Hook yielding the StorageManager
194:     /// </summary>
195:     /// <returns>The StorageManager to be used</returns>
196:     protected abstract IStorageManager GetStorageManager();
197:
198:     /// <summary>
199:     /// Yields the listener name
200:     /// </summary>
201:     /// <returns>The listener name</returns>
202:     protected String GetListenerName()
203:     {
204:       return listenerName;
205:     }
206:
207:     /// <summary>
208:     /// Yields the company number
209:     /// </summary>
210:     /// <returns>The company number</returns>
211:     protected String GetCompanyNo()
212:     {
213:       return companyNo;
214:     }
215:
216:     /// <summary>
217:     /// Yields the ACS end point
218:     /// </summary>
219:     /// <returns>The ACS end point</returns>
220:     protected String GetAcsEndPoint()
221:     {
222:       return acsEndPoint;
223:     }
224:   }
225: }
```

```
using System;
using log4net;
using Zonith.Listener.Core.Interfaces;
using Zonith.Listener.Core.Util;

namespace Zonith.Listener.Core.BackEnd
{
  /// <summary>
  /// Proxy strategy delivering events to ACS using SOAP/XML
  /// </summary>
  public class CoreTransmitProxy : ITransmitProxy
  {
    /// <summary>
    /// References to the system log
    /// </summary>
    private static ILog systemLog = LogManager.GetLogger(Constants.SYSTEM_LOG);

    /// <summary>
    /// References to the message log
    /// </summary>
    private static ILog messageLog = LogManager.GetLogger(Constants.MESSAGE_LOG);

    /// <summary>
    /// The listener ID
    /// </summary>
    private String listenerId;

    /// <summary>
    /// The company number
    /// </summary>
    private String companyNo;
    private String endPoint;

    /// <summary>
    /// Constructor taking end point, listener id, and company number as parameters
    /// </summary>
    /// <param name="endPoint">The ACS end point</param>
    /// <param name="listenerId">The ID of the listener</param>
    /// <param name="scfNo">The company number</param>
    public CoreTransmitProxy(String endPoint, String listenerId, String companyNo)
    {
      this.endPoint = endPoint;
      this.listenerId = listenerId;
      this.companyNo = companyNo;
      systemLog.Info("SOAP setup: " + this.endPoint + " ;Listener: " + this.listenerId + "; CompanyNo: " + this.companyNo);
    }

```

```
48:     /// <summary>
49:     /// Method which sends the given event to the ACS end point using a class
50:     /// (AlarmService) generated using the wsdl utility
51:     /// </summary>
52:     /// <param name="theEvent">The event to send</param>
53:     /// <returns>ReturnCode.OK if it succeeds, ReturnCode.ERROR otherwise</returns>
54:     private ReturnCode Call(IEvent theEvent)
55:     {
56:       AlarmService alarmService = new AlarmService();
57:       alarmService.Url = endPoint;
58:
59:       int ret = -1;
60:
61:       ret = alarmService.raiseAlarm(theEvent.GetUnitId(),
62:         Convert.ToInt32(companyNo),
63:         listenerId,
64:         theEvent.GetTimeStamp().ToString(Constants.INTERNAL_TIMESTAMP_MASK),
65:         theEvent.GetCode(),
66:         theEvent.GetText(),
67:         theEvent.GetAreaGroups());
68:
69:       if (ret == 0)
70:       {
71:         systemLog.Info("Event transmitted SUCCESSFULLY :" + theEvent);
72:       }
73:       else
74:       {
75:         systemLog.Error("Event transmitted UNSUCCESSFULLY - not recieved on the server (" + ret + ") :" + theEvent);
76:       }
77:       return (ret == 0) ? ReturnCode.OK : ReturnCode.ERROR;
78:     }
79:
80:     #region ITransmitProxy Members
81:
82:     /// <summary>
83:     /// <see cref="Zonith.Listener.Core.Interfaces.ITransmitProxy.Transmit(IEvent theEvent)"/>
84:     /// </summary>
85:     public ReturnCode Transmit(IEvent theEvent)
86:     {
87:       try
88:       {
89:         systemLog.Info("SOAP transmit: " + this.endPoint + " ;Listener: " + this.listenerId + " ;CompanyNo: " + this.
             companyNo);
90:         return Call(theEvent);
91:       }
92:       catch (System.Net.WebException e)
93:       {
```

```
94:           messageLog.Error("Event transmitted UNSUCCESSFULLY - WebException:" + theEvent);
95:           systemLog.Error(e.Message, e);
96:           return ReturnCode.ERROR;
97:         }
98:         catch (System.Web.Services.Protocols.SoapException e)
99:         {
100:          messageLog.Error("Event transmitted UNSUCCESFULLY - SoapException:" + theEvent);
101:          systemLog.Error(e.Message, e);
102:          return ReturnCode.ERROR;
103:        }
104:      }
105:
106:      #endregion
107:    }
108: }
```

```
1: using System;
2: using System.IO;
3: using Zonith.Listener.Core.Interfaces;
4: using Zonith.Listener.Core.Util;
5: using log4net;
6:
7: namespace Zonith.Listener.Core.BackEnd
8: {
9:   /// <summary>
10:   /// Default Back End utilizing the PersistentListener
11:   /// </summary>
12:   public class DefaultBackEnd : IBackEnd
13:   {
14:     #region IBackEnd Members
15:
16:     /// <summary>
17:     /// <see cref="Zonith.Listener.Core.Interfaces.IBackEnd.Initialize()"/>
18:     /// </summary>
19:     public void Initialize()
20:     {
21:       PersistentListener.GetInstance().Start();
22:     }
23:
24:     /// <summary>
25:     /// <see cref="Zonith.Listener.Core.Interfaces.IBackEnd.Terminate()"/>
26:     /// </summary>
27:     public void Terminate()
28:     {
29:       PersistentListener.GetInstance().Terminate();
30:     }
31:
32:     /// <summary>
33:     /// <see cref="Zonith.Listener.Core.Interfaces.IBackEnd.FireEvent(IEvent theEvent)"/>
34:     /// </summary>
35:     public void FireEvent(IEvent theEvent)
36:     {
37:       PersistentListener.GetInstance().FireEvent(theEvent);
38:     }
39:
40:     #endregion
41:   }
42: }
43:
```

```
1: using System;
2: using Zonith.Listener.Core.Interfaces;
3: using Zonith.Listener.Core.Util;
4:
5: namespace Zonith.Listener.Core
6: {
7:   /// <summary>
8:   /// The default event used in the listener framework
9:   /// </summary>
10:  public class DefaultEvent : IEvent
11:  {
12:    /// <summary>
13:    /// The event ID
14:    /// </summary>
15:    private int eventId;
16:
17:    /// <summary>
18:    /// The alarm/event code
19:    /// </summary>
20:    private String code;
21:
22:    /// <summary>
23:    /// The Unit ID
24:    /// </summary>
25:    private String unitId;
26:
27:    /// <summary>
28:    /// The text associated with the alarm/event
29:    /// </summary>
30:    private String text;
31:
32:    /// <summary>
33:    /// The area groups associated with the alarm/event
34:    /// </summary>
35:    private String areagroups;
36:
37:    /// <summary>
38:    /// The timestamp associated with the alarm/event
39:    /// </summary>
40:    private DateTime timestamp;
41:
42:    /// <summary>
43:    /// Default constructor taking all details as parameters
44:    /// </summary>
45:    /// <param name="eventId">The event ID</param>
46:    /// <param name="code">The alarm/event code</param>
47:    /// <param name="unitId">The unit ID</param>
```

```
48:     /// <param name="text">The text associated with the alarm/event</param>
49:     /// <param name="areagroups">The area groups associated with the alarm/event</param>
50:     /// <param name="timestamp">The timestamp associated with the alarm/event</param>
51:     public DefaultEvent(int eventId,
52:               String code,
53:               String unitId,
54:               String text,
55:               String areagroups,
56:               DateTime timestamp)
57:     {
58:       this.eventId = eventId;
59:       this.code = code;
60:       this.unitId = unitId;
61:       this.text = text;
62:       this.areagroups = areagroups;
63:       this.timestamp = timestamp;
64:     }
65:
66:     /// <summary>
67:     /// Overridden method used to return the event as a String
68:     /// </summary>
69:     /// <returns>The event represented as a string</returns>
70:     public override String ToString()
71:     {
72:       return "EVENT(" + GetEventId() + ")[" + GetTimeStamp().ToString(Constants.INTERNAL_TIMESTAMP_MASK) +
73:           "]{" + GetCode() + ":" + GetUnitId() + ":" + GetText() + ":" + GetAreaGroups() + "}";
74:     }
75:
76:     #region IEvent Members
77:
78:     /// <summary>
79:     /// <see cref="Zonith.Listener.Core.Interfaces.IEvent.GetCode()"/>
80:     /// </summary>
81:     public String GetCode()
82:     {
83:       return code;
84:     }
85:
86:     /// <summary>
87:     /// <see cref="Zonith.Listener.Core.Interfaces.IEvent.GetText()"/>
88:     /// </summary>
89:     public String GetText()
90:     {
91:       return text;
92:     }
93:
94:     /// <summary>
```

```
95:     /// <see cref="Zonith.Listener.Core.Interfaces.IEvent.GetAreaGroups()"/>
96:     /// </summary>
97:     public String GetAreaGroups()
98:     {
99:       return areagroups;
100:    }
101:
102:    /// <summary>
103:    /// <see cref="Zonith.Listener.Core.Interfaces.IEvent.GetUnitId()"/>
104:    /// </summary>
105:    public String GetUnitId()
106:    {
107:      return unitId;
108:    }
109:
110:    /// <summary>
111:    /// <see cref="Zonith.Listener.Core.Interfaces.IEvent.GetEventId()"/>
112:    /// </summary>
113:    public int GetEventId()
114:    {
115:      return eventId;
116:    }
117:
118:    /// <summary>
119:    /// <see cref="Zonith.Listener.Core.Interfaces.IEvent.GetTimeStamp()"/>
120:    /// </summary>
121:    public DateTime GetTimeStamp()
122:    {
123:      return timestamp;
124:    }
125:
126:    #endregion
127:  }
128: }
129:
```

```cs
using System;

namespace Zonith.Listener.Core.Exceptions
{
  /// <summary>
  /// Exception used when problems occur in the Storage Manager
  /// </summary>
  public class EStorageManagerException : Exception
  {
    /// <summary>
    /// Constructor taking a message as parameter
    /// </summary>
    /// <param name="message">The message to associate with the exception</param>
    public EStorageManagerException(String message) : base(message) {}

    /// <summary>
    /// Constructor taking a message and an inner exception as parameters
    /// </summary>
    /// <param name="message">The message to associate with the exception</param>
    /// <param name="innerException">The inner exception to associate with the exception</param>
    public EStorageManagerException(String message, Exception innerException) : base(message, innerException) {}
  }

  /// <summary>
  /// Exception used when problems occur in reading properties or starting Worker Threads
  /// </summary>
  public class EListenerException : Exception
  {
    /// <summary>
    /// Constructor taking a message as parameter
    /// </summary>
    /// <param name="message">The message to associate with the exception</param>
    public EListenerException(String message) : base(message) {}

    /// <summary>
    /// Constructor taking a message and an inner exception as parameters
    /// </summary>
    /// <param name="message">The message to associate with the exception</param>
    /// <param name="innerException">The inner exception to associate with the exception</param>
    public EListenerException(String message, Exception innerException) : base(message, innerException) {}
  }
}

```

```
using System;
using System.Text;
using System.Threading;
using System.IO;
using Zonith.Listener.Core.Interfaces;
using Zonith.Listener.Core.Exceptions;
using Zonith.Listener.Core.Util;
using log4net;
using SharpHSQL;

namespace Zonith.Listener.Core.BackEnd
{
  /// <summary>
  /// Implementation of Storage Strategy using HyperSQL. This allows for temporary storage
  /// of events in local files, until the events have been successfully sent to the ACS.
  ///
  /// NOTE: System.Data.SqlClient.SqlCommand is the eqivalence of Java's PreparedStatement
  /// but it is required to take a connection object as parameter and a SharpHSQL Database
  /// object does not implement the correct interface. Therefore the statements containing
  /// variables are implemented as pure string concatenation using a StringBuilder.
  /// </summary>
  public class HSQLStorageManager : IStorageManager
  {
    /// <summary>
    /// References to the system log
    /// </summary>
    private static ILog systemLog = LogManager.GetLogger(Constants.SYSTEM_LOG);

    /// <summary>
    /// References to the message log
    /// </summary>
    private static ILog messageLog = LogManager.GetLogger(Constants.MESSAGE_LOG);

    /// <summary>
    /// The delay between the continous wake up calls sent to the Worker Thread
    /// </summary>
    private const int DELAY = 10000;

    /// <summary>
    /// Indicates whether or not the Storage Manager has been initialized
    /// </summary>
    private bool inited = false;

    /// <summary>
    /// Indicates when to terminate the Storage Manager. Used to kill the thread
    /// that is continously waking up the Worker Thread.
    /// </summary>
```

```
48:     private bool close  = false;
49:
50:     /// <summary>
51:     /// SQL statement used to create the table storing the events in the database
52:     /// </summary>
53:     private const String createTableStatement = "CREATE TABLE events (eventId INTEGER IDENTITY, unitId VARCHAR(256),
          eventCode VARCHAR(256),text VARCHAR(256),areagroups VARCHAR(256),original_time TIMESTAMP,created TIMESTAMP, nextRetry
          TIMESTAMP)";
54:
55:     /// <summary>
56:     /// SQL statement used to insert an event in the database
57:     /// </summary>
58:     private String insertEventStatement = "INSERT INTO events(unitId,eventCode,text,areagroups,created,nextRetry,
          original_time) values (?,?,?,?,?,?,?)";
59:
60:     /// <summary>
61:     /// SQL statement used to delete an event from the database
62:     /// </summary>
63:     private String deleteEventStatement = "DELETE FROM events WHERE eventId = ?";
64:
65:     /// <summary>
66:     /// SQL statement used to retrieve an event from the database
67:     /// </summary>
68:     private String retrieveEventStatement = "SELECT eventId,eventCode,unitId,text,areagroups,original_time FROM events WHERE
          nextRetry <= ? ORDER BY created";
69:
70:     /// <summary>
71:     /// SQL statement used to update an event in the database
72:     /// </summary>
73:     private String updateEventStatement = "UPDATE events SET nextRetry = ? WHERE eventId = ?";
74:
75:     /// <summary>
76:     /// SQL statement used to select the number of waiting events stored in the database
77:     /// </summary>
78:     private String selectNoWaitingEvents = "SELECT count(*) as noEvents FROM events WHERE nextRetry <= ?";
79:
80:     /// <summary>
81:     /// SQL statement used to select the number of events stored in the database
82:     /// </summary>
83:     private const String selectNoEvents = "SELECT count(*) as noEvents FROM events";
84:
85:     /// <summary>
86:     /// The name of the database (actually this is the filename)
87:     /// </summary>
88:     private String dbName;
89:
90:     /// <summary>
```

```
  91:     /// The username used for ADMIN access to the database
  92:     /// </summary>
  93:     private String dbUserName = "sa";
  94:
  95:     /// <summary>
  96:     /// The password used for ADMIN access to the database
  97:     /// </summary>
  98:     private String dbPassword = "";
  99:
 100:     /// <summary>
 101:     /// Reference to the database object
 102:     /// </summary>
 103:     private Database db;
 104:
 105:     /// <summary>
 106:     /// Reference to the Worker Thread retrieving events from the database.
 107:     /// The reference is used to continously wake it up.
 108:     /// </summary>
 109:     private IWorker worker;
 110:
 111:     private Thread thread = null;
 112:
 113:     public void ThreadMain()
 114:     {
 115:       while (!close)
 116:       {
 117:         Thread.Sleep(DELAY);
 118:         worker.WakeUp();
 119:       }
 120:       systemLog.Info("Storage manager terminated");
 121:     }
 122:
 123:     #region IStorageManager Members
 124:
 125:     /// <summary>
 126:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.Init(IWorker worker)"/>
 127:     /// </summary>
 128:     public void Init(IWorker worker)
 129:     {
 130:       lock (this)
 131:       {
 132:         try
 133:         {
 134:           this.worker = worker;
 135:           PropertyFileHelper properties = new PropertyFileHelper(Constants.APP_HOME + Constants.PROPERTY_FILE);
 136:           dbName = Constants.APP_HOME + properties.GetProperty(Constants.PROP_STORAGE_FILE, "hsql_storage");
 137:           systemLog.Info("Trying to create database at: " + dbName);
```

```
138:           if (!Directory.Exists(dbName.Substring(0, dbName.LastIndexOf("\\"))))
139:             Directory.CreateDirectory(dbName.Substring(0, dbName.LastIndexOf("\\")));
140:           db = new Database(dbName);
141:           Channel channel = db.connect(dbUserName, dbPassword);
142:           Result rs = db.execute(createTableStatement, channel);
143:           if (rs.sError != null)
144:             systemLog.Info("The event table already exists ... continue");
145:           systemLog.Info("StorageManager initialized");
146:           inited = true;
147:         }
148:         catch (Exception e)
149:         {
150:           throw new EStorageManagerException("Error initializing StorageManager!", e);
151:         }
152:       }
153:     }
154:
155:     /// <summary>
156:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.Start()"/>
157:     /// </summary>
158:     public void Start()
159:     {
160:       thread = new Thread(new ThreadStart(this.ThreadMain));
161:       thread.Start();
162:     }
163:
164:     /// <summary>
165:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.Terminate()"/>
166:     /// </summary>
167:     public void Terminate()
168:     {
169:       close = true;
170:       if (thread != null)
171:       {
172:         thread.Join();
173:       }
174:     }
175:     /// <summary>
176:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.Add(IEvent theEvent)"/>
177:     /// </summary>
178:     public void Add(IEvent theEvent)
179:     {
180:       while (!inited);
181:       Channel channel = db.connect(dbUserName, dbPassword);
182:       PreparedStatement ps = new PreparedStatement(insertEventStatement);
183:       ps.SetString(1, theEvent.GetUnitId());
184:       ps.SetString(2, theEvent.GetCode());
```

```
185:       ps.SetString(3, theEvent.GetText());
186:       ps.SetString(4, theEvent.GetAreaGroups());
187:       ps.SetDateTime(5, DateTime.Now);
188:       ps.SetDateTime(6, DateTime.Now);
189:       ps.SetDateTime(7, theEvent.GetTimeStamp());
190:
191:       Result rs = db.execute(ps.GetStatement(), channel);
192:       if (rs.sError != null)
193:         throw new EStorageManagerException("HSQLStorageManager.Add() - Could not add event " + theEvent + "[" + rs.sError +
             "]");
194:       else
195:         systemLog.Info("The event " + theEvent + " is added");
196:     }
197:
198:     /// <summary>
199:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.Release(IEvent theEvent)"/>
200:     /// </summary>
201:     public void Release(IEvent theEvent)
202:     {
203:       while (!inited);
204:       Channel channel = db.connect(dbUserName, dbPassword);
205:       PreparedStatement ps = new PreparedStatement(deleteEventStatement);
206:       ps.SetInteger(1, Convert.ToInt32(theEvent.GetEventId()));
207:
208:       Result rs = db.execute(ps.GetStatement(), channel);
209:       if (rs.sError != null)
210:         throw new EStorageManagerException("HSQLStorageManager.Release() - The event " + theEvent + " could not be deleted!
             [" + rs.sError + "]");
211:       else
212:         systemLog.Info("The event " + theEvent + " is deleted");
213:     }
214:
215:     /// <summary>
216:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.RetrieveEvent()"/>
217:     /// </summary>
218:     public IEvent RetrieveEvent()
219:     {
220:       while (!inited);
221:       Channel channel = db.connect(dbUserName, dbPassword);
222:       IEvent theEvent = null;
223:       PreparedStatement ps = new PreparedStatement(retrieveEventStatement);
224:       ps.SetDateTime(1, DateTime.Now);
225:
226:       Result rs = db.execute(ps.GetStatement(), channel);
227:       if (rs.rRoot != null)
228:       {
229:         Record r = rs.rRoot;
```

```
230:         theEvent = new DefaultEvent(Convert.ToInt32(r.data[0]), (String) r.data[1], (String) r.data[2], (String) r.data[3],
               (String) r.data[4], Convert.ToDateTime(r.data[5]));
231:       }
232:       else
233:       {
234:         throw new EStorageManagerException("HSQLStorageManager.RetrieveEvent() - Event could not be selected");
235:       }
236:       return theEvent;
237:     }
238:
239:     /// <summary>
240:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.HasWaitingEvent()"/>
241:     /// </summary>
242:     public bool HasWaitingEvent()
243:     {
244:       while (!inited);
245:       Channel channel = db.connect(dbUserName, dbPassword);
246:       int no = 0;
247:       PreparedStatement ps = new PreparedStatement(selectNoWaitingEvents);
248:       ps.SetDateTime(1, DateTime.Now);
249:
250:       Result rs = db.execute(ps.GetStatement(), channel);
251:
252:       if (rs.sError != null)
253:         throw new EStorageManagerException("HSQLStorageManager.HasWaitingEvent() - Could not select from database [" + rs.
               sError + "]");
254:
255:       if (rs.rRoot != null)
256:       {
257:         Record r = rs.rRoot;
258:         no = Convert.ToInt32(r.data[0]);
259:       }
260:       return no > 0 ? true : false;
261:     }
262:
263:     /// <summary>
264:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.WaitToRetry(IEvent theEvent)"/>
265:     /// </summary>
266:     public void WaitToRetry(IEvent theEvent)
267:     {
268:       while (!inited);
269:       Channel channel = db.connect(dbUserName, dbPassword);
270:       PreparedStatement ps = new PreparedStatement(updateEventStatement);
271:       ps.SetDateTime(1, DateTime.Now);
272:       ps.SetInteger(2, Convert.ToInt32(theEvent.GetEventId()));
273:
274:       Result rs = db.execute(ps.GetStatement(), channel);
```

```
275:       if (rs.sError != null)
276:         throw new EStorageManagerException("HSQLStorageManager.WaitToRetry() - The event " + theEvent + " could not be
               updated! [" + rs.sError + "]");
277:       else
278:         systemLog.Info("The event " + theEvent + " is updated");
279:     }
280:
281:     /// <summary>
282:     /// <see cref="Zonith.Listener.Core.Interfaces.IStorageManager.IsEmpty()"/>
283:     /// </summary>
284:     public bool IsEmpty()
285:     {
286:       while (!inited);
287:       Channel channel = db.connect(dbUserName, dbPassword);
288:       int no = 0;
289:       Result rs = db.execute(selectNoEvents, channel);
290:
291:       if (rs.sError != null)
292:         throw new EStorageManagerException("HSQLStorageManager.IsEmpty() - Could not select from database [" + rs.sError + "
              ]");
293:
294:       if (rs.rRoot != null)
295:       {
296:         Record r = rs.rRoot;
297:         no = Convert.ToInt32(r.data[0]);
298:       }
299:       return no == 0 ? true : false;
300:     }
301:
302:     #endregion
303:   }
304: }
305:
```

```
1: using System;
2: using Zonith.Listener.Core.Util;
3:
4: namespace Zonith.Listener.Core.Interfaces
5: {
6:   /// <summary>
7:   /// Interface for Storage Managers
8:   /// </summary>
9:   public interface IStorageManager
10:   {
11:     /// <summary>
12:     /// Initializes the Storage Manager to use the given worker thread
13:     /// </summary>
14:     /// <param name="worker">The worker thread to associate with the Storage Manager</param>
15:     void Init(IWorker worker);
16:
17:     /// <summary>
18:     /// Starts the StorageManager
19:     /// </summary>
20:     void Start();
21:
22:     /// <summary>
23:     /// Terminates the StorageManager
24:     /// </summary>
25:     void Terminate();
26:
27:     /// <summary>
28:     /// Adds an event to the Storage Manager storage
29:     /// </summary>
30:     /// <param name="theEvent">The event to add to store</param>
31:     void Add(IEvent theEvent);
32:
33:     /// <summary>
34:     /// Releases an event once it has been succesfully transmitted to the end point
35:     /// </summary>
36:     /// <param name="theEvent">The event to release (delete)</param>
37:     void Release(IEvent theEvent);
38:
39:     /// <summary>
40:     /// Fetches the event that has been waiting the longest time
41:     /// </summary>
42:     /// <returns>The event which has waited the longest time</returns>
43:     IEvent RetrieveEvent();
44:
45:     /// <summary>
46:     /// Yields whether or not the Storage Manager has waiting events
47:     /// </summary>
```

```
48:     /// <returns>true if any events are in the store, false otherwise</returns>
49:     bool HasWaitingEvent();
50:
51:     /// <summary>
52:     /// Updates the given event, setting the nextRetry to the curret system time
53:     /// </summary>
54:     /// <param name="theEvent">The event to update</param>
55:     void WaitToRetry(IEvent theEvent);
56:
57:     /// <summary>
58:     /// Yields whether or not the Storage Manager storage is empty
59:     /// </summary>
60:     /// <returns>true if empty, false otherwise</returns>
61:     bool IsEmpty();
62:   }
63:
64:   /// <summary>
65:   /// Interface for worker threads
66:   /// </summary>
67:   public interface IWorker
68:   {
69:     /// <summary>
70:     /// Initialize the Worker Thread
71:     /// </summary>
72:     /// <returns>True if initalization succeeds, False otherwise</returns>
73:     bool Init();
74:
75:     /// <summary>
76:     /// Wake up the thread if it is sleeping
77:     /// </summary>
78:     void WakeUp();
79:
80:     /// <summary>
81:     /// Terminate the thread
82:     /// </summary>
83:     void Terminate();
84:
85:     /// <summary>
86:     /// Main method of the worker thread. Perform thread tasks in here.
87:     /// </summary>
88:     void Run();
89:   }
90:
91:   /// <summary>
92:   /// Interface for Transmit Proxies
93:   /// </summary>
94:   public interface ITransmitProxy
```

```
  95:   {
  96:     /// <summary>
  97:     /// Transmits the given event to the end point associated with the Transmit Proxy
  98:     /// </summary>
  99:     /// <param name="theEvent">The event to transmit</param>
 100:     /// <returns>ReturnCode.OK if it succeeds, ReturnCode.ERROR otherwise</returns>
 101:     ReturnCode Transmit(IEvent theEvent);
 102:   }
 103:
 104:   /// <summary>
 105:   /// Interface for events
 106:   /// </summary>
 107:   public interface IEvent
 108:   {
 109:     /// <summary>
 110:     /// Yields the code (error code) of the event
 111:     /// </summary>
 112:     /// <returns>The code of the event</returns>
 113:     String GetCode();
 114:
 115:     /// <summary>
 116:     /// Yields the text of the event
 117:     /// </summary>
 118:     /// <returns>The text of the event</returns>
 119:     String GetText();
 120:
 121:     /// <summary>
 122:     /// Yields the area groups associated with the event
 123:     /// </summary>
 124:     /// <returns>The area groups associated with the event</returns>
 125:     String GetAreaGroups();
 126:
 127:     /// <summary>
 128:     /// Yields the unit id of the event
 129:     /// </summary>
 130:     /// <returns>The unit id of the event</returns>
 131:     String GetUnitId();
 132:
 133:     /// <summary>
 134:     /// Yields the event id of the event
 135:     /// </summary>
 136:     /// <returns>The event id of the event</returns>
 137:     int GetEventId();
 138:
 139:     /// <summary>
 140:     /// Yields the time stamp of the event
 141:     /// </summary>
```

```
142:     /// <returns>The time stamp of the event</returns>
143:     DateTime GetTimeStamp();
144:   }
145:
146:   public interface IOpcEvent : IEvent
147:   {
148:     /// <summary>
149:     /// Yields the server name of the event
150:     /// </summary>
151:     /// <returns>The server name of the event</returns>
152:     String GetServerName();
153:
154:     /// <summary>
155:     /// Yields the server host of the event
156:     /// </summary>
157:     /// <returns>The server host of the event</returns>
158:     String GetServerHost();
159:
160:     /// <summary>
161:     /// Yields the subscription name of the event
162:     /// </summary>
163:     /// <returns>The subscription name of the event</returns>
164:     String GetSubscriptionName();
165:   }
166:
167:   /// <summary>
168:   /// Interface for Back Ends
169:   /// </summary>
170:   public interface IBackEnd
171:   {
172:     /// <summary>
173:     /// Initializes the Back End
174:     /// </summary>
175:     void Initialize();
176:
177:     /// <summary>
178:     /// Terminates the Back End
179:     /// </summary>
180:     void Terminate();
181:
182:     /// <summary>
183:     /// Fires an event (ie. adds the event to the Storage Manager's storage)
184:     /// </summary>
185:     /// <param name="theEvent">The event to add</param>
186:     void FireEvent(IEvent theEvent);
187:   }
188:
```

```
189:   /// <summary>
190:   /// Interface for Front Ends
191:   /// </summary>
192:   public interface IFrontEnd
193:   {
194:     /// <summary>
195:     /// Initializes the Front End
196:     /// </summary>
197:     void Initialize();
198:
199:     /// <summary>
200:     /// Terminate the Front End
201:     /// </summary>
202:     void Terminate();
203:
204:     /// <summary>
205:     /// Method which listens for incoming events, and sends them to end
206:     /// Storage Manager using <see cref="FireEvent(IEvent theEvent)"/>
207:     /// </summary>
208:     void Start();
209:
210:     /// <summary>
211:     /// Fires an event (ie. sends the event to the Back End)
212:     /// </summary>
213:     /// <param name="theEvent">The event to fire</param>
214:     void FireEvent(IEvent theEvent);
215:   }
216: }
217:
```

```
1: using System;
2: using Zonith.Listener.Core;
3: using Zonith.Listener.Core.Util;
4:
5: namespace Zonith.Listener.Core
6: {
7:   /// <summary>
8:   ///
9:   /// </summary>
10:   public class OpcEvent : DefaultEvent, Interfaces.IOpcEvent
11:   {
12:     /// <summary>
13:     /// The server name
14:     /// </summary>
15:     private String serverName;
16:
17:     /// <summary>
18:     /// The server host
19:     /// </summary>
20:     private String serverHost;
21:
22:     /// <summary>
23:     /// The subscription name
24:     /// </summary>
25:     private String subscriptionName;
26:
27:     /// <summary>
28:     /// Default constructor taking all details as parameters
29:     /// </summary>
30:     /// <param name="serverName">The server name</param>
31:     /// <param name="serverHost">The server host</param>
32:     /// <param name="subscriptionName">The subscription name</param>
33:     /// <param name="eventId">The event ID</param>
34:     /// <param name="code">The alarm/event code</param>
35:     /// <param name="unitId">The unit ID</param>
36:     /// <param name="text">The text associated with the alarm/event</param>
37:     /// <param name="areagroups">The area groups associated with the alarm/event</param>
38:     /// <param name="timestamp">The timestamp associated with the alarm/event</param>
39:     public OpcEvent(String serverName,
40:       String serverHost,
41:       String subscriptionName,
42:       int eventId,
43:       String code,
44:       String unitId,
45:       String text,
46:       String areagroups,
47:       DateTime timestamp) : base(eventId, code, unitId, text, areagroups, timestamp)
```

```
48:     {
49:       this.serverName = serverName;
50:       this.serverHost = serverHost;
51:       this.subscriptionName = subscriptionName;
52:     }
53:     /// <summary>
54:     /// Overridden method used to return the event as a String
55:     /// </summary>
56:     /// <returns>The event represented as a string</returns>
57:     public override String ToString()
58:     {
59:       String str = "EVENT";
60:       if (GetEventId() != -1)
61:         str += "(" + GetEventId() + ")";
62:       str += "[" + GetTimeStamp().ToString(Constants.INTERNAL_TIMESTAMP_MASK) +
63:           "]{" + GetServerName() + ":" + GetServerHost() + ":" + GetSubscriptionName() +
64:           ":" + GetCode() + ":" + GetUnitId() + ":" + GetText() + ":" + GetAreaGroups() + "}";
65:       return str;
66:     }
67:
68:     #region IOpcEvent Members
69:
70:     /// <summary>
71:     /// <see cref="Zonith.Listener.Core.Interfaces.IOpc.GetServerName()"/>
72:     /// </summary>
73:     public String GetServerName()
74:     {
75:       return serverName;
76:     }
77:
78:     /// <summary>
79:     /// <see cref="Zonith.Listener.Core.Interfaces.IOpc.GetServerHost()"/>
80:     /// </summary>
81:     public String GetServerHost()
82:     {
83:       return serverHost;
84:     }
85:
86:     /// <summary>
87:     /// <see cref="Zonith.Listener.Core.Interfaces.IOpc.GetSubscriptionName()"/>
88:     /// </summary>
89:     public String GetSubscriptionName()
90:     {
91:       return subscriptionName;
92:     }
93:
94:     #endregion
```

```
95:    }
96: }
97:
```

```
1: using System;
2: using Zonith.Listener.Core.Interfaces;
3:
4: namespace Zonith.Listener.Core.BackEnd
5: {
6:   /// <summary>
7:   /// Subclass of the CoreListener implementing a persistent storage using
8:   /// an HSQLStorageManager and the default CoreTransmitProxy.
9:   /// </summary>
10:  public class PersistentListener : CoreListener
11:  {
12:    /// <summary>
13:    /// The Storage Manager used in the listener
14:    /// </summary>
15:    private IStorageManager manager = new HSQLStorageManager();
16:
17:    /// <summary>
18:    /// Default constructor calling the constructor of the base class
19:    /// </summary>
20:    private PersistentListener() : base() {}
21:
22:    /// <summary>
23:    /// Static method yielding a reference to the listener (singleton)
24:    /// </summary>
25:    /// <returns>Reference to the listener instance</returns>
26:    public static CoreListener GetInstance()
27:    {
28:      if (instance == null)
29:      {
30:        instance = new PersistentListener();
31:      }
32:      return instance;
33:    }
34:
35:    /// <summary>
36:    /// Overridden method yielding the Storage Manager
37:    /// </summary>
38:    /// <returns></returns>
39:    protected override IStorageManager GetStorageManager()
40:    {
41:      return manager;
42:    }
43:
44:    /// <summary>
45:    /// Overridden method yielding the Transmit Proxy
46:    /// </summary>
47:    /// <returns></returns>
```

```
48:     protected override ITransmitProxy GetTransmitProxy()
49:     {
50:       return new CoreTransmitProxy(GetAcsEndPoint(), GetListenerName(), GetCompanyNo());
51:     }
52:   }
53: }
```

```
using System;
using System.Collections;

namespace Zonith.Listener.Core.Util
{
  /// <summary>
  /// Summary description for PreparedStatement.
  /// </summary>
  public class PreparedStatement
  {
    private string sql;
    private Hashtable hashtable;

    public PreparedStatement(String sql)
    {
      this.sql = sql;
      hashtable = new Hashtable();
    }

    public void SetBoolean(int parameterIndex, bool theValue)
    {
      hashtable[parameterIndex] = Convert.ToString(theValue);
    }

    public void SetInteger(int parameterIndex, int theValue)
    {
      hashtable[parameterIndex] = Convert.ToString(theValue);
    }

    public void SetDouble(int parameterIndex, double theValue)
    {
      // Replace commas with periods
      hashtable[parameterIndex] = Convert.ToString(theValue).Replace(",", ".");
    }

    public void SetString(int parameterIndex, string theValue)
    {
      // Replace single quotes by two single quotes
      hashtable[parameterIndex] = "'" + Convert.ToString(theValue).Replace("'", "''") + "'";
    }

    public void SetDateTime(int parameterIndex, DateTime theValue)
    {
      hashtable[parameterIndex] = "'" + theValue.ToString() + "'";
    }

    public void SetDateTime(int parameterIndex, DateTime theValue, string dateFormat)
```

```
48:     {
49:       hashtable[parameterIndex] = "'" + theValue.ToString(dateFormat) + "'";
50:     }
51:
52:     public string GetStatement()
53:     {
54:       int index = sql.IndexOf("?");
55:       int count = 1;
56:       while (index != -1)
57:       {
58:         string front = sql.Substring(0, index);
59:         string back = sql.Substring(index+1);
60:         string middle = (string) hashtable[count++];
61:         middle = (middle==null) ? "?" : middle;
62:         sql = front + middle + back;
63:         index = sql.IndexOf("?", (front.Length + middle.Length));
64:       }
65:       return sql;
66:     }
67:   }
68: }
69:
```

```
using System;
using System.Collections;
using System.IO;
using Zonith.Listener.Core.Exceptions;

namespace Zonith.Listener.Core.Util
{
  /// <summary>
  /// Simple implementation of a Properties class as known from java.util.Properties
  ///
  /// Note: Implemented using framework specific exceptions
  /// </summary>
  public class Properties
  {
    /// <summary>
    /// Hashtable used to hold all the key/value pairs
    /// </summary>
    Hashtable hash = new Hashtable();

    /// <summary>
    /// Method used to load the properties from the given filename
    /// </summary>
    /// <param name="path">The file from which to load the properties</param>
    public void Load(String path)
    {
      try
      {
        TextReader reader = new StreamReader(path);
        String line = reader.ReadLine();
        int index;
        while (line != null)
        {
          if (!line.StartsWith("#") && !(line.Trim().Length==0))
          {
            index = line.IndexOf("=");
            if (index != -1)
            {
              hash.Add(line.Substring(0,index), line.Substring(index+1));
            }
            else
            {
              throw new EListenerException("Corrupted properties file! '" + path + "'");
            }
          }
          line = reader.ReadLine();
        }
      }
```

```
48:       catch (FileNotFoundException e)
49:       {
50:         throw new EListenerException("Problem reading '" + path + "'", e);
51:       }
52:     }
53:
54:     /// <summary>
55:     /// Method which returns a value corresponding to the given key
56:     /// </summary>
57:     /// <param name="key">The key who's value should be returned</param>
58:     /// <returns>The value associated with the given key, null if it does not exist</returns>
59:     public String GetProperty(String key)
60:     {
61:       return (String) hash[key];
62:     }
63:   }
64: }
65:
```

```
using System;
using log4net;
using Zonith.Listener.Core.Exceptions;

namespace Zonith.Listener.Core.Util
{
  /// <summary>
  /// A wrapper around the Properties class implementing a GetProperty method
  /// which takes a default return value as parameter along with the key.
  /// </summary>
  public class PropertyFileHelper
  {
    /// <summary>
    /// References to the system log
    /// </summary>
    private static ILog systemLog = LogManager.GetLogger(Constants.SYSTEM_LOG);

    /// <summary>
    /// References to the message log
    /// </summary>
    private static ILog messageLog = LogManager.GetLogger(Constants.MESSAGE_LOG);

    /// <summary>
    /// The Properties object used to load the properties
    /// </summary>
    private Properties properties;

    /// <summary>
    /// Constructor taking the path to the property file as parameter
    /// </summary>
    /// <param name="path">The path to the property file</param>
    public PropertyFileHelper(String path)
    {
      properties = new Properties();
      properties.Load(path);
    }

    /// <summary>
    /// Method which returns the value associated with the given key. If the key does
    /// not exist in the property file, the default value is returned
    /// </summary>
    /// <param name="key">The key who's value should be returned</param>
    /// <param name="defaultVal">The default value to use if the key does not exist in the property file</param>
    /// <returns>The value associated with the key if present, defaultVal otherwise</returns>
    public String GetProperty(String key, String defaultVal)
    {
      String res = properties.GetProperty(key);
```

```
48:       if (res == null) {
49:         systemLog.Warn("The property " + key + " is not found using default " + defaultVal);
50:         return defaultVal;
51:       }
52:       return res;
53:     }
54:
55:     /// <summary>
56:     /// Method which returns the value associated with the given key.
57:     /// </summary>
58:     /// <param name="key">The key who's value should be returned</param>
59:     /// <returns>The value associated with the key if present, null otherwise</returns>
60:     public String GetProperty(String key)
61:     {
62:       String res = properties.GetProperty(key);
63:       if (res == null) {
64:         throw new EListenerException("Missing property " + key);
65:       }
66:       return res;
67:     }
68:
69:     /// <summary>
70:     /// Method which returns the Properties object contained in the PropertyFileHelper
71:     /// </summary>
72:     /// <returns>The Properties object contained in the PropertyFileHelper</returns>
73:     public Properties GetProperties()
74:     {
75:         return properties;
76:     }
77:   }
78: }
```

```
using System;
using log4net;
using Zonith.Listener.Core.Exceptions;
using Zonith.Listener.Core.Interfaces;
using Zonith.Listener.Core.Util;
using System.Threading;
using System.Collections;

namespace Zonith.Listener.Core.BackEnd
{
  /// <summary>
  /// Worker Thread class which continously checks for events stored in the
  /// Storage Manager waiting to be transmitted.
  /// </summary>
  public class WorkerThread : IWorker
  {
    /// <summary>
    /// References to the system log
    /// </summary>
    private static ILog systemLog = LogManager.GetLogger(Constants.SYSTEM_LOG);

    /// <summary>
    /// References to the message log
    /// </summary>
    private static ILog messageLog = LogManager.GetLogger(Constants.MESSAGE_LOG);

    /// <summary>
    /// Indicates whether or not the Worker Thread should be closed/terminated.
    ///
    /// Once this is set to true, the Worker Thread transmits what is left in
    /// the database, and then terminates.
    /// </summary>
    private bool close = false;

    /// <summary>
    /// Indicates whether or not the Worker Thread is started
    /// </summary>
    private bool isStarted = false;

    /// <summary>
    /// Reference to the Transmit Proxy
    /// </summary>
    private ITransmitProxy proxy;

    /// <summary>
    /// Reference to the Storage Manager
    /// </summary>
```

```
48:     private IStorageManager manager;
49:
50:     /// <summary>
51:     /// Queue used to perform Wait and Pulse operations using a Monitor
52:     /// </summary>
53:     private Queue queue;
54:
55:     /// <summary>
56:     /// Constructor taking the Transmit Proxy and Storage Manager as parameters
57:     /// </summary>
58:     /// <param name="proxy">The Transmit Proxy to associate with the Worker Thread</param>
59:     /// <param name="manager">The Storage Manager to associate with teh Worker Thread</param>
60:     public WorkerThread(ITransmitProxy proxy, IStorageManager manager)
61:     {
62:       this.proxy = proxy;
63:       this.manager = manager;
64:       queue = new Queue();
65:     }
66:
67:     /// <summary>
68:     /// The body of the Worker Thread. Continously checks for events waiting to be sent,
69:     /// retrieves them if possible, and transmits them to the end point if possible. Otherwise
70:     /// the events are updated in the Storage Manager (for later retrieval).
71:     /// </summary>
72:     public void Run() {
73:       while (!Complete()) {
74:
75:         while (!ContinueJob()) {
76:           WaitForEvents();
77:         }
78:
79:         try {
80:           if (manager.HasWaitingEvent()) {
81:             IEvent theEvent = null;
82:             try {
83:               theEvent = manager.RetrieveEvent();
84:             } catch (EStorageManagerException e) {
85:               systemLog.Error("WorkerThread.Run() - Problem retrieving the event from the store ", e);
86:             }
87:
88:             if (proxy.Transmit(theEvent) == ReturnCode.OK) {
89:               messageLog.Info("The event " + theEvent + " was sent succesfully");
90:               try {
91:                 manager.Release(theEvent);
92:               } catch (EStorageManagerException e1) {
93:                 systemLog.Error("WorkerThread.Run() - Problem relaesing the event from the store ", e1);
94:               }
```

```
 95:             } else {
 96:               manager.WaitToRetry(theEvent);
 97:             }
 98:           }
 99:         } catch (EStorageManagerException e1) {
100:           systemLog.Error("WorkerThread.Run() - Problem checking waiting events ", e1);
101:         }
102:       }
103:       try {
104:         manager.Terminate();
105:       } catch (EStorageManagerException e) {
106:         systemLog.Error("WorkerThread.Run() - Problem shutting down storage manager ", e);
107:       }
108:       systemLog.Info("Worker thread terminated");
109:     }
110:
111:     /// <summary>
112:     /// Method which checks if the Worker Thread is ready to terminate
113:     /// </summary>
114:     /// <returns>True if ready to terminate, false otherwise</returns>
115:     private bool Complete()
116:     {
117:       if (close)
118:       {
119:         try
120:         {
121:
122:           // If the WorkerThread has been told to terminate, then why only stop once all events have ben transmitted?
123:           // The program loops eternally if stopped while events are waiting and transmission to ACS fails
124:
125:           if (manager.IsEmpty())
126:           {
127:             return true;
128:           }
129:           else
130:           {
131:             //return false;
132:             return true;
133:           }
134:         }
135:         catch (EStorageManagerException e)
136:         {
137:           systemLog.Error("WorkerThread.Complete() - Problem checking empty on storage manager ", e);
138:         }
139:       }
140:       return false;
141:     }
```

```
142:
143:     /// <summary>
144:     /// Method which checks if there are any waiting events in the Storage Manager
145:     /// </summary>
146:     /// <returns>True if events are waiting in the Storage Manager, false otherwise</returns>
147:     private bool ContinueJob()
148:     {
149:       try
150:       {
151:         if (close)
152:           return true;
153:         else
154:           return (manager.HasWaitingEvent() ? true : false);
155:       }
156:       catch (EStorageManagerException e)
157:       {
158:         systemLog.Error("WorkerThread.ContinueJob() - Problem checking empty on storage manager ", e);
159:         return false;
160:       }
161:     }
162:
163:     /// <summary>
164:     /// Method which puts the Worker Thread "to sleep" until woken up by a new event being added
165:     /// or the Storage Manager waking it up.
166:     /// </summary>
167:     private void WaitForEvents()
168:     {
169:       lock(queue)
170:       {
171:         Monitor.Wait(queue, Timeout.Infinite);
172:       }
173:     }
174:
175:     #region IWorker Members
176:
177:     /// <summary>
178:     /// <see cref="Zonith.Listener.Core.Interfaces.IWorker.Init()"/>
179:     /// </summary>
180:     public bool Init()
181:     {
182:       systemLog.Info("Worker thread is started");
183:       try
184:       {
185:         manager.Init(this);
186:         manager.Start();
187:       }
188:       catch (EStorageManagerException e)
```

```
189:       {
190:         systemLog.Fatal("WorkerThread.Run() - Problem intializing the storage manager, the listener can't start", e);
191:         return false;
192:       }
193:       return true;
194:     }
195:
196:     /// <summary>
197:     /// <see cref="Zonith.Listener.Core.Interfaces.IWorker.WakeUp()"/>
198:     /// </summary>
199:     public void WakeUp()
200:     {
201:       try
202:       {
203:         lock(queue)
204:         {
205:           if (isStarted)
206:           {
207:             Monitor.Pulse(queue);
208:           }
209:           else
210:           {
211:             isStarted = true;
212:             Monitor.Pulse(queue);
213:           }
214:         }
215:       }
216:       catch (Exception e)
217:       {
218:         systemLog.Fatal("Synchronization error!", e);
219:       }
220:     }
221:
222:     /// <summary>
223:     /// <see cref="Zonith.Listener.Core.Interfaces.IWorker.Terminate()"/>
224:     /// </summary>
225:     public void Terminate()
226:     {
227:       lock(queue)
228:       {
229:         close = true;
230:         Monitor.Pulse(queue);
231:       }
232:     }
233:
234:     #endregion
235:   }
```

```
236: }
237:
```

```
using System;
using Zonith.Configuration.OpcListener;
using Zonith.Listener.Core;
using Zonith.Listener.Core.FrontEnd;
using Zonith.Listener.Core.Interfaces;
using TsOpcNet;
using log4net;
using ParserEngine;

namespace Zonith.Listener.OpcListener
{
  /// <summary>
  /// This is the Event Handler for incoming A&E Events. A&E Events are, by definition, alarms and are therefore
  /// always converted to an IEvent and sent to the back-end.
  /// </summary>
  public class AeEventHandler
  {
    /// <summary>
    /// Reference to the system log
    /// </summary>
    private ILog m_systemLog = LogManager.GetLogger(Core.Util.Constants.SYSTEM_LOG);

    /// <summary>
    /// The server which this Event Handler is connected to
    /// </summary>
    private OpcServerNode m_server;

    /// <summary>
    /// The subscription which this Event Handler is capturing Events for
    /// </summary>
    private OpcAeSubscriptionNode m_subscription;

    /// <summary>
    /// Reference to the abstract front-end enabling the Handler to send IEvents to the back-end.
    /// </summary>
    private AbstractFrontEnd m_frontEnd;

    /// <summary>
    /// Constructor taking the server, subscription and a reference to the abstract front end as parameters.
    /// </summary>
    /// <param name="server">The server which the Event Handler is connected to</param>
    /// <param name="subscription">The subscription which the Event Handler is capturing Events for</param>
    /// <param name="frontEnd">The abstract front end</param>
    public AeEventHandler(OpcServerNode server, OpcAeSubscriptionNode subscription, AbstractFrontEnd frontEnd)
    {
      this.m_server = server;
      this.m_subscription = subscription;
```

```
48:       this.m_frontEnd = frontEnd;
49:     }
50:
51:     /// <summary>
52:     /// Method handling all incoming Events.
53:     /// </summary>
54:     /// <param name="events">Array containing Events captured</param>
55:     /// <param name="refresh"></param>
56:     /// <param name="lastRefresh"></param>
57:     public void HandleEvent(TsCAeEventNotification[] events, bool refresh, bool lastRefresh)
58:     {
59:       IOpcEvent theEvent = null;
60:       foreach (TsCAeEventNotification aeEvent in events)
61:       {
62:         theEvent = new OpcEvent(m_server.OpcServerName,
63:           m_server.OpcServerHost,
64:           m_subscription.SubscriptionName,
65:           -1, // event id is first created when the event is persisted, use -1 to indicate N/A
66:           GetSeverity(aeEvent.Severity),
67:           GetUnitId(),
68:           GetMessage(aeEvent, m_subscription.MessageFormat),
69:           GetAreaGroups(),
70:           aeEvent.Time);
71:         m_frontEnd.FireEvent(theEvent);
72:       }
73:     }
74:
75:     /// <summary>
76:     /// Method which returns the a severity based on the given severity. This is done by searching
77:     /// the severity mappings from the subscription for a matching value/range. If one is found, the
78:     /// mapping is returned, otherwise the original severity is returned.
79:     /// </summary>
80:     /// <param name="severity">The severity to find a mapping for</param>
81:     /// <returns>The string value of the mapping/value of the given severity</returns>
82:     private String GetSeverity(int severity)
83:     {
84:       OpcSeverityMapping[] mappings = m_subscription.SeverityMappings;
85:       foreach (OpcSeverityMapping mapping in mappings)
86:       {
87:         if (mapping.OpcRange.IndexOf("-") != -1)
88:         {
89:           // Range
90:           String[] values = mapping.OpcRange.Split('-');
91:           int low = System.Convert.ToInt32(values[0]);
92:           int high = System.Convert.ToInt32(values[1]);
93:           if (severity >= low && severity <= high)
94:             return mapping.AcsName;
```

```
 95:         }
 96:         else
 97:         {
 98:           // Single value
 99:           if (severity == System.Convert.ToInt32(mapping.OpcRange))
100:             return mapping.AcsName;
101:         }
102:       }
103:       return System.Convert.ToString(severity);
104:     }
105:
106:     /// <summary>
107:     /// Method which replaces the listed elements in the message format with elements fetched
108:     /// from the Event
109:     /// </summary>
110:     /// <param name="aeEvent">The Event to fetch elements from</param>
111:     /// <param name="messageFormat">The raw message format</param>
112:     /// <returns>The message format where $ elements have been replaced</returns>
113:     private String GetMessage(TsCAeEventNotification aeEvent, String messageFormat)
114:     {
115:       String ELEMENT_SOURCE = "$source";
116:       String ELEMENT_SEVERITY = "$severity";
117:       String ELEMENT_MESSAGE = "$message";
118:       String ELEMENT_CONDITION = "$condition";
119:       String ELEMENT_SERVER = "$server";
120:       String ELEMENT_SUBSCRIPTION = "$subscription";
121:       String ELEMENT_FORMULA = "$formula(";
122:
123:       String msg = messageFormat;
124:       msg = msg.Replace(ELEMENT_SOURCE, aeEvent.SourceID);
125:       msg = msg.Replace(ELEMENT_SEVERITY, GetSeverity(aeEvent.Severity));
126:       msg = msg.Replace(ELEMENT_MESSAGE, aeEvent.Message);
127:       msg = msg.Replace(ELEMENT_CONDITION, aeEvent.ConditionName);
128:       msg = msg.Replace(ELEMENT_SERVER, m_server.OpcServerName);
129:       msg = msg.Replace(ELEMENT_SUBSCRIPTION, m_subscription.SubscriptionName);
130:       msg = CalculateFormula(ELEMENT_FORMULA, msg);
131:
132:       return msg;
133:     }
134:
135:     /// <summary>
136:     /// Method which calculates a formula element in the message format.
137:     /// </summary>
138:     /// <param name="pattern">The pattern to search for. This indicates the beginning of the formula</param>
139:     /// <param name="msg">The message format as it looks now. This can take place several times, thus msg is 'under
          construction'</param>
140:     /// <returns>The message format where the formula has been replaced with a value</returns>
```

```
141:     private String CalculateFormula(String pattern, String msg)
142:     {
143:       int index1 = msg.IndexOf(pattern);
144:       if (index1 == -1)
145:       {
146:         return msg;
147:       }
148:       else
149:       {
150:         char[] array = msg.ToCharArray();
151:         int openBrackets = 1;
152:         int closedBrackets = 0;
153:         int index2 = index1 + pattern.Length;
154:         while (openBrackets != closedBrackets)
155:         {
156:           if (index2 > msg.Length - 1)
157:           {
158:             // Message end reached and no closing bracket found -> error!
159:             m_systemLog.Warn("Missing bracket in formula, formula not processed!");
160:             return msg;
161:           }
162:           else if (array[index2] == '(')
163:             openBrackets++;
164:           else if (array[index2] == ')')
165:             closedBrackets++;
166:           index2++;
167:         }
168:
169:         double res;
170:         bool IsError=false;
171:
172:         String formula = msg.Substring(index1 + pattern.Length, (index2 - (index1 + pattern.Length + 1)));
173:         res = Parser.calculate(formula, ref IsError);
174:         if (IsError)
175:         {
176:           m_systemLog.Warn("Formula '" + formula + "' could not be processed!");
177:           return msg;
178:         }
179:         else
180:         {
181:           return msg.Substring(0, index1) + res + msg.Substring(index2);
182:         }
183:       }
184:     }
185:
186:     /// <summary>
187:     /// Method which returns the area groups defined for this subscription. If none are defined, the default area groups
```

```
188:     /// from the server are returned
189:     /// </summary>
190:     /// <returns>The ACS Area Groups</returns>
191:     private String GetAreaGroups()
192:     {
193:       if (m_subscription.AcsAreaGroups != null && m_subscription.AcsAreaGroups != "")
194:         return m_subscription.AcsAreaGroups;
195:       else
196:         return m_server.AcsDefaultAreaGroups;
197:     }
198:
199:     /// <summary>
200:     /// Method which returns the unit id defined for this subscription. If none is defined, the default unit id from the
201:     /// server is returned
202:     /// </summary>
203:     /// <returns>The Unit Id</returns>
204:     private String GetUnitId()
205:     {
206:       if (m_subscription.AcsUnitId != null && m_subscription.AcsUnitId != "")
207:         return m_subscription.AcsUnitId;
208:       else
209:         return m_server.AcsDefaultUnitId;
210:     }
211:   }
212: }
213:
```

```
using System;
using Zonith.Configuration.OpcListener;
using Zonith.Listener.Core;
using Zonith.Listener.Core.FrontEnd;
using Zonith.Listener.Core.Interfaces;
using log4net;
using TsOpcNet;
using ParserEngine;

namespace Zonith.Listener.OpcListener
{
  /// <summary>
  /// This is the Event Handler for incoming DA Events. DA Events merely represent changes in values, and thus all
  /// new values must be checked to see if they fall in the category of Alarms. Only those events that fall in this
  /// category are converted to IEvents and sent to the the back-end.
  /// </summary>
  public class DaEventHandler
  {
    /// <summary>
    /// Reference to the system log
    /// </summary>
    private ILog m_systemLog = LogManager.GetLogger(Core.Util.Constants.SYSTEM_LOG);

    /// <summary>
    /// The server which this Event Handler is connected to
    /// </summary>
    private OpcServerNode m_server;

    /// <summary>
    /// The subscription which this Event Handler is capturing Events for
    /// </summary>
    private OpcDaSubscriptionNode m_subscription;

    /// <summary>
    /// Reference to the abstract front-end enabling the Handler to send IEvents to the back-end.
    /// </summary>
    private AbstractFrontEnd m_frontEnd;

    /// <summary>
    /// Constructor taking the server, subscription and a reference to the abstract front end as parameters.
    /// </summary>
    /// <param name="server">The server which the Event Handler is connected to</param>
    /// <param name="subscription">The subscription which the Event Handler is capturing Events for</param>
    /// <param name="frontEnd">The abstract front end</param>
    public DaEventHandler(OpcServerNode server, OpcDaSubscriptionNode subscription, AbstractFrontEnd frontEnd)
    {
      this.m_server = server;
```

```
48:       this.m_subscription = subscription;
49:       this.m_frontEnd = frontEnd;
50:     }
51:
52:     /// <summary>
53:     /// Method handling all incoming Events
54:     /// </summary>
55:     /// <param name="subscriptionHandle"></param>
56:     /// <param name="requestHandle"></param>
57:     /// <param name="values">Array containing incoming Events</param>
58:     public void HandleEvent(object subscriptionHandle, object requestHandle, TsCDaItemValueResult[] values)
59:     {
60:       IOpcEvent theEvent = null;
61:       foreach (TsCDaItemValueResult daEvent in values)
62:       {
63:         m_systemLog.Info("Value is: " + daEvent.Value);
64:         if (IsAlarmValue(daEvent))
65:         {
66:           theEvent = new OpcEvent(m_server.OpcServerName,
67:             m_server.OpcServerHost,
68:             m_subscription.SubscriptionName,
69:             -1, // event id is first created when the event is persisted, use -1 to indicate N/A
70:             m_subscription.AcsAlarmCode,
71:             GetUnitId(),
72:             GetMessage(daEvent, m_subscription.MessageFormat),
73:             GetAreaGroups(),
74:             daEvent.Timestamp);
75:           m_frontEnd.FireEvent(theEvent);
76:         }
77:       }
78:     }
79:
80:     /// <summary>
81:     /// Method which replaces the listed elements in the message format with elements fetched
82:     /// from the Event
83:     /// </summary>
84:     /// <param name="daEvent">The Event to fetch elements from</param>
85:     /// <param name="messageFormat">The raw message format</param>
86:     /// <returns>The message format where $ elements have been replaced</returns>
87:     private String GetMessage(TsCDaItemValueResult daEvent, String messageFormat)
88:     {
89:       String ELEMENT_VALUE = "$value";
90:       String ELEMENT_ITEM_NAME = "$itemname";
91:       String ELEMENT_ITEM_PATH = "$itempath";
92:       String ELEMENT_RESULT = "$result";
93:       String ELEMENT_SERVER = "$server";
94:       String ELEMENT_SUBSCRIPTION = "$subscription";
```

```
 95:       String ELEMENT_FORMULA = "$formula(";
 96:
 97:       String msg = messageFormat;
 98:       msg = msg.Replace(ELEMENT_VALUE, daEvent.Value.ToString());
 99:       msg = msg.Replace(ELEMENT_ITEM_NAME, daEvent.ItemName);
100:       msg = msg.Replace(ELEMENT_ITEM_PATH, daEvent.ItemPath);
101:       msg = msg.Replace(ELEMENT_RESULT, daEvent.Result.Description());
102:       msg = msg.Replace(ELEMENT_SERVER, m_server.OpcServerName);
103:       msg = msg.Replace(ELEMENT_SUBSCRIPTION, m_subscription.SubscriptionName);
104:       msg = CalculateFormula(ELEMENT_FORMULA, msg);
105:       return msg;
106:     }
107:
108:     /// <summary>
109:     /// Method which calculates a formula element in the message format.
110:     /// </summary>
111:     /// <param name="pattern">The pattern to search for. This indicates the beginning of the formula</param>
112:     /// <param name="msg">The message format as it looks now. This can take place several times, thus msg is 'under
          construction'</param>
113:     /// <returns>The message format where the formula has been replaced with a value</returns>
114:     private String CalculateFormula(String pattern, String msg)
115:     {
116:       int index1 = msg.IndexOf(pattern);
117:       if (index1 == -1)
118:       {
119:         return msg;
120:       }
121:       else
122:       {
123:         char[] array = msg.ToCharArray();
124:         int openBrackets = 1;
125:         int closedBrackets = 0;
126:         int index2 = index1 + pattern.Length;
127:         while (openBrackets != closedBrackets)
128:         {
129:           if (index2 > msg.Length - 1)
130:           {
131:             // Message end reached and no closing bracket found -> error!
132:             m_systemLog.Warn("Missing bracket in formula, formula not processed!");
133:             return msg;
134:           }
135:           else if (array[index2] == '(')
136:             openBrackets++;
137:           else if (array[index2] == ')')
138:             closedBrackets++;
139:           index2++;
140:         }
```

```
141:
142:         double res;
143:         bool IsError=false;
144:
145:         String formula = msg.Substring(index1 + pattern.Length, (index2 - (index1 + pattern.Length + 1)));
146:         res = Parser.calculate(formula, ref IsError);
147:         if (IsError)
148:         {
149:           m_systemLog.Warn("Formula '" + formula + "' could not be processed!");
150:           return msg;
151:         }
152:         else
153:         {
154:           return msg.Substring(0, index1) + res + msg.Substring(index2);
155:         }
156:       }
157:     }
158:
159:     /// <summary>
160:     /// Method which returns the area groups defined for this subscription. If none are defined, the default area groups
161:     /// from the server are returned
162:     /// </summary>
163:     /// <returns>The ACS Area Groups</returns>
164:     private String GetAreaGroups()
165:     {
166:       if (m_subscription.AcsAreaGroups != null && m_subscription.AcsAreaGroups != "")
167:         return m_subscription.AcsAreaGroups;
168:       else
169:         return m_server.AcsDefaultAreaGroups;
170:     }
171:
172:     /// <summary>
173:     /// Method which returns the unit id defined for this subscription. If none is defined, the default unit id from the
174:     /// server is returned
175:     /// </summary>
176:     /// <returns>The Unit Id</returns>
177:     private String GetUnitId()
178:     {
179:       if (m_subscription.AcsUnitId != null && m_subscription.AcsUnitId != "")
180:         return m_subscription.AcsUnitId;
181:       else
182:         return m_server.AcsDefaultUnitId;
183:     }
184:
185:     /// <summary>
186:     /// Method which checks whether or not the given event falls in the alarms category
187:     /// </summary>
```

```
188:     /// <param name="daEvent">The Event to check</param>
189:     /// <returns>True if the Events falls in the alarms category, False otherwise</returns>
190:     private bool IsAlarmValue(TsCDaItemValueResult daEvent)
191:     {
192:       switch (m_subscription.AlarmStateType)
193:       {
194:         case AlarmStateTypes.BOOLEAN:
195:           try
196:           {
197:             bool subValue = System.Convert.ToBoolean(m_subscription.AlarmState);
198:             bool daValue = System.Convert.ToBoolean(daEvent.Value);
199:             return (daValue == subValue);
200:           }
201:           catch (FormatException)
202:           {
203:             return false;
204:           }
205:         case AlarmStateTypes.SINGLE:
206:           return (m_subscription.AlarmState.Equals(daEvent.Value.ToString()));
207:         case AlarmStateTypes.RANGE:
208:           string s1, s2;
209:           int low, high;
210:           try
211:           {
212:             s1 = m_subscription.AlarmState.Substring(0, m_subscription.AlarmState.IndexOf(" to "));
213:             s2 = m_subscription.AlarmState.Substring(m_subscription.AlarmState.IndexOf(" to ")+4);
214:             low = System.Convert.ToInt32(s1);
215:             high = System.Convert.ToInt32(s2);
216:             int val = System.Convert.ToInt32(daEvent.Value);
217:             return (val >= low && val <= high);
218:           }
219:           catch (FormatException)
220:           {
221:             s1 = m_subscription.AlarmState.Substring(0, m_subscription.AlarmState.IndexOf(" to "));
222:             s2 = m_subscription.AlarmState.Substring(m_subscription.AlarmState.IndexOf(" to ")+4);
223:             return (daEvent.Value.ToString().CompareTo(s1) >= 0 && daEvent.Value.ToString().CompareTo(s2) <= 0);
224:           }
225:         default:
226:           m_systemLog.Warn("Wrong Alarm State Type specified! [" + m_subscription.AlarmStateType.ToString() + "]");
227:           return false;
228:       }
229:     }
230:   }
231: }
232:
```

```
using System;
using System.Threading;
using System.Windows.Forms;
using System.Collections;
using Zonith.Listener.Core;
using Zonith.Listener.Core.Interfaces;
using Zonith.Listener.Core.Util;
using Zonith.Listener.Core.FrontEnd;
using Zonith.Configuration.OpcListener;
using log4net;
using TsOpcNet;

namespace Zonith.Listener.OpcListener
{
  /// <summary>
  /// The Implementation of an OPC front-end capturing events from OPC Servers. This is done by reading
  /// a configuration file (XML), and establishing the defined connections and subscriptions. An EventHandler
  /// is created for each subscription being handled which processes the Event and sends the appropriate ones
  /// to the back-end
  /// </summary>
  public class OpcFrontEnd : AbstractFrontEnd
  {
    /// <summary>
    /// Reference to the system log
    /// </summary>
    private ILog m_systemLog = LogManager.GetLogger(Core.Util.Constants.SYSTEM_LOG);

    /// <summary>
    /// Path to the configuration file
    /// </summary>
    private String CONFIG_FILE;

    /// <summary>
    /// List of servers which where a connection has been established
    /// </summary>
    private IList m_servers;

    /// <summary>
    /// Number used to create unique group numbers when establishing multiple connections on the same server
    /// </summary>
    private int m_alarmGroupNumber;

    /// <summary>
    /// Constructor calling the constructor of the base class
    /// </summary>
    /// <param name="listener">Reference the the OpcListener which created the front-end</param>
    public OpcFrontEnd(AbstractListener listener) : base(listener)
```

```
48:     {
49:       CONFIG_FILE = Constants.APP_HOME + "/config/listener.cfg";
50:       m_servers = new ArrayList();
51:     }
52:
53:     /// <summary>
54:     /// Initialization hook
55:     /// </summary>
56:     public override void Initialize()
57:     {
58:       base.Initialize ();
59:       m_alarmGroupNumber = 0;
60:     }
61:
62:     /// <summary>
63:     /// Start hook
64:     /// </summary>
65:     public override void Start()
66:     {
67:       try
68:       {
69:         OpcServerNode[] servers = XmlConfiguration.ReadConfigurationFromFile(CONFIG_FILE);
70:         ArrayList comServers = new ArrayList();
71:         ArrayList xmlServers = new ArrayList();
72:
73:         foreach (OpcServerNode server in servers)
74:         {
75:           if (server.UseCom)
76:             comServers.Add(server);
77:           else
78:             xmlServers.Add(server);
79:         }
80:         ConnectToComServers(comServers);
81:         ConnectToXmlServers(xmlServers);
82:
83:       }
84:       catch (System.InvalidOperationException)
85:       {
86:         m_systemLog.Error("Error connecting through Technosoftware API - Make sure license file is present and valid");
87:       }
88:       catch (Exception e)
89:       {
90:         m_systemLog.Error("Error while initializing server listeners", e);
91:       }
92:     }
93:
94:     /// <summary>
```

```
  95:     /// Terminate hook
  96:     /// </summary>
  97:     public override void Terminate()
  98:     {
  99:       object[] serverArray = new object[m_servers.Count];
 100:       m_servers.CopyTo(serverArray, 0);
 101:
 102:       for (int i=0; i<serverArray.Length; i++)
 103:       {
 104:         object server = serverArray[i];
 105:         if (server is TsCAeServer)
 106:           ((TsCAeServer) server).Disconnect();
 107:         else if (server is TsCDaServer)
 108:           ((TsCDaServer) server).Disconnect();
 109:       }
 110:     }
 111:
 112:     /// <summary>
 113:     /// Method establishing connections to the given list of COM servers
 114:     /// </summary>
 115:     /// <param name="comServers">List of COM servers to connect to</param>
 116:     private void ConnectToComServers(ArrayList comServers)
 117:     {
 118:       foreach (OpcServerNode server in comServers)
 119:       {
 120:         foreach(TreeNode node in server.Nodes)
 121:         {
 122:           OpcSubscriptionNode subscription = (OpcSubscriptionNode) node;
 123:
 124:           if (subscription.Enabled)
 125:           {
 126:             if (subscription is OpcAeSubscriptionNode)
 127:               CreateSubscriptionToAeServer(server, (OpcAeSubscriptionNode) subscription);
 128:             if (subscription is OpcDaSubscriptionNode)
 129:               CreateSubscriptionToDaServer(server, (OpcDaSubscriptionNode) subscription);
 130:           }
 131:         }
 132:       }
 133:     }
 134:
 135:     /// <summary>
 136:     /// Method which creates the given A&E subscription to the given server
 137:     /// </summary>
 138:     /// <param name="server">Server on which to create a subscription</param>
 139:     /// <param name="subscription">The subscription to create</param>
 140:     private void CreateSubscriptionToAeServer(OpcServerNode server, OpcAeSubscriptionNode subscription)
 141:     {
```

```
142:       bool createNewServer = false;
143:
144:       TsOpcComputerInfo host = new TsOpcComputerInfo(server.OpcServerHost);
145:
146:       TsCAeServer comServer;
147:       object o = GetServerIfAlreadyCreated(server.OpcServerName);
148:       if (o == null)
149:       {
150:         comServer = new TsCAeServer();
151:         createNewServer = true;
152:         comServer.Connect(server.OpcServerName, host);
153:       }
154:       else
155:       {
156:         comServer = (TsCAeServer) o;
157:       }
158:
159:       AeEventHandler handler = new AeEventHandler(server, subscription, this);
160:
161:       TsOpcNet.TsCAeSubscriptionState state = new TsOpcNet.TsCAeSubscriptionState();
162:       state.Active = true;
163:       state.BufferTime = 0;
164:       state.MaxSize = 0;
165:       state.ClientHandle = 100;
166:       state.Name = subscription.SubscriptionName;
167:
168:       TsOpcNet.TsCAeSubscription comSubscription;
169:       comSubscription = (TsOpcNet.TsCAeSubscription) comServer.CreateSubscription(state);
170:       comSubscription.EventChanged += new TsCAeEventChangedHandler(handler.HandleEvent);
171:
172:       // Apply filter if enabled
173:       if (subscription.FilterEnabled)
174:       {
175:         // Fetch the filter from the subscription
176:         TsCAeSubscriptionFilters filters = comSubscription.GetFilters();
177:
178:         // Add sources
179:         try
180:         {
181:           AddFilterElements(filters.Sources, subscription.Sources, 0);
182:           comSubscription.SetFilters(filters);
183:         }
184:         catch (TsOpcNet.TsOpcResultException)
185:         {
186:           m_systemLog.Error("Invalid source name specified! - Ignoring subscription '" + subscription.SubscriptionName + "'!
                ");
187:           comServer.Disconnect();
```

```
188:           return;
189:         }
190:
191:         // Add areas
192:         try
193:         {
194:           AddFilterElements(filters.Areas, subscription.Areas, 0);
195:           comSubscription.SetFilters(filters);
196:         }
197:         catch (TsOpcNet.TsOpcResultException)
198:         {
199:           m_systemLog.Error("Invalid area name specified! - Ignoring subscription '" + subscription.SubscriptionName + "'!")
                 ;
200:           comServer.Disconnect();
201:           return;
202:         }
203:
204:         // Add event type
205:         switch(subscription.EventType.ToLower())
206:         {
207:           case "all":
208:             //filters.EventTypes = (int) TsCAeEventType.All;  // Yields 65535 instead of 7 !!?
209:             filters.EventTypes = 7;
210:             break;
211:           case "condition":
212:             filters.EventTypes = (int) TsCAeEventType.Condition;
213:             break;
214:           case "simple":
215:             filters.EventTypes = (int) TsCAeEventType.Simple;
216:             break;
217:           case "tracking":
218:             filters.EventTypes = (int) TsCAeEventType.Tracking;
219:             break;
220:           default:
221:             m_systemLog.Error("Invalid event type specified! - Ignoring subscription '" + subscription.SubscriptionName + "'
                   !");
222:             comServer.Disconnect();
223:             return;
224:         }
225:         comSubscription.SetFilters(filters);
226:
227:         // Add event categories
228:         try
229:         {
230:           AddFilterElements(filters.Categories, subscription.CategoryIds, 1);
231:           comSubscription.SetFilters(filters);
232:         }
```

```
233:          catch (TsOpcNet.TsOpcResultException)
234:          {
235:            m_systemLog.Error("Invalid category id specified! - Ignoring subscription '" + subscription.SubscriptionName + "'!
                  ");
236:            comServer.Disconnect();
237:            return;
238:          }
239:
240:          // Add lowest severity
241:          try
242:          {
243:            int i = System.Convert.ToInt32(subscription.LowestSeverity);
244:            if (i > 0)
245:            {
246:              filters.LowSeverity = i;
247:              comSubscription.SetFilters(filters);
248:            }
249:          }
250:          catch (TsOpcNet.TsOpcResultException)
251:          {
252:            m_systemLog.Error("Invalid lowest severity specified! - Ignoring subscription '" + subscription.SubscriptionName +
                  "'!");
253:            comServer.Disconnect();
254:            return;
255:          }
256:          catch (System.FormatException)
257:          {
258:            m_systemLog.Error("Invalid lowest severity specified! - Ignoring subscription '" + subscription.SubscriptionName +
                  "'!");
259:            comServer.Disconnect();
260:            return;
261:          }
262:
263:          // Add highest severity
264:          try
265:          {
266:            int i = System.Convert.ToInt32(subscription.HighestSeverity);
267:            if (i < 1000)
268:            {
269:              filters.HighSeverity = i;
270:              comSubscription.SetFilters(filters);
271:            }
272:          }
273:          catch (TsOpcNet.TsOpcResultException)
274:          {
275:            m_systemLog.Error("Invalid highest severity specified! - Ignoring subscription '" + subscription.SubscriptionName
                  + "'!");
```

```
276:           comServer.Disconnect();
277:           return;
278:         }
279:         catch (System.FormatException)
280:         {
281:           m_systemLog.Error("Invalid highest severity specified! - Ignoring subscription '" + subscription.SubscriptionName
                 + "'!");
282:           comServer.Disconnect();
283:           return;
284:         }
285:       }
286:       if (createNewServer)
287:         m_servers.Add(comServer);
288:     }
289:
290:     /// <summary>
291:     /// Method which creates the given DA subscription to the given server
292:     /// </summary>
293:     /// <param name="server">Server on which to create a subscription</param>
294:     /// <param name="subscription">The subscription to create</param>
295:     private void CreateSubscriptionToDaServer(OpcServerNode server, OpcDaSubscriptionNode subscription)
296:     {
297:       bool createNewServer = false;
298:
299:       TsOpcComputerInfo host = new TsOpcComputerInfo(server.OpcServerHost);
300:
301:       TsCDaServer comServer;
302:       object o = GetServerIfAlreadyCreated(server.OpcServerName);
303:       if (o == null)
304:       {
305:         comServer = new TsCDaServer();
306:         createNewServer = true;
307:         comServer.Connect(server.OpcServerName, host);
308:       }
309:       else
310:       {
311:         comServer = (TsCDaServer) o;
312:       }
313:
314:       DaEventHandler handler = new DaEventHandler(server, subscription, this);
315:
316:       TsCDaSubscription group;
317:       TsCDaSubscriptionState groupState = new TsCDaSubscriptionState();
318:       groupState.Name = "AlarmGroup" + m_alarmGroupNumber++;                          // Group Name
319:       group = (TsCDaSubscription) comServer.CreateSubscription(groupState);
320:
321:       TsCDaItem[] items = new TsCDaItem[1];
```

```
322:        TsCDaItemResult[] itemResults;
323:        items[0] = new TsCDaItem();
324:        items[0].ItemName = subscription.Item;
325:        items[0].ClientHandle = 100;
326:
327:        itemResults = group.AddItems(items);
328:
329:        for (int i = 0; i < itemResults.GetLength(0); i++)
330:        {
331:          if ( itemResults[i].Result.IsError() )
332:          {
333:            m_systemLog.Warn("   Item " + itemResults[i].ItemName + "could not be added to the group");
334:          }
335:        }
336:
337:        group.DataChanged += new TsCDaDataChangedHandler(handler.HandleEvent);
338:
339:        if (createNewServer)
340:          m_servers.Add(comServer);
341:      }
342:
343:      private void ConnectToXmlServers(ArrayList comServers)
344:      {
345:        // Not implemented
346:        m_systemLog.Warn("This version of the OPC Listener does not support XML connections! Please contact Zonith A/S for a
            newer version.");
347:      }
348:
349:      /// <summary>
350:      /// Method which checks if a connection to the given server has already been established. If so, the
351:      /// connection is returned, otherwise null is returned to indicate that a new connection needs to be
352:      /// established
353:      /// </summary>
354:      /// <param name="serverName">The server name to look for</param>
355:      /// <returns>An existing connection or null if it does not exist</returns>
356:      private object GetServerIfAlreadyCreated(string serverName)
357:      {
358:        foreach (object o in m_servers)
359:        {
360:          if (o is TsCAeServer)
361:          {
362:            TsCAeServer server = (TsCAeServer) o;
363:            if (server.Name.Equals(serverName))
364:              return server;
365:          }
366:          else if (o is TsCDaServer)
367:          {
```

```
368:           TsCDaServer server = (TsCDaServer) o;
369:           if (server.Name.Equals(serverName))
370:             return server;
371:         }
372:       }
373:       return null;
374:     }
375:
376:     /// <summary>
377:     /// Method which tokenizes a comma separated string, and adds the tokens to the given
378:     /// collection according to the given type. If the type is 0, tokens are treated as strings, otherwise
379:     /// the tokens are treated as numbers
380:     /// </summary>
381:     /// <param name="collection">The collection where the tokens are added</param>
382:     /// <param name="str">The comma separated string</param>
383:     /// <param name="type">Type indication of the tokens</param>
384:     private void AddFilterElements(IList collection, String str, int type)
385:     {
386:       String[] array = str.Split(',');
387:       foreach (String s in array)
388:       {
389:         if (s != "")
390:         {
391:           if (type==0)
392:           {
393:             collection.Add(s.Trim());
394:           }
395:           else
396:           {
397:             try
398:             {
399:               int i = System.Convert.ToInt32(s.Trim());
400:               collection.Add(i);
401:             }
402:             catch (Exception)
403:             {
404:               // Throw an exception to indicate an error occurred while converting the id
405:               throw new TsOpcNet.TsOpcResultException(new TsOpcNet.TsOpcResult(-1));
406:             }
407:           }
408:         }
409:       }
410:     }
411:   }
412: }
413:
```

```
1: using System;
2: using Zonith.Listener.Core;
3: using Zonith.Listener.Core.Interfaces;
4:
5: namespace Zonith.Listener.OpcListener
6: {
7:   /// <summary>
8:   /// Implementation of an OPC Listener
9:   /// </summary>
10:  public class OpcListener : AbstractListener
11:  {
12:    /// <summary>
13:    /// GetFrontEnd hook
14:    /// </summary>
15:    /// <returns>Reference to the front-end</returns>
16:    protected override IFrontEnd GetFrontEnd()
17:    {
18:      return new OpcFrontEnd(this);
19:    }
20:  }
21: }
22:
```

```
1: using System;
2: using System.Collections;
3: using System.ComponentModel;
4: using System.Data;
5: using System.Diagnostics;
6: using System.ServiceProcess;
7: using Zonith.Listener;
8: using Zonith.Listener.Core;
9:
10: namespace Zonith.Listener.OpcListener
11: {
12:   /// <summary>
13:   /// Implementation of an OPC Listener Service. This will install as a Windows Service.
14:   /// </summary>
15:   public class OpcService : AbstractService
16:   {
17:     /// <summary>
18:     /// The name of the Windows Service (installed as)
19:     /// </summary>
20:     public static String serviceName = "ACS OPC Listener Service";
21:
22:     /// <summary>
23:     /// The display name of the Windows Service (what is displayed in the list of Services)
24:     /// </summary>
25:     public static String displayName = "ACS OPC Listener";
26:
27:     // The main entry point for the process
28:     static void Main()
29:     {
30:       System.ServiceProcess.ServiceBase[] ServicesToRun;
31:
32:       // More than one user Service may run within the same process. To add
33:       // another service to this process, change the following line to
34:       // create a second service object. For example,
35:       //
36:       //   ServicesToRun = new System.ServiceProcess.ServiceBase[] {new Service1(), new MySecondUserService()};
37:       //
38:       ServicesToRun = new System.ServiceProcess.ServiceBase[] { new OpcService() };
39:
40:       System.ServiceProcess.ServiceBase.Run(ServicesToRun);
41:     }
42:
43:     /// <summary>
44:     /// GetListener hook
45:     /// </summary>
46:     /// <returns>The OpcListener</returns>
47:     protected override AbstractListener GetListener()
```

```
48:     {
49:       return new OpcListener();
50:     }
51:
52:     /// <summary>
53:     /// GetServiceName hook
54:     /// </summary>
55:     /// <returns>The service name of the Opc Listener</returns>
56:     protected override string GetServiceName()
57:     {
58:       return serviceName;
59:     }
60:   }
61: }
62:
```

```
1: using System;
2: using System.Collections;
3: using System.ComponentModel;
4: using System.Configuration.Install;
5:
6: namespace Zonith.Listener.OpcListener
7: {
8:   /// <summary>
9:   /// Summary description for ProjectInstaller.
10:  /// </summary>
11:  [RunInstaller(true)]
12:  public class ProjectInstaller : System.Configuration.Install.Installer
13:  {
14:    private System.ServiceProcess.ServiceProcessInstaller serviceProcessInstaller1;
15:    private System.ServiceProcess.ServiceInstaller serviceInstaller1;
16:    /// <summary>
17:    /// Required designer variable.
18:    /// </summary>
19:    private System.ComponentModel.Container components = null;
20:
21:    public ProjectInstaller()
22:    {
23:      // This call is required by the Designer.
24:      InitializeComponent();
25:
26:      // TODO: Add any initialization after the InitializeComponent call
27:    }
28:
29:    /// <summary>
30:    /// Clean up any resources being used.
31:    /// </summary>
32:    protected override void Dispose( bool disposing )
33:    {
34:      if( disposing )
35:      {
36:        if(components != null)
37:        {
38:          components.Dispose();
39:        }
40:      }
41:      base.Dispose( disposing );
42:    }
43:
44:
45:    #region Component Designer generated code
46:    /// <summary>
47:    /// Required method for Designer support - do not modify
```

```
48:     /// the contents of this method with the code editor.
49:     /// </summary>
50:     private void InitializeComponent()
51:     {
52:       this.serviceProcessInstaller1 = new System.ServiceProcess.ServiceProcessInstaller();
53:       this.serviceInstaller1 = new System.ServiceProcess.ServiceInstaller();
54:       //
55:       // serviceProcessInstaller1
56:       //
57:       this.serviceProcessInstaller1.Account = System.ServiceProcess.ServiceAccount.User;
58:       //
59:       // serviceInstaller1
60:       //
61:       this.serviceInstaller1.DisplayName = OpcService.displayName;
62:       this.serviceInstaller1.ServiceName = OpcService.serviceName;
63:       this.serviceInstaller1.StartType = System.ServiceProcess.ServiceStartMode.Automatic;
64:       //
65:       // ProjectInstaller
66:       //
67:       this.Installers.AddRange(new System.Configuration.Install.Installer[] {
68:                                               this.serviceProcessInstaller1,
69:                                               this.serviceInstaller1});
70:
71:     }
72:     #endregion
73:   }
74: }
75:
```

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace Zonith.Listener.OpcListener.Configuration
{
  /// <summary>
  /// Summary description for AboutBox.
  /// </summary>
  public class AboutBox : System.Windows.Forms.Form
  {
    public static string ProgramName = "";
    public static string Title = "";

    private System.Windows.Forms.PictureBox pictureBox1;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public AboutBox()
    {
      //
      // Required for Windows Form Designer support
      //
      InitializeComponent();

      //
      // TODO: Add any constructor code after InitializeComponent call
      //
      this.Text = Title;
      label2.Text = ProgramName;
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
```

```
48:         if(components != null)
49:         {
50:           components.Dispose();
51:         }
52:       }
53:       base.Dispose( disposing );
54:     }
55:
56:     #region Windows Form Designer generated code
57:     /// <summary>
58:     /// Required method for Designer support - do not modify
59:     /// the contents of this method with the code editor.
60:     /// </summary>
61:     private void InitializeComponent()
62:     {
63:       System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(AboutBox));
64:       this.pictureBox1 = new System.Windows.Forms.PictureBox();
65:       this.label1 = new System.Windows.Forms.Label();
66:       this.button1 = new System.Windows.Forms.Button();
67:       this.label2 = new System.Windows.Forms.Label();
68:       this.label3 = new System.Windows.Forms.Label();
69:       this.SuspendLayout();
70:       //
71:       // pictureBox1
72:       //
73:       this.pictureBox1.Anchor = System.Windows.Forms.AnchorStyles.None;
74:       this.pictureBox1.Image = ((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
75:       this.pictureBox1.Location = new System.Drawing.Point(8, 8);
76:       this.pictureBox1.Name = "pictureBox1";
77:       this.pictureBox1.Size = new System.Drawing.Size(128, 24);
78:       this.pictureBox1.TabIndex = 14;
79:       this.pictureBox1.TabStop = false;
80:       //
81:       // label1
82:       //
83:       this.label1.Location = new System.Drawing.Point(240, 8);
84:       this.label1.Name = "label1";
85:       this.label1.Size = new System.Drawing.Size(104, 16);
86:       this.label1.TabIndex = 15;
87:       this.label1.Text = "© Zonith A/S 2005  ";
88:       //
89:       // button1
90:       //
91:       this.button1.FlatStyle = System.Windows.Forms.FlatStyle.System;
92:       this.button1.Location = new System.Drawing.Point(136, 88);
93:       this.button1.Name = "button1";
94:       this.button1.Size = new System.Drawing.Size(64, 24);
```

```
95:       this.button1.TabIndex = 16;
96:       this.button1.Text = "OK";
97:       this.button1.Click += new System.EventHandler(this.button1_Click);
98:       //
99:       // label2
100:      //
101:      this.label2.Location = new System.Drawing.Point(84, 40);
102:      this.label2.Name = "label2";
103:      this.label2.Size = new System.Drawing.Size(168, 16);
104:      this.label2.TabIndex = 17;
105:      this.label2.Text = "OPC Listener Configuration";
106:      this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
107:      //
108:      // label3
109:      //
110:      this.label3.Location = new System.Drawing.Point(130, 56);
111:      this.label3.Name = "label3";
112:      this.label3.Size = new System.Drawing.Size(76, 16);
113:      this.label3.TabIndex = 18;
114:      this.label3.Text = "Version 1.0.1";
115:      this.label3.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
116:      //
117:      // AboutBox
118:      //
119:      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
120:      this.ClientSize = new System.Drawing.Size(336, 118);
121:      this.ControlBox = false;
122:      this.Controls.Add(this.label3);
123:      this.Controls.Add(this.label2);
124:      this.Controls.Add(this.button1);
125:      this.Controls.Add(this.label1);
126:      this.Controls.Add(this.pictureBox1);
127:      this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
128:      this.MaximizeBox = false;
129:      this.MinimizeBox = false;
130:      this.Name = "AboutBox";
131:      this.SizeGripStyle = System.Windows.Forms.SizeGripStyle.Hide;
132:      this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
133:      this.Text = "About OPC Listener Configuration...";
134:      this.ResumeLayout(false);
135:
136:    }
137:    #endregion
138:
139:    private void button1_Click(object sender, System.EventArgs e)
140:    {
141:      this.Close();
```

```
142:      }
143:    }
144: }
145:
```

```
using System;
using System.Collections;
using System.Text.RegularExpressions;
using Zonith.Configuration.OpcListener;

namespace Zonith.Listener.OpcListener.Configuration
{
  /// <summary>
  /// Summary description for Controller.
  /// </summary>
  public class Controller
  {
    /// <summary>
    /// Reference to the View
    /// </summary>
    private View view;

    /// <summary>
    /// Reference to the Model
    /// </summary>
    private Model model;

    #region Default Values

    /// <summary>
    /// Default values used when creating servers/subscriptions
    ///
    /// Some of these are modified once a language is loaded.
    /// </summary>
    public static string DEFAULT_SERVER_SERVER_NAME = "New Server";
    private const string DEFAULT_SERVER_SERVER_HOST = "";
    private const string DEFAULT_SERVER_DEFAULT_UNIT_ID = "";
    private const string DEFAULT_SERVER_DEFAULT_AREA_GROUPS = "";
    private const bool DEFAULT_SERVER_USE_COM = true;
    private string[] DEFAULT_SERVER_ACS_ALARM_CODES = new string[1] { "NONE" };
    private const bool DEFAULT_SUBSCRIPTION_ENABLED = true;
    public static string DEFAULT_SUBSCRIPTION_SUBSCRIPTION_NAME = "New Subscription";
    private const string DEFAULT_SUBSCRIPTION_UNIT_ID = "";
    private const string DEFAULT_SUBSCRIPTION_AREA_GROUPS = "";
    public static string DEFAULT_AE_SUBSCRIPTION_MESSAGE_FORMAT = "$message";
    private const bool DEFAULT_AE_SUBSCRIPTION_FILTER_ENABLED = false;
    private const string DEFAULT_AE_SUBSCRIPTION_FILTER_SOURCES = "";
    private const string DEFAULT_AE_SUBSCRIPTION_FILTER_AREAS = "";
    private const string DEFAULT_AE_SUBSCRIPTION_FILTER_EVENT_TYPE = "All";
    private const string DEFAULT_AE_SUBSCRIPTION_FILTER_EVENT_CAT_IDS = "";
    private const string DEFAULT_AE_SUBSCRIPTION_FILTER_LOWEST_SEVERITY = "0";
    private const string DEFAULT_AE_SUBSCRIPTION_FILTER_HIGHEST_SEVERITY = "1000";
```

```
48:     private OpcSeverityMapping[] DEFAULT_AE_SUBSCRIPTION_SEVERITY_MAPPINGS = new OpcSeverityMapping[0] {};
49:     public static string DEFAULT_DA_SUBSCRIPTION_MESSAGE_FORMAT = "$itemname has value $value";
50:     private const string DEFAULT_DA_SUBSCRIPTION_ITEM = "";
51:     private const AlarmStateTypes DEFAULT_DA_SUBSCRIPTION_ALARM_STATE_TYPE = AlarmStateTypes.BOOLEAN;
52:     private const string DEFAULT_DA_SUBSCRIPTION_ALARM_STATE = "TRUE";
53:     private const string DEFAULT_DA_SUBSCRIPTION_ACS_ALARM_CODE = "NONE";
54:
55:     #endregion
56:
57:     /// <summary>
58:     /// Constructor taking the view as parameter
59:     /// </summary>
60:     /// <param name="view">The view associated with the Controller</param>
61:     public Controller(View view)
62:     {
63:       this.view = view;
64:       model = new Model();
65:     }
66:
67:     #region Utility Methods
68:
69:     private string GetNewServerName(int index)
70:     {
71:       ArrayList list = new ArrayList();
72:       list.AddRange(model.GetServers());
73:       string name = DEFAULT_SERVER_SERVER_NAME + " " + index;
74:       if (XmlConfiguration.IsInCollection(list.GetEnumerator(), name))
75:         return GetNewServerName(index+1);
76:       else
77:         return name;
78:     }
79:
80:     private string GetNewSubscriptionName(OpcServerNode server, int index)
81:     {
82:       string name = DEFAULT_SUBSCRIPTION_SUBSCRIPTION_NAME + " " + index;
83:       if (XmlConfiguration.IsInCollection(server.Nodes, name))
84:         return GetNewSubscriptionName(server, index+1);
85:       else
86:         return name;
87:     }
88:
89:     #endregion
90:
91:     #region Model Manipulation
92:
93:     public OpcServerNode AddServer()
94:     {
```

```
 95:       OpcServerNode server = new OpcServerNode(GetNewServerName(1));
 96:       server.OpcServerHost = DEFAULT_SERVER_SERVER_HOST;
 97:       server.AcsDefaultUnitId = DEFAULT_SERVER_DEFAULT_UNIT_ID;
 98:       server.AcsDefaultUnitId = DEFAULT_SERVER_DEFAULT_AREA_GROUPS;
 99:       server.UseCom = DEFAULT_SERVER_USE_COM;
100:       server.AcsAlarmCodes = DEFAULT_SERVER_ACS_ALARM_CODES;
101:       model.AddServer(server);
102:       view.UpdateTreeView(model.GetServers());
103:       return server;
104:     }
105:
106:     public void RemoveServer(OpcServerNode server)
107:     {
108:       model.RemoveServer(server);
109:       view.UpdateTreeView(model.GetServers());
110:     }
111:
112:     public void RemoveSubscription(OpcSubscriptionNode subscription, OpcServerNode server)
113:     {
114:       server.Nodes.Remove(subscription);
115:       view.UpdateTreeView(model.GetServers());
116:     }
117:
118:     public OpcSubscriptionNode AddAeSubscription(OpcServerNode server)
119:     {
120:       OpcAeSubscriptionNode subscription = new OpcAeSubscriptionNode(GetNewSubscriptionName(server, 1));
121:       subscription.Enabled = DEFAULT_SUBSCRIPTION_ENABLED;
122:       subscription.AcsUnitId = DEFAULT_SUBSCRIPTION_UNIT_ID;
123:       subscription.AcsAreaGroups = DEFAULT_SUBSCRIPTION_AREA_GROUPS;
124:       subscription.MessageFormat = DEFAULT_AE_SUBSCRIPTION_MESSAGE_FORMAT;
125:       subscription.FilterEnabled = DEFAULT_AE_SUBSCRIPTION_FILTER_ENABLED;
126:       subscription.Sources = DEFAULT_AE_SUBSCRIPTION_FILTER_SOURCES;
127:       subscription.Areas = DEFAULT_AE_SUBSCRIPTION_FILTER_AREAS;
128:       subscription.EventType = DEFAULT_AE_SUBSCRIPTION_FILTER_EVENT_TYPE;
129:       subscription.CategoryIds = DEFAULT_AE_SUBSCRIPTION_FILTER_EVENT_CAT_IDS;
130:       subscription.HighestSeverity = DEFAULT_AE_SUBSCRIPTION_FILTER_HIGHEST_SEVERITY;
131:       subscription.LowestSeverity = DEFAULT_AE_SUBSCRIPTION_FILTER_LOWEST_SEVERITY;
132:       subscription.SeverityMappings = DEFAULT_AE_SUBSCRIPTION_SEVERITY_MAPPINGS;
133:       subscription.ImageIndex = 2;
134:       subscription.SelectedImageIndex = 2;
135:       server.Nodes.Add(subscription);
136:       view.ShowAllNodesInTree();
137:       return subscription;
138:     }
139:
140:     public OpcSubscriptionNode AddDaSubscription(OpcServerNode server)
141:     {
```

```
142:       OpcDaSubscriptionNode subscription = new OpcDaSubscriptionNode(GetNewSubscriptionName(server, 1));
143:       subscription.Enabled = DEFAULT_SUBSCRIPTION_ENABLED;
144:       subscription.AcsUnitId = DEFAULT_SUBSCRIPTION_UNIT_ID;
145:       subscription.AcsAreaGroups = DEFAULT_SUBSCRIPTION_AREA_GROUPS;
146:       subscription.MessageFormat = DEFAULT_DA_SUBSCRIPTION_MESSAGE_FORMAT;
147:       subscription.Item = DEFAULT_DA_SUBSCRIPTION_ITEM;
148:       subscription.AlarmStateType = DEFAULT_DA_SUBSCRIPTION_ALARM_STATE_TYPE;
149:       subscription.AlarmState = DEFAULT_DA_SUBSCRIPTION_ALARM_STATE;
150:       subscription.AcsAlarmCode = DEFAULT_DA_SUBSCRIPTION_ACS_ALARM_CODE;
151:       subscription.ImageIndex = 3;
152:       subscription.SelectedImageIndex = 3;
153:       server.Nodes.Add(subscription);
154:       view.ShowAllNodesInTree();
155:       return subscription;
156:     }
157:
158:     public void ClearServers()
159:     {
160:       model.ClearServers();
161:       view.UpdateTreeView(model.GetServers());
162:     }
163:
164:     public void LoadConfiguration(String path)
165:     {
166:       OpcServerNode[] servers = XmlConfiguration.ReadConfigurationFromFile(path);
167:       // Ensure that all servers have the default acs alarm code
168:       foreach (OpcServerNode server in servers)
169:       {
170:         bool hasDefaultValue = false;
171:         foreach (string code in server.AcsAlarmCodes)
172:         {
173:           if (code.Equals(DEFAULT_SERVER_ACS_ALARM_CODES[0]))
174:           {
175:             hasDefaultValue = true;
176:             break;
177:           }
178:         }
179:         if (!hasDefaultValue)
180:         {
181:           string[] alarmCodes = new string[server.AcsAlarmCodes.Length + 1];
182:           alarmCodes[0] = DEFAULT_SERVER_ACS_ALARM_CODES[0];
183:           int count=1;
184:           foreach (string code in server.AcsAlarmCodes)
185:           {
186:             alarmCodes[count++] = code;
187:           }
188:           server.AcsAlarmCodes = alarmCodes;
```

```
189:         }
190:       }
191:       ClearServers();
192:       model.AddServers(servers);
193:       view.UpdateTreeView(model.GetServers());
194:     }
195:
196:     public void SaveConfiguration(String path)
197:     {
198:       XmlConfiguration.WriteConfigurationToFile(path, model.GetServers());
199:     }
200:
201:     #endregion
202:
203:     #region Validation
204:
205:     public bool ValidOpcRange(string opcRange)
206:     {
207:       try
208:       {
209:         // single number between 0 and 1000
210:         string single = @"(^\d{1,3}$)|(^1000$)";
211:         // range of numbers between 0-999 to 0-1000
212:         string range =  @"^\d{1,3}-(\d{1,3}|1000)$";
213:
214:         Match sMatch = Regex.Match(opcRange, single);
215:         Match rMatch = Regex.Match(opcRange, range);
216:
217:         if (sMatch.Success)
218:         {
219:           return true;;
220:         }
221:         else if (rMatch.Success)
222:         {
223:           string s1, s2;
224:           int n1, n2;
225:           s1 = opcRange.Substring(0, opcRange.IndexOf("-"));
226:           s2 = opcRange.Substring(opcRange.IndexOf("-")+1);
227:           n1 = Convert.ToInt32(s1);
228:           n2 = Convert.ToInt32(s2);
229:           if (n2 < n1)
230:           {
231:             return false;
232:           }
233:           else if (n2 == n1)
234:             return false;
235:           else
```

```
236:             return true;
237:           }
238:           else
239:             return false;
240:         }
241:         catch (Exception)
242:         {
243:           return false;
244:         }
245:       }
246:
247:       public bool ValidNumber(string theRange)
248:       {
249:         // single number (positive or negative)
250:         string single = @"^-?\d+$";
251:         Match sMatch = Regex.Match(theRange, single);
252:         if (sMatch.Success)
253:           return true;
254:         else
255:           return false;
256:       }
257:
258:       public bool ServerNameInUse(string serverName)
259:       {
260:         ArrayList list = new ArrayList();
261:         list.AddRange(model.GetServers());
262:             return XmlConfiguration.IsInCollection(list.GetEnumerator(), serverName, 2);
263:       }
264:
265:       public bool SubscriptionNameInUse(string serverName, OpcServerNode server)
266:       {
267:             return XmlConfiguration.IsInCollection(server.Nodes, serverName, 2);
268:       }
269:
270:       #endregion
271:
272:     }
273: }
274:
```

```
1: using System;
2: using System.Collections;
3: using Zonith.Configuration.OpcListener;
4:
5: namespace Zonith.Listener.OpcListener.Configuration
6: {
7:   /// <summary>
8:   /// Summary description for Model.
9:   /// </summary>
10:  public class Model
11:  {
12:    private ArrayList servers;
13:
14:    public Model()
15:    {
16:      servers = new ArrayList();
17:    }
18:
19:    public void AddServer(OpcServerNode server)
20:    {
21:      servers.Add(server);
22:    }
23:
24:    public void AddServers(OpcServerNode[] servers)
25:    {
26:      this.servers.AddRange(servers);
27:    }
28:
29:    public void RemoveServer(OpcServerNode server)
30:    {
31:      servers.Remove(server);
32:    }
33:
34:    public object[] GetServers()
35:    {
36:      return servers.ToArray();
37:    }
38:
39:    public void ClearServers()
40:    {
41:      servers.Clear();
42:    }
43:  }
44: }
45:
```

```
/* ************************************
 *       ModifyRegistry.cs
 * ------------------------------------
 *         a very simple class
 *    to read, write, delete and count
 *       registry values with C#
 * ------------------------------------
 *      if you improve this code
 *   please email me your improvement!
 * ------------------------------------
 *         by Francesco Natali
 *        - fn.varie@libero.it -
 * ************************************/

using System;
// it's required for reading/writing into the registry:
using Microsoft.Win32;
// and for the MessageBox function:
using System.Windows.Forms;

namespace Utility.ModifyRegistry
{
  /// <summary>
  /// An useful class to read/write/delete/count registry keys
  /// </summary>
  public class ModifyRegistry
  {
    private bool showError = false;
    /// <summary>
    /// A property to show or hide error messages
    /// (default = false)
    /// </summary>
    public bool ShowError
    {
      get { return showError; }
      set { showError = value; }
    }

    private string subKey = "SOFTWARE\\" + Application.ProductName.ToUpper();
    /// <summary>
    /// A property to set the SubKey value
    /// (default = "SOFTWARE\\" + Application.ProductName.ToUpper())
    /// </summary>
    public string SubKey
    {
      get { return subKey; }
      set { subKey = value; }
```

```
48:     }
49:
50:     private RegistryKey baseRegistryKey = Registry.LocalMachine;
51:     /// <summary>
52:     /// A property to set the BaseRegistryKey value.
53:     /// (default = Registry.LocalMachine)
54:     /// </summary>
55:     public RegistryKey BaseRegistryKey
56:     {
57:       get { return baseRegistryKey; }
58:       set { baseRegistryKey = value; }
59:     }
60:
61:     /* *****************************************************************************
62:      * *****************************************************************************/
63:
64:     /// <summary>
65:     /// To read a registry key.
66:     /// input: KeyName (string)
67:     /// output: value (string)
68:     /// </summary>
69:     public string Read(string KeyName)
70:     {
71:       // Opening the registry key
72:       RegistryKey rk = baseRegistryKey ;
73:       // Open a subKey as read-only
74:       RegistryKey sk1 = rk.OpenSubKey(subKey);
75:       // If the RegistrySubKey doesn't exist -> (null)
76:       if ( sk1 == null )
77:       {
78:         return null;
79:       }
80:       else
81:       {
82:         try
83:         {
84:           // If the RegistryKey exists I get its value
85:           // or null is returned.
86:           return (string)sk1.GetValue(KeyName.ToUpper());
87:         }
88:         catch (Exception e)
89:         {
90:           // AAAAAAAAAAARGH, an error!
91:           ShowErrorMessage(e, "Reading registry " + KeyName.ToUpper());
92:           return null;
93:         }
94:       }
```

```
 95:     }
 96:
 97:     /* ************************************************************************
 98:      * ************************************************************************/
 99:
100:     /// <summary>
101:     /// To write into a registry key.
102:     /// input: KeyName (string) , Value (object)
103:     /// output: true or false
104:     /// </summary>
105:     public bool Write(string KeyName, object Value)
106:     {
107:       try
108:       {
109:         // Setting
110:         RegistryKey rk = baseRegistryKey ;
111:         // I have to use CreateSubKey
112:         // (create or open it if already exits),
113:         // 'cause OpenSubKey open a subKey as read-only
114:         RegistryKey sk1 = rk.CreateSubKey(subKey);
115:         // Save the value
116:         sk1.SetValue(KeyName.ToUpper(), Value);
117:
118:         return true;
119:       }
120:       catch (Exception e)
121:       {
122:         // AAAAAAAAAAARGH, an error!
123:         ShowErrorMessage(e, "Writing registry " + KeyName.ToUpper());
124:         return false;
125:       }
126:     }
127:
128:     /* ************************************************************************
129:      * ************************************************************************/
130:
131:     /// <summary>
132:     /// To delete a registry key.
133:     /// input: KeyName (string)
134:     /// output: true or false
135:     /// </summary>
136:     public bool DeleteKey(string KeyName)
137:     {
138:       try
139:       {
140:         // Setting
141:         RegistryKey rk = baseRegistryKey ;
```

```
142:         RegistryKey sk1 = rk.CreateSubKey(subKey);
143:         // If the RegistrySubKey doesn't exists -> (true)
144:         if ( sk1 == null )
145:           return true;
146:         else
147:           sk1.DeleteValue(KeyName);
148:
149:         return true;
150:       }
151:       catch (Exception e)
152:       {
153:         // AAAAAAAAAAARGH, an error!
154:         ShowErrorMessage(e, "Deleting SubKey " + subKey);
155:         return false;
156:       }
157:     }
158:
159:     /* ********************************************************************
160:      * ********************************************************************/
161:
162:     /// <summary>
163:     /// To delete a sub key and any child.
164:     /// input: void
165:     /// output: true or false
166:     /// </summary>
167:     public bool DeleteSubKeyTree()
168:     {
169:       try
170:       {
171:         // Setting
172:         RegistryKey rk = baseRegistryKey ;
173:         RegistryKey sk1 = rk.OpenSubKey(subKey);
174:         // If the RegistryKey exists, I delete it
175:         if ( sk1 != null )
176:           rk.DeleteSubKeyTree(subKey);
177:
178:         return true;
179:       }
180:       catch (Exception e)
181:       {
182:         // AAAAAAAAAAARGH, an error!
183:         ShowErrorMessage(e, "Deleting SubKey " + subKey);
184:         return false;
185:       }
186:     }
187:
188:     /* ********************************************************************
```

```
189:      * ************************************************************************/
190:
191:     /// <summary>
192:     /// Retrive the count of subkeys at the current key.
193:     /// input: void
194:     /// output: number of subkeys
195:     /// </summary>
196:     public int SubKeyCount()
197:     {
198:       try
199:       {
200:         // Setting
201:         RegistryKey rk = baseRegistryKey ;
202:         RegistryKey sk1 = rk.OpenSubKey(subKey);
203:         // If the RegistryKey exists...
204:         if ( sk1 != null )
205:           return sk1.SubKeyCount;
206:         else
207:           return 0;
208:       }
209:       catch (Exception e)
210:       {
211:         // AAAAAAAAAAARGH, an error!
212:         ShowErrorMessage(e, "Retriving subkeys of " + subKey);
213:         return 0;
214:       }
215:     }
216:
217:     /* ************************************************************************
218:      * ************************************************************************/
219:
220:     /// <summary>
221:     /// Retrive the count of values in the key.
222:     /// input: void
223:     /// output: number of keys
224:     /// </summary>
225:     public int ValueCount()
226:     {
227:       try
228:       {
229:         // Setting
230:         RegistryKey rk = baseRegistryKey ;
231:         RegistryKey sk1 = rk.OpenSubKey(subKey);
232:         // If the RegistryKey exists...
233:         if ( sk1 != null )
234:           return sk1.ValueCount;
235:         else
```

```
236:             return 0;
237:         }
238:         catch (Exception e)
239:         {
240:           // AAAAAAAAAARGH, an error!
241:           ShowErrorMessage(e, "Retriving keys of " + subKey);
242:           return 0;
243:         }
244:       }
245:
246:       /* ********************************************************************
247:        * ********************************************************************/
248:
249:       private void ShowErrorMessage(Exception e, string Title)
250:       {
251:         if (showError == true)
252:           MessageBox.Show(e.Message,
253:                   Title
254:                   ,MessageBoxButtons.OK
255:                   ,MessageBoxIcon.Error);
256:       }
257:     }
258: }
259:
```

```cs
using System;
using System.Collections;
using System.IO;

namespace Zonith.Listener.OpcListener.Configuration
{
  /// <summary>
  /// Simple implementation of a Properties class as known from java.util.Properties
  ///
  /// Note: Implemented using framework specific exceptions
  /// </summary>
  public class Properties
  {
    /// <summary>
    /// Hashtable used to hold all the key/value pairs
    /// </summary>
    Hashtable hash = new Hashtable();

    /// <summary>
    /// Method used to load the properties from the given filename
    /// </summary>
    /// <param name="path">The file from which to load the properties</param>
    public void Load(String path)
    {
      try
      {
        TextReader reader = new StreamReader(path, System.Text.Encoding.Unicode);
        String line = reader.ReadLine();
        int index;
        while (line != null)
        {
          if (!line.StartsWith("#") && !(line.Trim().Length==0))
          {
            index = line.IndexOf("=");
            if (index != -1)
            {
              hash.Add(line.Substring(0,index), line.Substring(index+1));
            }
            else
            {
              throw new Exception("Corrupted language file! [" + path + "]");
            }
          }
          line = reader.ReadLine();
        }
        reader.Close();
      }
```

```
48:       catch (FileNotFoundException e)
49:       {
50:         throw new Exception("Language file not found [" + path + "]", e);
51:       }
52:     }
53:
54:     /// <summary>
55:     /// Method which returns a value corresponding to the given key
56:     /// </summary>
57:     /// <param name="key">The key who's value should be returned</param>
58:     /// <returns>The value associated with the given key, null if it does not exist</returns>
59:     public String GetProperty(String key)
60:     {
61:       return (String) hash[key];
62:     }
63:   }
64: }
65:
```

```
1: using System;
2:
3: namespace Zonith.Listener.OpcListener.Configuration
4: {
5:   /// <summary>
6:   /// A wrapper around the Properties class implementing a GetProperty method
7:   /// which takes a default return value as parameter along with the key.
8:   /// </summary>
9:   public class PropertyFileHelper
10:   {
11:     /// <summary>
12:     /// The Properties object used to load the properties
13:     /// </summary>
14:     private Properties properties;
15:
16:     /// <summary>
17:     /// Constructor taking the path to the property file as parameter
18:     /// </summary>
19:     /// <param name="path">The path to the property file</param>
20:     public PropertyFileHelper(String path)
21:     {
22:       properties = new Properties();
23:       properties.Load(path);
24:     }
25:
26:     /// <summary>
27:     /// Method which returns the value associated with the given key. If the key does
28:     /// not exist in the property file, the default value is returned
29:     /// </summary>
30:     /// <param name="key">The key who's value should be returned</param>
31:     /// <param name="defaultVal">The default value to use if the key does not exist in the property file</param>
32:     /// <returns>The value associated with the key if present, defaultVal otherwise</returns>
33:     public String GetProperty(String key, String defaultVal)
34:     {
35:       String res = properties.GetProperty(key);
36:       if (res == null) {
37:         return defaultVal;
38:       }
39:       return res;
40:     }
41:
42:     /// <summary>
43:     /// Method which returns the value associated with the given key.
44:     /// </summary>
45:     /// <param name="key">The key who's value should be returned</param>
46:     /// <returns>The value associated with the key if present, null otherwise</returns>
47:     public String GetProperty(String key)
```

```
48:     {
49:       String res = properties.GetProperty(key);
50:       if (res == null) {
51:         throw new Exception("Missing property " + key);
52:       }
53:       return res;
54:     }
55:
56:     /// <summary>
57:     /// Method which returns the Properties object contained in the PropertyFileHelper
58:     /// </summary>
59:     /// <returns>The Properties object contained in the PropertyFileHelper</returns>
60:     public Properties GetProperties()
61:     {
62:         return properties;
63:     }
64:   }
65: }
```

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using TsOpcNet;

namespace Zonith.Listener.OpcListener.Configuration
{
  public enum ConnectionType { COM, XML };
  public enum Item { ITEM, SOURCES, AREAS, CAT_IDS, SERVER };

  /// <summary>
  /// Summary description for SelectItemBox.
  /// </summary>
  public class SelectItemBox : System.Windows.Forms.Form
  {
    public static string TITLE_SERVER = "";
    public static string TITLE_OTHER = "";
    public static string ERROR_TITLE = "";
    public static string ERROR_MESSAGE_SELECT_ITEM = "";
    public static string LABEL_SOURCES = "";
    public static string LABEL_AREAS = "";
    public static string LABEL_EVENT_CAT_IDS = "";
    public static string LABEL_ITEM = "";
    public static string LABEL_HOST = "";
    public static string BUTTON_LIST_SERVERS_TEXT = "";
    public static string BUTTON_OK_TEXT = "";

    private View view;
    private string serverName;
    private string serverHost;
    private ConnectionType connectionType;
    private Item item;
    private string defaultAdr;

    private System.Windows.Forms.Label labelServerName;
    private System.Windows.Forms.ListBox listBoxItems;
    private System.Windows.Forms.Button buttonSelectItems;
    private System.Windows.Forms.Panel panelSubscription;
    private System.Windows.Forms.Panel panelServer;
    private System.Windows.Forms.Label labelHost;
    private System.Windows.Forms.Button buttonList;
    private System.Windows.Forms.TextBox textBoxHost;
    /// <summary>
    /// Required designer variable.
```

```
48:     /// </summary>
49:     private System.ComponentModel.Container components = null;
50:
51:     #region Constructor/Destructor
52:
53:     public SelectItemBox(View view, string serverName, string serverHost, ConnectionType connectionType, Item item, string
          defaultAdr)
54:     {
55:       //
56:       // Required for Windows Form Designer support
57:       //
58:       InitializeComponent();
59:
60:       this.view = view;
61:       this.serverName = serverName;
62:       this.serverHost = serverHost;
63:       this.connectionType = connectionType;
64:       this.item = item;
65:       this.defaultAdr = defaultAdr;
66:       panelSubscription.Bounds = panelServer.Bounds;
67:     }
68:
69:     /// <summary>
70:     /// Clean up any resources being used.
71:     /// </summary>
72:     protected override void Dispose( bool disposing )
73:     {
74:       if( disposing )
75:       {
76:         if(components != null)
77:         {
78:           components.Dispose();
79:         }
80:       }
81:       base.Dispose( disposing );
82:     }
83:
84:     #endregion
85:
86:     #region Windows Form Designer generated code
87:     /// <summary>
88:     /// Required method for Designer support - do not modify
89:     /// the contents of this method with the code editor.
90:     /// </summary>
91:     private void InitializeComponent()
92:     {
93:       this.labelServerName = new System.Windows.Forms.Label();
```

```
94:         this.listBoxItems = new System.Windows.Forms.ListBox();
95:         this.buttonSelectItems = new System.Windows.Forms.Button();
96:         this.panelSubscription = new System.Windows.Forms.Panel();
97:         this.panelServer = new System.Windows.Forms.Panel();
98:         this.buttonList = new System.Windows.Forms.Button();
99:         this.labelHost = new System.Windows.Forms.Label();
100:        this.textBoxHost = new System.Windows.Forms.TextBox();
101:        this.panelSubscription.SuspendLayout();
102:        this.panelServer.SuspendLayout();
103:        this.SuspendLayout();
104:        //
105:        // labelServerName
106:        //
107:        this.labelServerName.Dock = System.Windows.Forms.DockStyle.Left;
108:        this.labelServerName.Location = new System.Drawing.Point(0, 0);
109:        this.labelServerName.Name = "labelServerName";
110:        this.labelServerName.Size = new System.Drawing.Size(408, 32);
111:        this.labelServerName.TabIndex = 0;
112:        this.labelServerName.Text = "Server:";
113:        this.labelServerName.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
114:        //
115:        // listBoxItems
116:        //
117:        this.listBoxItems.Dock = System.Windows.Forms.DockStyle.Fill;
118:        this.listBoxItems.Location = new System.Drawing.Point(0, 64);
119:        this.listBoxItems.Name = "listBoxItems";
120:        this.listBoxItems.Size = new System.Drawing.Size(352, 199);
121:        this.listBoxItems.TabIndex = 2;
122:        this.listBoxItems.DoubleClick += new System.EventHandler(this.listBoxItems_DoubleClick);
123:        //
124:        // buttonSelectItems
125:        //
126:        this.buttonSelectItems.Dock = System.Windows.Forms.DockStyle.Bottom;
127:        this.buttonSelectItems.FlatStyle = System.Windows.Forms.FlatStyle.System;
128:        this.buttonSelectItems.Location = new System.Drawing.Point(0, 271);
129:        this.buttonSelectItems.Name = "buttonSelectItems";
130:        this.buttonSelectItems.Size = new System.Drawing.Size(352, 23);
131:        this.buttonSelectItems.TabIndex = 3;
132:        this.buttonSelectItems.Text = "Ok";
133:        this.buttonSelectItems.Click += new System.EventHandler(this.buttonSelectItems_Click);
134:        //
135:        // panelSubscription
136:        //
137:        this.panelSubscription.Controls.Add(this.labelServerName);
138:        this.panelSubscription.Dock = System.Windows.Forms.DockStyle.Top;
139:        this.panelSubscription.Location = new System.Drawing.Point(0, 0);
140:        this.panelSubscription.Name = "panelSubscription";
```

```
141:        this.panelSubscription.Size = new System.Drawing.Size(352, 32);
142:        this.panelSubscription.TabIndex = 4;
143:        //
144:        // panelServer
145:        //
146:        this.panelServer.Controls.Add(this.buttonList);
147:        this.panelServer.Controls.Add(this.labelHost);
148:        this.panelServer.Controls.Add(this.textBoxHost);
149:        this.panelServer.Dock = System.Windows.Forms.DockStyle.Top;
150:        this.panelServer.Location = new System.Drawing.Point(0, 32);
151:        this.panelServer.Name = "panelServer";
152:        this.panelServer.Size = new System.Drawing.Size(352, 32);
153:        this.panelServer.TabIndex = 5;
154:        this.panelServer.Visible = false;
155:        //
156:        // buttonList
157:        //
158:        this.buttonList.FlatStyle = System.Windows.Forms.FlatStyle.System;
159:        this.buttonList.Location = new System.Drawing.Point(248, 1);
160:        this.buttonList.Name = "buttonList";
161:        this.buttonList.Size = new System.Drawing.Size(96, 23);
162:        this.buttonList.TabIndex = 2;
163:        this.buttonList.Text = "List Servers";
164:        this.buttonList.Click += new System.EventHandler(this.buttonList_Click);
165:        //
166:        // labelHost
167:        //
168:        this.labelHost.Location = new System.Drawing.Point(0, 8);
169:        this.labelHost.Name = "labelHost";
170:        this.labelHost.Size = new System.Drawing.Size(56, 16);
171:        this.labelHost.TabIndex = 0;
172:        this.labelHost.Text = "Host:";
173:        this.labelHost.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
174:        //
175:        // textBoxHost
176:        //
177:        this.textBoxHost.Location = new System.Drawing.Point(56, 4);
178:        this.textBoxHost.Name = "textBoxHost";
179:        this.textBoxHost.Size = new System.Drawing.Size(184, 20);
180:        this.textBoxHost.TabIndex = 1;
181:        this.textBoxHost.Text = "localhost";
182:        //
183:        // SelectItemBox
184:        //
185:        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
186:        this.ClientSize = new System.Drawing.Size(352, 294);
187:        this.Controls.Add(this.listBoxItems);
```

```
188:       this.Controls.Add(this.buttonSelectItems);
189:       this.Controls.Add(this.panelServer);
190:       this.Controls.Add(this.panelSubscription);
191:       this.MaximizeBox = false;
192:       this.MaximumSize = new System.Drawing.Size(360, 328);
193:       this.MinimumSize = new System.Drawing.Size(360, 328);
194:       this.Name = "SelectItemBox";
195:       this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
196:       this.Text = "SelectItemBox";
197:       this.panelSubscription.ResumeLayout(false);
198:       this.panelServer.ResumeLayout(false);
199:       this.ResumeLayout(false);
200:
201:     }
202:     #endregion
203:
204:     #region Initialization
205:
206:     public bool InitSelectItemBox()
207:     {
208:       try
209:       {
210:         buttonSelectItems.Text = BUTTON_OK_TEXT;
211:         switch(item)
212:         {
213:           case Item.SERVER:
214:             panelSubscription.Visible = false;
215:             panelServer.Visible = true;
216:             this.Text = TITLE_SERVER;
217:             labelHost.Text = LABEL_HOST;
218:             buttonList.Text = BUTTON_LIST_SERVERS_TEXT;
219:             listBoxItems.SelectionMode = SelectionMode.One;
220:             textBoxHost.Text = (defaultAdr == null || defaultAdr.Equals("")) ? "localhost" : defaultAdr;
221:             if (connectionType == ConnectionType.COM)
222:               FetchComServers();
223:             else if (connectionType == ConnectionType.XML)
224:               FetchXmlServers();
225:             break;
226:           case Item.SOURCES:
227:             this.Text = TITLE_OTHER + " " + serverName;
228:             labelServerName.Text = LABEL_SOURCES;
229:             listBoxItems.SelectionMode = SelectionMode.MultiExtended;
230:             if (connectionType == ConnectionType.COM)
231:               FetchSourcesFromAeComServer();
232:             else if (connectionType == ConnectionType.XML)
233:               FetchSourcesFromAeXmlServer();
234:             break;
```

```
235:           case Item.AREAS:
236:             this.Text = TITLE_OTHER + " " + serverName;
237:             labelServerName.Text = LABEL_AREAS;
238:             listBoxItems.SelectionMode = SelectionMode.MultiExtended;
239:             if (connectionType == ConnectionType.COM)
240:               FetchAreasFromAeComServer();
241:             else if (connectionType == ConnectionType.XML)
242:               FetchAreasFromAeXmlServer();
243:             break;
244:           case Item.CAT_IDS:
245:             this.Text = TITLE_OTHER + " " + serverName;
246:             labelServerName.Text = LABEL_EVENT_CAT_IDS;
247:             listBoxItems.SelectionMode = SelectionMode.MultiExtended;
248:             if (connectionType == ConnectionType.COM)
249:               FetchEventCatIdsFromAeComServer();
250:             else if (connectionType == ConnectionType.XML)
251:               FetchEventCatIdsFromAeXmlServer();
252:             break;
253:           case Item.ITEM:
254:             this.Text = TITLE_OTHER + " " + serverName;
255:             labelServerName.Text = LABEL_ITEM;
256:             listBoxItems.SelectionMode = SelectionMode.One;
257:             if (connectionType == ConnectionType.COM)
258:               FetchItemsFromDaComServer();
259:             else if (connectionType == ConnectionType.XML)
260:               FetchItemsFromDaXmlServer();
261:             break;
262:         }
263:
264:       }
265:       catch (Exception e)
266:       {
267:         MessageBox.Show(ERROR_MESSAGE_SELECT_ITEM + "\n\nException: " + e.Message, ERROR_TITLE);
268:         return false;
269:       }
270:       return true;
271:     }
272:
273:     #endregion
274:
275:     #region COM Servers
276:
277:     private void FetchComServers()
278:     {
279:       listBoxItems.Items.Clear();
280:       TsOpcComputerInfo host = new TsOpcComputerInfo(textBoxHost.Text);
281:       TsOpcBrowse serverListing = new TsOpcBrowse();
```

```
282:       TsOpcResult res;
283:       TsOpcServerInfo[] si;
284:       res = serverListing.ListAeServers(out si, host);
285:
286:       if (res.Failed())
287:         MessageBox.Show("ERROR " + res.Code + " : " + res.Name + " : " + res.Description());
288:       if (res.IsSuccess())
289:       {
290:         if (si != null)
291:         {
292:           for (int i = si.GetLowerBound(0); i <= si.GetUpperBound(0); i++)
293:           {
294:             AddItemToList(si[i].m_ServerName);
295:           }
296:         }
297:       }
298:       else if (res.IsError())
299:       {
300:         throw new Exception("1 - " + res.Description() + " - " + res.Name);
301:       }
302:       res = serverListing.ListDaServers(out si, host);
303:       if (res.IsSuccess())
304:       {
305:         if (si != null)
306:         {
307:           for (int i = si.GetLowerBound(0); i <= si.GetUpperBound(0); i++)
308:           {
309:             AddItemToList(si[i].m_ServerName);
310:           }
311:         }
312:       }
313:       else if (res.IsError())
314:       {
315:         throw new Exception("2 - " + res.Description() + " - " + res.Name);
316:       }
317:     }
318:
319:     #endregion
320:
321:     #region XML Servers
322:
323:     private void FetchXmlServers()
324:     {
325:     }
326:
327:     #endregion
328:
```

```
329:     #region AE COM Sources
330:
331:     private void FetchSourcesFromAeComServer()
332:     {
333:       TsOpcComputerInfo host = new TsOpcComputerInfo(serverHost);
334:       TsCAeServer comServer = new TsCAeServer();
335:       comServer.Connect(serverName, host);
336:       TsCAeBrowseElement[] be = comServer.Browse("", TsCAeBrowseType.Area, "");
337:       AeComBrowseAreas(comServer, be);
338:       AeComAddSourcesFromArea("", comServer);
339:       comServer.Disconnect();
340:     }
341:
342:     private void AeComBrowseAreas(TsCAeServer comServer, TsCAeBrowseElement[] elements)
343:     {
344:       foreach(TsCAeBrowseElement be in elements)
345:       {
346:         AeComAddSourcesFromArea(be.QualifiedName, comServer);
347:         AeComBrowseAreas(comServer, comServer.Browse(be.QualifiedName, TsCAeBrowseType.Area, ""));
348:       }
349:     }
350:
351:     private void AeComAddSourcesFromArea(string area, TsCAeServer comServer)
352:     {
353:       foreach (TsCAeBrowseElement be in comServer.Browse(area, TsCAeBrowseType.Source, ""))
354:       {
355:         AddItemToList(be.QualifiedName);
356:       }
357:     }
358:
359:     #endregion
360:
361:     #region AE XML Sources
362:
363:     private void FetchSourcesFromAeXmlServer()
364:     {
365:     }
366:
367:     #endregion
368:
369:     #region AE COM Areas
370:
371:     private void FetchAreasFromAeComServer()
372:     {
373:       TsOpcComputerInfo host = new TsOpcComputerInfo(serverHost);
374:       TsCAeServer comServer = new TsCAeServer();
375:       comServer.Connect(serverName, host);
```

```
376:       TsCAeBrowseElement[] be = comServer.Browse("", TsCAeBrowseType.Area, "");
377:       AeComAddAreas(comServer, be);
378:       comServer.Disconnect();
379:     }
380:
381:     private void AeComAddAreas(TsCAeServer comServer, TsCAeBrowseElement[] elements)
382:     {
383:       foreach(TsCAeBrowseElement be in elements)
384:       {
385:         AddItemToList(be.QualifiedName);
386:         AeComAddAreas(comServer, comServer.Browse(be.QualifiedName, TsCAeBrowseType.Area, ""));
387:       }
388:     }
389:
390:     #endregion
391:
392:     #region AE XML Areas
393:
394:     private void FetchAreasFromAeXmlServer()
395:     {
396:     }
397:
398:     #endregion
399:
400:     #region AE COM Event Category Ids
401:
402:     private void FetchEventCatIdsFromAeComServer()
403:     {
404:       TsOpcComputerInfo host = new TsOpcComputerInfo(serverHost);
405:       TsCAeServer comServer = new TsCAeServer();
406:       comServer.Connect(serverName, host);
407:       //TsCAeCategory[] categories = comServer.QueryEventCategories((int) TsCAeEventType.All);  // Does not work!
408:       TsCAeCategory[] categories = comServer.QueryEventCategories(7);
409:       foreach (TsCAeCategory cat in categories)
410:       {
411:         AddItemToList(cat.Name + " (" + cat.ID + ")");
412:       }
413:       comServer.Disconnect();
414:     }
415:
416:     #endregion
417:
418:     #region AE XML Event Category Ids
419:
420:     private void FetchEventCatIdsFromAeXmlServer()
421:     {
422:     }
```

```
423:
424:     #endregion
425:
426:     #region DA COM Items
427:
428:     private void FetchItemsFromDaComServer()
429:     {
430:       TsCDaBrowseElement[] be;
431:       TsCDaBrowsePosition bp;
432:       TsOpcItem path = new TsOpcItem("");
433:
434:       TsOpcComputerInfo host = new TsOpcComputerInfo(serverHost);
435:       TsCDaServer comServer = new TsCDaServer();
436:       comServer.Connect(serverName, host);
437:       be = comServer.Browse(path, new TsCDaBrowseFilters(), out bp);
438:       DaComAddItems(comServer, bp, be);
439:       comServer.Disconnect();
440:     }
441:
442:     private void DaComAddItems(TsCDaServer comServer, TsCDaBrowsePosition bp, TsCDaBrowseElement[] elements)
443:     {
444:       foreach (TsCDaBrowseElement be in elements)
445:       {
446:         if (be.IsItem)
447:         {
448:           AddItemToList(be.ItemPath + be.ItemName);
449:         }
450:         else
451:         {
452:           if (be.HasChildren)
453:           {
454:             string path = (be.ItemPath == null || be.ItemPath.Equals("")) ? be.ItemName : be.ItemPath;
455:             DaComAddItems(comServer, bp, comServer.Browse(new TsOpcItem(path), new TsCDaBrowseFilters(), out bp));
456:           }
457:         }
458:       }
459:     }
460:
461:     #endregion
462:
463:     #region DA XML Items
464:
465:     private void FetchItemsFromDaXmlServer()
466:     {
467:     }
468:
469:     #endregion
```

```
470:
471:     #region Event Handlers
472:
473:     private void buttonSelectItems_Click(object sender, System.EventArgs e)
474:     {
475:       object obj;
476:       string items;
477:       switch(item)
478:       {
479:         case Item.SERVER:
480:           obj = listBoxItems.SelectedItem;
481:           if (obj != null && !((string) obj).Equals(""))
482:             view.UpdateOpcServerNameRemote((string) obj, textBoxHost.Text);
483:           break;
484:         case Item.SOURCES:
485:           items = "";
486:           foreach (object o in listBoxItems.SelectedItems)
487:           {
488:             items += o.ToString() + ", ";
489:
490:           }
491:           if (!items.Equals(""))
492:           {
493:             items = items.Substring(0, items.LastIndexOf(","));
494:             view.UpdateAeSubscriptionFilterSources(items);
495:           }
496:           break;
497:         case Item.AREAS:
498:           items = "";
499:           foreach (object o in listBoxItems.SelectedItems)
500:           {
501:             items += o.ToString() + ", ";
502:
503:           }
504:           if (!items.Equals(""))
505:           {
506:             items = items.Substring(0, items.LastIndexOf(","));
507:             view.UpdateAeSubscriptionFilterAreas(items);
508:           }
509:           break;
510:         case Item.CAT_IDS:
511:           items = "";
512:           foreach (object o in listBoxItems.SelectedItems)
513:           {
514:             String str = o.ToString();
515:                         items += str.Substring(str.LastIndexOf("(")+1, str.LastIndexOf(")")-(str.LastIndexOf("(")+1)) + ", "
                              ;
```

```
516:           }
517:           if (!items.Equals(""))
518:           {
519:             items = items.Substring(0, items.LastIndexOf(","));
520:             view.UpdateAeSubscriptionFilterEventCatIds(items);
521:           }
522:           break;
523:         case Item.ITEM:
524:           obj = listBoxItems.SelectedItem;
525:           if (obj != null && !((string) obj).Equals(""))
526:             view.UpdateDaSubscriptionItem((string) obj);
527:           break;
528:       }
529:       Close();
530:     }
531:
532:     private void buttonList_Click(object sender, System.EventArgs e)
533:     {
534:       if (connectionType == ConnectionType.COM)
535:         FetchComServers();
536:       else if (connectionType == ConnectionType.XML)
537:         FetchXmlServers();
538:     }
539:
540:     private void listBoxItems_DoubleClick(object sender, System.EventArgs e)
541:     {
542:       buttonSelectItems_Click(sender, e);
543:     }
544:
545:     #endregion
546:
547:     public void AddItemToList(String item)
548:     {
549:       if (!listBoxItems.Items.Contains(item))
550:         listBoxItems.Items.Add(item);
551:     }
552:   }
553: }
554:
```

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Threading;
using System.ServiceProcess;
using System.Xml;
using Microsoft.Win32;
using Zonith.Configuration.OpcListener;
using Utility.ModifyRegistry;

namespace Zonith.Listener.OpcListener.Configuration
{
  /// <summary>
  /// Delegates used to show a progress bar while starting/stopping the service
  /// </summary>
  public delegate void DelegateUpdateProgress(int newValue);
  public delegate void DelegateInitProgress(int max);

  /// <summary>
  /// Enumeration indication an operation performed on the service
  /// </summary>
  public enum ServiceAction { Stopped, Started };

  /// <summary>
  /// Summary description for Form1.
  /// </summary>
  public class View : System.Windows.Forms.Form
  {
    /// <summary>
    /// Delegate references
    /// </summary>
    public DelegateUpdateProgress delegateUpdateProgress;
    public DelegateInitProgress delegateInitProgress;

    /// <summary>
    /// Variable (dynamically changed) indicating the amount of time spent when processing service actions
    /// </summary>
    private int time = 15;

    /// <summary>
    /// Indication of which action was last performed on the service
    /// </summary>
    private ServiceAction lastAction;

```

```
48:     /// <summary>
49:     /// Indicates whether or not the service is in process (starting/stopping)
50:     /// </summary>
51:     private bool serviceActionInProcess = false;
52:
53:     /// <summary>
54:     /// The name of the service which can be started/stopped from the user interface
55:     /// </summary>
56:     private const string SERVICE_NAME = "ACS OPC Listener Service";
57:
58:     /// <summary>
59:     /// Registry location where the path to the last used language file is stored
60:     /// </summary>
61:     private const string REGISTRY_SUB_KEY = "SOFTWARE\\ZONITH\\OPC Listener Configuration";
62:     private const string REGISTRY_VALUE = "language";
63:
64:     /// <summary>
65:     /// Language files
66:     /// </summary>
67:     private const string LANGUAGE_ENGLISH = "languages\\english.lan";
68:     private const string LANGUAGE_DANISH = "languages\\danish.lan";
69:
70:     /// <summary>
71:     /// Location of the last loaded/saved configuration
72:     /// </summary>
73:     private String currentConfigurationPath = null;
74:
75:     /// <summary>
76:     /// Indication of whether of not the data has changed since last save
77:     /// </summary>
78:     private bool dirty = false;
79:
80:     /// <summary>
81:     /// Reference to the controller object
82:     /// </summary>
83:     private Controller controller;
84:
85:     private String HOME_DIRECTORY = null;
86:
87:     /// <summary>
88:     /// Default string to place in the status bar. These are language dependent, and loaded dynamically
89:     /// </summary>
90:     private string DEFAULT_STATUS_BAR_TEXT;
91:     private string STATUS_BAR_REMOVE_SERVER;
92:     private string STATUS_BAR_REMOVE_SUBSCRIPTION;
93:     private string STATUS_BAR_ADD_SERVER;
94:     private string STATUS_BAR_ADD_AE_SUBSCRIPTION;
```

```
 95:     private string STATUS_BAR_ADD_DA_SUBSCRIPTION;
 96:     private string STATUS_BAR_ADD_ACS_ALARM_CODE;
 97:     private string STATUS_BAR_ADD_SEVERITY_MAPPING;
 98:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_SOURCE;
 99:         private string STATUS_BAR_ADD_MESSAGE_FORMAT_SEVERITY;
100:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_MESSAGE;
101:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_CONDITION;
102:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_VALUE;
103:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_NAME;
104:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_PATH;
105:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_RESULT;
106:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_SERVER;
107:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_SUBSCRIPTION;
108:     private string STATUS_BAR_ADD_MESSAGE_FORMAT_FORMULA;
109:     private string STATUS_BAR_SELECT_SERVER;
110:     private string STATUS_BAR_SELECT_SOURCES;
111:     private string STATUS_BAR_SELECT_AREAS;
112:     private string STATUS_BAR_SELECT_EVENT_CAT_IDS;
113:     private string STATUS_BAR_SELECT_ITEM;
114:     private string STATUS_BAR_START_SERVICE;
115:     private string STATUS_BAR_STOP_SERVICE;
116:
117:     /// <summary>
118:     /// The last selected node in the server tree view (used when saving changes)
119:     /// </summary>
120:     private TreeNode lastSelectedNode = null;
121:
122:     /// <summary>
123:     /// Strings used when adding dynamic values to the message format strings
124:     /// </summary>
125:     private const string MESSAGE_FORMAT_SOURCE = "$source ";
126:     private const string MESSAGE_FORMAT_SEVERITY = "$severity ";
127:     private const string MESSAGE_FORMAT_MESSAGE = "$message ";
128:     private const string MESSAGE_FORMAT_CONDITION = "$condition ";
129:     private const string MESSAGE_FORMAT_SERVER = "$server ";
130:     private const string MESSAGE_FORMAT_SUBSCRIPTION = "$subscription ";
131:     private const string MESSAGE_FORMAT_VALUE = "$value ";
132:     private const string MESSAGE_FORMAT_ITEM_NAME = "$itemname ";
133:     private const string MESSAGE_FORMAT_ITEM_PATH = "$itempath ";
134:     private const string MESSAGE_FORMAT_RESULT = "$result ";
135:     private const string MESSAGE_FORMAT_FORMULA = "$formula() ";
136:
137:     /// <summary>
138:     /// Language dependent strings giving information about the Service
139:     /// </summary>
140:     private string SERVICE_FETCHING_INFO;
141:     private string SERVICE_RUNNING;
```

```
142:     private string SERVICE_STOPPED;
143:     private string SERVICE_NOT_INSTALLED;
144:
145:     private string ERROR_TITLE;
146:     private string ERROR_MESSAGE_OPEN_FILE;
147:     private string ERROR_MESSAGE_SAVE_FILE;
148:
149:     #region Designer Generated Variables
150:
151:     private System.Windows.Forms.MainMenu mainMenu1;
152:     private System.Windows.Forms.MenuItem menuItem8;
153:     private System.Windows.Forms.MenuItem menuItem9;
154:     private System.Windows.Forms.StatusBar statusBar;
155:     private System.Windows.Forms.StatusBarPanel statusBarPanelLeft;
156:     private System.Windows.Forms.StatusBarPanel statusBarPanelRight;
157:     private System.Windows.Forms.TabPage tabService;
158:     private System.Windows.Forms.GroupBox groupBoxServer;
159:     private System.Windows.Forms.TextBox textBoxServerDefaultAreaGroups;
160:     private System.Windows.Forms.Label labelServerDefaultAreaGroups;
161:     private System.Windows.Forms.TextBox textBoxServerDefaultUnitId;
162:     private System.Windows.Forms.TextBox textBoxServerOpcServerName;
163:     private System.Windows.Forms.Label labelServerDefaultUnitId;
164:     private System.Windows.Forms.Label labelServerOpcServerHost;
165:     private System.Windows.Forms.Label labelServerOpcServerName;
166:     private System.Windows.Forms.GroupBox groupBoxServerAcsAlarmCodes;
167:     private System.Windows.Forms.ListBox listBoxServerAcsAlarmCodes;
168:     private System.Windows.Forms.Button buttonServerAcsAlarmCodesAddAlarmCode;
169:     private System.Windows.Forms.TextBox textBoxServerAcsAlarmCodesAlarmCode;
170:     private System.Windows.Forms.Label labelServerAcsAlarmCodesAlarmCode;
171:     private System.Windows.Forms.GroupBox groupBoxSubscription;
172:     private System.Windows.Forms.Panel panelDaSubscription;
173:     private System.Windows.Forms.Label labelDaSubscriptionAcsAlarmCode;
174:     private System.Windows.Forms.ComboBox comboBoxDaSubscriptionAcsAlarmCode;
175:     private System.Windows.Forms.Button buttonDaSubscriptionItemSelect;
176:     private System.Windows.Forms.TextBox textBoxDaSubscriptionItem;
177:     private System.Windows.Forms.Label labelDaSubscriptionItem;
178:     private System.Windows.Forms.Button buttonDaSubscriptionSubscription;
179:     private System.Windows.Forms.Button buttonDaSubscriptionServer;
180:     private System.Windows.Forms.Button buttonDaSubscriptionResult;
181:     private System.Windows.Forms.Button buttonDaSubscriptionItemPath;
182:     private System.Windows.Forms.Button buttonDaSubscriptionItemName;
183:     private System.Windows.Forms.Button buttonDaSubscriptionValue;
184:     private System.Windows.Forms.TextBox textBoxDaSubscriptionMessageFormat;
185:     private System.Windows.Forms.Label labelDaSubscriptionMessageFormat;
186:     private System.Windows.Forms.TextBox textBoxSubscriptionAcsAreaGroups;
187:     private System.Windows.Forms.TextBox textBoxSubscriptionAcsUnitId;
188:     private System.Windows.Forms.TextBox textBoxSubscriptionName;
```

```
189:     private System.Windows.Forms.Label labelSubscriptionName;
190:     private System.Windows.Forms.Panel panelAeSubscription;
191:     private System.Windows.Forms.Button buttonAeSubscriptionSubscription;
192:     private System.Windows.Forms.Button buttonAeSubscriptionServer;
193:     private System.Windows.Forms.Button buttonAeSubscriptionCondition;
194:     private System.Windows.Forms.Button buttonAeSubscriptionMessage;
195:     private System.Windows.Forms.Button buttonAeSubscriptionSeverity;
196:     private System.Windows.Forms.Button buttonAeSubscriptionSource;
197:     private System.Windows.Forms.TextBox textBoxAeSubscriptionMessageFormat;
198:     private System.Windows.Forms.GroupBox groupBoxSeverityMappings;
199:     private System.Windows.Forms.ListBox listBoxAeSubscriptionSeverityMappings;
200:     private System.Windows.Forms.Button buttonAeSubscriptionSeverityMappingAddMapping;
201:     private System.Windows.Forms.ComboBox comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode;
202:     private System.Windows.Forms.TextBox textBoxAeSubscriptionSeverityMappingsRange;
203:     private System.Windows.Forms.GroupBox groupBoxFilter;
204:     private System.Windows.Forms.Button buttonAeSubscriptionFilterEventCatIdsSelect;
205:     private System.Windows.Forms.Button buttonAeSubscriptionFilterAreasSelect;
206:     private System.Windows.Forms.Button buttonAeSubscriptionFilterSourcesSelect;
207:     private System.Windows.Forms.ComboBox comboBoxAeSubscriptionFilterEventType;
208:     private System.Windows.Forms.TextBox textBoxAeSubscriptionFilterHighestSeverity;
209:     private System.Windows.Forms.TextBox textBoxAeSubscriptionFilterLowestSeverity;
210:     private System.Windows.Forms.TextBox textBoxAeSubscriptionFilterEventCatIds;
211:     private System.Windows.Forms.TextBox textBoxAeSubscriptionFilterAreas;
212:     private System.Windows.Forms.TextBox textBoxAeSubscriptionFilterSources;
213:     private System.Windows.Forms.Button buttonAddDaSubscription;
214:     private System.Windows.Forms.Button buttonAddAeSubscription;
215:     private System.Windows.Forms.Button buttonAddServer;
216:     private System.Windows.Forms.Button buttonRemoveSelected;
217:     private System.Windows.Forms.GroupBox groupBoxServerList;
218:     private System.Windows.Forms.TabPage tabConfiguration;
219:     private System.Windows.Forms.GroupBox groupBoxInformation;
220:     private System.Windows.Forms.Label labelInformation;
221:     private System.Windows.Forms.TreeView treeViewServers;
222:     private System.Windows.Forms.PictureBox pictureBoxConfigurationLogo;
223:     private System.Windows.Forms.PictureBox pictureBoxServiceLogo;
224:     private System.Windows.Forms.ImageList imageListTreeView;
225:     private System.Windows.Forms.TextBox textBoxServerOpcServerHost;
226:     private System.Windows.Forms.MenuItem menuItemFile;
227:     private System.Windows.Forms.MenuItem menuItemNew;
228:     private System.Windows.Forms.MenuItem menuItemOpen;
229:     private System.Windows.Forms.MenuItem menuItemSave;
230:     private System.Windows.Forms.MenuItem menuItemSaveAs;
231:     private System.Windows.Forms.MenuItem menuItemExit;
232:     private System.Windows.Forms.MenuItem menuItemHelp;
233:     private System.Windows.Forms.MenuItem menuItemAbout;
234:     private System.Windows.Forms.ImageList imageListMenu;
235:     private Chris.Beckett.MenuImageLib.MenuImage menuImage;
```

```
236:     private System.Windows.Forms.GroupBox groupBoxServiceInformation;
237:     private System.Windows.Forms.Label labelServiceStatus;
238:     private System.Windows.Forms.ProgressBar progressBarService;
239:     private System.Windows.Forms.Button buttonServiceStopService;
240:     private System.Windows.Forms.Label labelServiceDisplayName;
241:     private System.Windows.Forms.Label labelServiceServiceName;
242:     private System.Windows.Forms.Label labelServiceServiceType;
243:     private System.Windows.Forms.Label labelServiceServiceStatus;
244:     private System.Windows.Forms.Label labelServiceDisplayNameData;
245:     private System.Windows.Forms.Label labelServiceServiceNameData;
246:     private System.Windows.Forms.Label labelServiceServiceTypeData;
247:     private System.Windows.Forms.Label labelServiceServiceStatusData;
248:     private System.Windows.Forms.TabControl tabControl;
249:     private System.Windows.Forms.Button buttonServiceStartService;
250:     private System.Windows.Forms.OpenFileDialog openFileDialog;
251:     private System.Windows.Forms.SaveFileDialog saveFileDialog;
252:     private System.Windows.Forms.Label labelServerConnectionType;
253:     private System.Windows.Forms.RadioButton radioButtonServerConnectionTypeCom;
254:     private System.Windows.Forms.RadioButton radioButtonServerConnectionTypeXml;
255:     private System.Windows.Forms.Button buttonServerServerNameSelect;
256:     private System.Windows.Forms.RadioButton radioButtonAeSubscriptionFilterEnabledOff;
257:     private System.Windows.Forms.RadioButton radioButtonAeSubscriptionFilterEnabledOn;
258:     private System.Windows.Forms.Label labelSubscriptionEnabled;
259:     private System.Windows.Forms.RadioButton radioButtonSubscriptionEnabledOn;
260:     private System.Windows.Forms.RadioButton radioButtonSubscriptionEnabledOff;
261:     private System.Windows.Forms.Label labelAeSubscriptionFilterHighestSeverityInfo;
262:     private System.Windows.Forms.Label labelAeSubscriptionFilterLowestSeverityInfo;
263:     private System.Windows.Forms.Label labelDaSubscriptionAlarmState;
264:     private System.Windows.Forms.RadioButton radioButtonDaSubscriptionAlarmStateTypeBoolean;
265:     private System.Windows.Forms.RadioButton radioButtonDaSubscriptionAlarmStateTypeSingleValue;
266:     private System.Windows.Forms.RadioButton radioButtonDaSubscriptionAlarmStateTypeRange;
267:     private System.Windows.Forms.ComboBox comboBoxDaSubscriptionAlarmStateBooleanValue;
268:     private System.Windows.Forms.TextBox textBoxDaSubscriptionAlarmStateSingleValue;
269:     private System.Windows.Forms.TextBox textBoxDaSubscriptionAlarmStateRangeHigh;
270:     private System.Windows.Forms.Label labelDaSubscriptionAlarmStateRangeDash;
271:     private System.Windows.Forms.TextBox textBoxDaSubscriptionAlarmStateRangeLow;
272:     private System.Windows.Forms.MenuItem menuItemLanguage;
273:     private System.Windows.Forms.MenuItem menuItemEnglish;
274:     private System.Windows.Forms.MenuItem menuItemDanish;
275:     private System.Windows.Forms.Label labelSubscriptionAcsAreaGroups;
276:     private System.Windows.Forms.Label labelSubscriptionAcsUnitId;
277:     private System.Windows.Forms.Label labelAeSubscriptionMessageFormat;
278:     private System.Windows.Forms.Label labelAeSubscriptionFilterHighestSeverity;
279:     private System.Windows.Forms.Label labelAeSubscriptionFilterLowestSeverity;
280:     private System.Windows.Forms.Label labelAeSubscriptionFilterEventCatIds;
281:     private System.Windows.Forms.Label labelAeSubscriptionFilterEventType;
282:     private System.Windows.Forms.Label labelAeSubscriptionFilterAreas;
```

```
283:     private System.Windows.Forms.Label labelAeSubscriptionFilterSources;
284:     private System.Windows.Forms.Label labelAeSubscriptionFilterEnabled;
285:     private System.Windows.Forms.Label labelAeSubscriptionSeverityMappingsAcsAlarmCode;
286:     private System.Windows.Forms.Label labelAeSubscriptionSeverityMappingsRange;
287:     private System.Windows.Forms.Button buttonAeSubscriptionFormula;
288:     private System.Windows.Forms.Button buttonDaSubscriptionFormula;
289:     private System.ComponentModel.IContainer components;
290:
291:     #endregion
292:
293:     #region Constructor/Desctructor
294:
295:     public View()
296:     {
297:       HOME_DIRECTORY = System.AppDomain.CurrentDomain.BaseDirectory;
298:       controller = new Controller(this);
299:       //
300:       // Required for Windows Form Designer support
301:       //
302:       InitializeComponent();
303:
304:       //
305:       // TODO: Add any constructor code after InitializeComponent call
306:       //
307:     }
308:
309:     /// <summary>
310:     /// Clean up any resources being used.
311:     /// </summary>
312:     protected override void Dispose( bool disposing )
313:     {
314:       if( disposing )
315:       {
316:         if (components != null)
317:         {
318:           components.Dispose();
319:         }
320:       }
321:       base.Dispose( disposing );
322:     }
323:
324:     #endregion
325:
326:     #region Windows Form Designer generated code
327:     /// <summary>
328:     /// Required method for Designer support - do not modify
329:     /// the contents of this method with the code editor.
```

```
330:     /// </summary>
331:     private void InitializeComponent()
332:     {
333:       this.components = new System.ComponentModel.Container();
334:       System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(View));
335:       this.mainMenu1 = new System.Windows.Forms.MainMenu();
336:       this.menuItemFile = new System.Windows.Forms.MenuItem();
337:       this.menuItemNew = new System.Windows.Forms.MenuItem();
338:       this.menuItemOpen = new System.Windows.Forms.MenuItem();
339:       this.menuItem8 = new System.Windows.Forms.MenuItem();
340:       this.menuItemSave = new System.Windows.Forms.MenuItem();
341:       this.menuItemSaveAs = new System.Windows.Forms.MenuItem();
342:       this.menuItem9 = new System.Windows.Forms.MenuItem();
343:       this.menuItemExit = new System.Windows.Forms.MenuItem();
344:       this.menuItemLanguage = new System.Windows.Forms.MenuItem();
345:       this.menuItemEnglish = new System.Windows.Forms.MenuItem();
346:       this.menuItemDanish = new System.Windows.Forms.MenuItem();
347:       this.menuItemHelp = new System.Windows.Forms.MenuItem();
348:       this.menuItemAbout = new System.Windows.Forms.MenuItem();
349:       this.statusBar = new System.Windows.Forms.StatusBar();
350:       this.statusBarPanelLeft = new System.Windows.Forms.StatusBarPanel();
351:       this.statusBarPanelRight = new System.Windows.Forms.StatusBarPanel();
352:       this.tabControl = new System.Windows.Forms.TabControl();
353:       this.tabConfiguration = new System.Windows.Forms.TabPage();
354:       this.groupBoxInformation = new System.Windows.Forms.GroupBox();
355:       this.labelInformation = new System.Windows.Forms.Label();
356:       this.groupBoxServer = new System.Windows.Forms.GroupBox();
357:       this.buttonServerServerNameSelect = new System.Windows.Forms.Button();
358:       this.radioButtonServerConnectionTypeXml = new System.Windows.Forms.RadioButton();
359:       this.radioButtonServerConnectionTypeCom = new System.Windows.Forms.RadioButton();
360:       this.labelServerConnectionType = new System.Windows.Forms.Label();
361:       this.textBoxServerOpcServerHost = new System.Windows.Forms.TextBox();
362:       this.textBoxServerDefaultAreaGroups = new System.Windows.Forms.TextBox();
363:       this.labelServerDefaultAreaGroups = new System.Windows.Forms.Label();
364:       this.textBoxServerDefaultUnitId = new System.Windows.Forms.TextBox();
365:       this.textBoxServerOpcServerName = new System.Windows.Forms.TextBox();
366:       this.labelServerDefaultUnitId = new System.Windows.Forms.Label();
367:       this.labelServerOpcServerHost = new System.Windows.Forms.Label();
368:       this.labelServerOpcServerName = new System.Windows.Forms.Label();
369:       this.groupBoxServerAcsAlarmCodes = new System.Windows.Forms.GroupBox();
370:       this.listBoxServerAcsAlarmCodes = new System.Windows.Forms.ListBox();
371:       this.buttonServerAcsAlarmCodesAddAlarmCode = new System.Windows.Forms.Button();
372:       this.textBoxServerAcsAlarmCodesAlarmCode = new System.Windows.Forms.TextBox();
373:       this.labelServerAcsAlarmCodesAlarmCode = new System.Windows.Forms.Label();
374:       this.groupBoxSubscription = new System.Windows.Forms.GroupBox();
375:       this.radioButtonSubscriptionEnabledOff = new System.Windows.Forms.RadioButton();
376:       this.radioButtonSubscriptionEnabledOn = new System.Windows.Forms.RadioButton();
```

```
377:         this.labelSubscriptionEnabled = new System.Windows.Forms.Label();
378:         this.panelDaSubscription = new System.Windows.Forms.Panel();
379:         this.textBoxDaSubscriptionAlarmStateRangeLow = new System.Windows.Forms.TextBox();
380:         this.labelDaSubscriptionAlarmStateRangeDash = new System.Windows.Forms.Label();
381:         this.textBoxDaSubscriptionAlarmStateRangeHigh = new System.Windows.Forms.TextBox();
382:         this.textBoxDaSubscriptionAlarmStateSingleValue = new System.Windows.Forms.TextBox();
383:         this.comboBoxDaSubscriptionAlarmStateBooleanValue = new System.Windows.Forms.ComboBox();
384:         this.radioButtonDaSubscriptionAlarmStateTypeRange = new System.Windows.Forms.RadioButton();
385:         this.radioButtonDaSubscriptionAlarmStateTypeSingleValue = new System.Windows.Forms.RadioButton();
386:         this.radioButtonDaSubscriptionAlarmStateTypeBoolean = new System.Windows.Forms.RadioButton();
387:         this.comboBoxDaSubscriptionAcsAlarmCode = new System.Windows.Forms.ComboBox();
388:         this.labelDaSubscriptionAlarmState = new System.Windows.Forms.Label();
389:         this.buttonDaSubscriptionItemSelect = new System.Windows.Forms.Button();
390:         this.textBoxDaSubscriptionItem = new System.Windows.Forms.TextBox();
391:         this.labelDaSubscriptionItem = new System.Windows.Forms.Label();
392:         this.buttonDaSubscriptionServer = new System.Windows.Forms.Button();
393:         this.buttonDaSubscriptionResult = new System.Windows.Forms.Button();
394:         this.buttonDaSubscriptionItemPath = new System.Windows.Forms.Button();
395:         this.buttonDaSubscriptionItemName = new System.Windows.Forms.Button();
396:         this.buttonDaSubscriptionValue = new System.Windows.Forms.Button();
397:         this.textBoxDaSubscriptionMessageFormat = new System.Windows.Forms.TextBox();
398:         this.labelDaSubscriptionMessageFormat = new System.Windows.Forms.Label();
399:         this.buttonDaSubscriptionSubscription = new System.Windows.Forms.Button();
400:         this.labelDaSubscriptionAcsAlarmCode = new System.Windows.Forms.Label();
401:         this.textBoxSubscriptionAcsAreaGroups = new System.Windows.Forms.TextBox();
402:         this.textBoxSubscriptionAcsUnitId = new System.Windows.Forms.TextBox();
403:         this.textBoxSubscriptionName = new System.Windows.Forms.TextBox();
404:         this.labelSubscriptionAcsAreaGroups = new System.Windows.Forms.Label();
405:         this.labelSubscriptionAcsUnitId = new System.Windows.Forms.Label();
406:         this.labelSubscriptionName = new System.Windows.Forms.Label();
407:         this.panelAeSubscription = new System.Windows.Forms.Panel();
408:         this.buttonAeSubscriptionSubscription = new System.Windows.Forms.Button();
409:         this.buttonAeSubscriptionServer = new System.Windows.Forms.Button();
410:         this.buttonAeSubscriptionCondition = new System.Windows.Forms.Button();
411:         this.buttonAeSubscriptionMessage = new System.Windows.Forms.Button();
412:         this.buttonAeSubscriptionSeverity = new System.Windows.Forms.Button();
413:         this.buttonAeSubscriptionSource = new System.Windows.Forms.Button();
414:         this.textBoxAeSubscriptionMessageFormat = new System.Windows.Forms.TextBox();
415:         this.groupBoxSeverityMappings = new System.Windows.Forms.GroupBox();
416:         this.listBoxAeSubscriptionSeverityMappings = new System.Windows.Forms.ListBox();
417:         this.buttonAeSubscriptionSeverityMappingAddMapping = new System.Windows.Forms.Button();
418:         this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode = new System.Windows.Forms.ComboBox();
419:         this.textBoxAeSubscriptionSeverityMappingsRange = new System.Windows.Forms.TextBox();
420:         this.labelAeSubscriptionSeverityMappingsAcsAlarmCode = new System.Windows.Forms.Label();
421:         this.labelAeSubscriptionSeverityMappingsRange = new System.Windows.Forms.Label();
422:         this.groupBoxFilter = new System.Windows.Forms.GroupBox();
423:         this.labelAeSubscriptionFilterLowestSeverityInfo = new System.Windows.Forms.Label();
```

```
424:         this.labelAeSubscriptionFilterHighestSeverityInfo = new System.Windows.Forms.Label();
425:         this.buttonAeSubscriptionFilterEventCatIdsSelect = new System.Windows.Forms.Button();
426:         this.buttonAeSubscriptionFilterAreasSelect = new System.Windows.Forms.Button();
427:         this.buttonAeSubscriptionFilterSourcesSelect = new System.Windows.Forms.Button();
428:         this.radioButtonAeSubscriptionFilterEnabledOff = new System.Windows.Forms.RadioButton();
429:         this.radioButtonAeSubscriptionFilterEnabledOn = new System.Windows.Forms.RadioButton();
430:         this.comboBoxAeSubscriptionFilterEventType = new System.Windows.Forms.ComboBox();
431:         this.textBoxAeSubscriptionFilterHighestSeverity = new System.Windows.Forms.TextBox();
432:         this.textBoxAeSubscriptionFilterLowestSeverity = new System.Windows.Forms.TextBox();
433:         this.textBoxAeSubscriptionFilterEventCatIds = new System.Windows.Forms.TextBox();
434:         this.textBoxAeSubscriptionFilterAreas = new System.Windows.Forms.TextBox();
435:         this.labelAeSubscriptionFilterHighestSeverity = new System.Windows.Forms.Label();
436:         this.labelAeSubscriptionFilterLowestSeverity = new System.Windows.Forms.Label();
437:         this.labelAeSubscriptionFilterEventCatIds = new System.Windows.Forms.Label();
438:         this.labelAeSubscriptionFilterEventType = new System.Windows.Forms.Label();
439:         this.labelAeSubscriptionFilterAreas = new System.Windows.Forms.Label();
440:         this.labelAeSubscriptionFilterEnabled = new System.Windows.Forms.Label();
441:         this.textBoxAeSubscriptionFilterSources = new System.Windows.Forms.TextBox();
442:         this.labelAeSubscriptionFilterSources = new System.Windows.Forms.Label();
443:         this.labelAeSubscriptionMessageFormat = new System.Windows.Forms.Label();
444:         this.buttonAddDaSubscription = new System.Windows.Forms.Button();
445:         this.buttonAddAeSubscription = new System.Windows.Forms.Button();
446:         this.buttonAddServer = new System.Windows.Forms.Button();
447:         this.buttonRemoveSelected = new System.Windows.Forms.Button();
448:         this.groupBoxServerList = new System.Windows.Forms.GroupBox();
449:         this.treeViewServers = new System.Windows.Forms.TreeView();
450:         this.imageListTreeView = new System.Windows.Forms.ImageList(this.components);
451:         this.pictureBoxConfigurationLogo = new System.Windows.Forms.PictureBox();
452:         this.tabService = new System.Windows.Forms.TabPage();
453:         this.buttonServiceStopService = new System.Windows.Forms.Button();
454:         this.buttonServiceStartService = new System.Windows.Forms.Button();
455:         this.progressBarService = new System.Windows.Forms.ProgressBar();
456:         this.labelServiceStatus = new System.Windows.Forms.Label();
457:         this.groupBoxServiceInformation = new System.Windows.Forms.GroupBox();
458:         this.labelServiceServiceStatusData = new System.Windows.Forms.Label();
459:         this.labelServiceServiceTypeData = new System.Windows.Forms.Label();
460:         this.labelServiceServiceNameData = new System.Windows.Forms.Label();
461:         this.labelServiceDisplayNameData = new System.Windows.Forms.Label();
462:         this.labelServiceServiceStatus = new System.Windows.Forms.Label();
463:         this.labelServiceServiceType = new System.Windows.Forms.Label();
464:         this.labelServiceServiceName = new System.Windows.Forms.Label();
465:         this.labelServiceDisplayName = new System.Windows.Forms.Label();
466:         this.pictureBoxServiceLogo = new System.Windows.Forms.PictureBox();
467:         this.imageListMenu = new System.Windows.Forms.ImageList(this.components);
468:         this.menuImage = new Chris.Beckett.MenuImageLib.MenuImage(this.components);
469:         this.openFileDialog = new System.Windows.Forms.OpenFileDialog();
470:         this.saveFileDialog = new System.Windows.Forms.SaveFileDialog();
```

```
471:        this.buttonAeSubscriptionFormula = new System.Windows.Forms.Button();
472:        this.buttonDaSubscriptionFormula = new System.Windows.Forms.Button();
473:        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanelLeft)).BeginInit();
474:        ((System.ComponentModel.ISupportInitialize)(this.statusBarPanelRight)).BeginInit();
475:        this.tabControl.SuspendLayout();
476:        this.tabConfiguration.SuspendLayout();
477:        this.groupBoxInformation.SuspendLayout();
478:        this.groupBoxServer.SuspendLayout();
479:        this.groupBoxServerAcsAlarmCodes.SuspendLayout();
480:        this.groupBoxSubscription.SuspendLayout();
481:        this.panelDaSubscription.SuspendLayout();
482:        this.panelAeSubscription.SuspendLayout();
483:        this.groupBoxSeverityMappings.SuspendLayout();
484:        this.groupBoxFilter.SuspendLayout();
485:        this.groupBoxServerList.SuspendLayout();
486:        this.tabService.SuspendLayout();
487:        this.groupBoxServiceInformation.SuspendLayout();
488:        this.SuspendLayout();
489:        //
490:        // mainMenu1
491:        //
492:        this.mainMenu1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
493:                                          this.menuItemFile,
494:                                          this.menuItemLanguage,
495:                                          this.menuItemHelp});
496:        //
497:        // menuItemFile
498:        //
499:        this.menuItemFile.Index = 0;
500:        this.menuItemFile.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
501:                                           this.menuItemNew,
502:                                           this.menuItemOpen,
503:                                           this.menuItem8,
504:                                           this.menuItemSave,
505:                                           this.menuItemSaveAs,
506:                                           this.menuItem9,
507:                                           this.menuItemExit});
508:        this.menuItemFile.Text = "&File";
509:        //
510:        // menuItemNew
511:        //
512:        this.menuItemNew.Index = 0;
513:        this.menuImage.SetMenuImage(this.menuItemNew, "0");
514:        this.menuItemNew.OwnerDraw = true;
515:        this.menuItemNew.Shortcut = System.Windows.Forms.Shortcut.CtrlN;
516:        this.menuItemNew.Text = "&New";
517:        this.menuItemNew.Click += new System.EventHandler(this.menuItemNew_Click);
```

```
518:        //
519:        // menuItemOpen
520:        //
521:        this.menuItemOpen.Index = 1;
522:        this.menuImage.SetMenuImage(this.menuItemOpen, "1");
523:        this.menuItemOpen.OwnerDraw = true;
524:        this.menuItemOpen.Shortcut = System.Windows.Forms.Shortcut.CtrlO;
525:        this.menuItemOpen.Text = "&Open...";
526:        this.menuItemOpen.Click += new System.EventHandler(this.menuItemOpen_Click);
527:        //
528:        // menuItem8
529:        //
530:        this.menuItem8.Index = 2;
531:        this.menuImage.SetMenuImage(this.menuItem8, null);
532:        this.menuItem8.OwnerDraw = true;
533:        this.menuItem8.Text = "-";
534:        //
535:        // menuItemSave
536:        //
537:        this.menuItemSave.Enabled = false;
538:        this.menuItemSave.Index = 3;
539:        this.menuImage.SetMenuImage(this.menuItemSave, "2");
540:        this.menuItemSave.OwnerDraw = true;
541:        this.menuItemSave.Shortcut = System.Windows.Forms.Shortcut.CtrlS;
542:        this.menuItemSave.Text = "&Save";
543:        this.menuItemSave.Click += new System.EventHandler(this.menuItemSave_Click);
544:        //
545:        // menuItemSaveAs
546:        //
547:        this.menuItemSaveAs.Index = 4;
548:        this.menuImage.SetMenuImage(this.menuItemSaveAs, null);
549:        this.menuItemSaveAs.OwnerDraw = true;
550:        this.menuItemSaveAs.Text = "S&ave as...";
551:        this.menuItemSaveAs.Click += new System.EventHandler(this.menuItemSaveAs_Click);
552:        //
553:        // menuItem9
554:        //
555:        this.menuItem9.Index = 5;
556:        this.menuImage.SetMenuImage(this.menuItem9, null);
557:        this.menuItem9.OwnerDraw = true;
558:        this.menuItem9.Text = "-";
559:        //
560:        // menuItemExit
561:        //
562:        this.menuItemExit.Index = 6;
563:        this.menuImage.SetMenuImage(this.menuItemExit, null);
564:        this.menuItemExit.OwnerDraw = true;
```

```
565:        this.menuItemExit.Shortcut = System.Windows.Forms.Shortcut.CtrlX;
566:        this.menuItemExit.Text = "E&xit";
567:        this.menuItemExit.Click += new System.EventHandler(this.menuItemExit_Click);
568:        //
569:        // menuItemLanguage
570:        //
571:        this.menuItemLanguage.Index = 1;
572:        this.menuItemLanguage.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
573:                                                 this.menuItemEnglish,
574:                                                 this.menuItemDanish});
575:        this.menuItemLanguage.Text = "&Language";
576:        //
577:        // menuItemEnglish
578:        //
579:        this.menuItemEnglish.Index = 0;
580:        this.menuImage.SetMenuImage(this.menuItemEnglish, null);
581:        this.menuItemEnglish.OwnerDraw = true;
582:        this.menuItemEnglish.Text = "&English";
583:        this.menuItemEnglish.Click += new System.EventHandler(this.menuItemEnglish_Click);
584:        //
585:        // menuItemDanish
586:        //
587:        this.menuItemDanish.Index = 1;
588:        this.menuImage.SetMenuImage(this.menuItemDanish, null);
589:        this.menuItemDanish.OwnerDraw = true;
590:        this.menuItemDanish.Text = "&Dansk";
591:        this.menuItemDanish.Click += new System.EventHandler(this.menuItemDanish_Click);
592:        //
593:        // menuItemHelp
594:        //
595:        this.menuItemHelp.Index = 2;
596:        this.menuItemHelp.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
597:                                             this.menuItemAbout});
598:        this.menuItemHelp.Text = "&Help";
599:        //
600:        // menuItemAbout
601:        //
602:        this.menuItemAbout.Index = 0;
603:        this.menuImage.SetMenuImage(this.menuItemAbout, "3");
604:        this.menuItemAbout.OwnerDraw = true;
605:        this.menuItemAbout.Text = "&About OPC Listener Configuration";
606:        this.menuItemAbout.Click += new System.EventHandler(this.menuItemAbout_Click);
607:        //
608:        // statusBar
609:        //
610:        this.statusBar.Location = new System.Drawing.Point(0, 457);
611:        this.statusBar.Name = "statusBar";
```

```
612:       this.statusBar.Panels.AddRange(new System.Windows.Forms.StatusBarPanel[] {
613:                                           this.statusBarPanelLeft,
614:                                           this.statusBarPanelRight});
615:       this.statusBar.ShowPanels = true;
616:       this.statusBar.Size = new System.Drawing.Size(904, 24);
617:       this.statusBar.SizingGrip = false;
618:       this.statusBar.TabIndex = 0;
619:       this.statusBar.Text = "statusBar1";
620:       //
621:       // statusBarPanelLeft
622:       //
623:       this.statusBarPanelLeft.AutoSize = System.Windows.Forms.StatusBarPanelAutoSize.Spring;
624:       this.statusBarPanelLeft.Text = "statusBarPanelLeft";
625:       this.statusBarPanelLeft.Width = 826;
626:       //
627:       // statusBarPanelRight
628:       //
629:       this.statusBarPanelRight.Alignment = System.Windows.Forms.HorizontalAlignment.Center;
630:       this.statusBarPanelRight.AutoSize = System.Windows.Forms.StatusBarPanelAutoSize.Contents;
631:       this.statusBarPanelRight.Text = "© Zonith A/S";
632:       this.statusBarPanelRight.Width = 78;
633:       //
634:       // tabControl
635:       //
636:       this.tabControl.Controls.Add(this.tabConfiguration);
637:       this.tabControl.Controls.Add(this.tabService);
638:       this.tabControl.Location = new System.Drawing.Point(0, 8);
639:       this.tabControl.Name = "tabControl";
640:       this.tabControl.SelectedIndex = 0;
641:       this.tabControl.Size = new System.Drawing.Size(1536, 1120);
642:       this.tabControl.TabIndex = 1;
643:       this.tabControl.SelectedIndexChanged += new System.EventHandler(this.tabControl_SelectedIndexChanged);
644:       //
645:       // tabConfiguration
646:       //
647:       this.tabConfiguration.Controls.Add(this.groupBoxInformation);
648:       this.tabConfiguration.Controls.Add(this.groupBoxServer);
649:       this.tabConfiguration.Controls.Add(this.groupBoxSubscription);
650:       this.tabConfiguration.Controls.Add(this.buttonAddDaSubscription);
651:       this.tabConfiguration.Controls.Add(this.buttonAddAeSubscription);
652:       this.tabConfiguration.Controls.Add(this.buttonAddServer);
653:       this.tabConfiguration.Controls.Add(this.buttonRemoveSelected);
654:       this.tabConfiguration.Controls.Add(this.groupBoxServerList);
655:       this.tabConfiguration.Controls.Add(this.pictureBoxConfigurationLogo);
656:       this.tabConfiguration.Location = new System.Drawing.Point(4, 22);
657:       this.tabConfiguration.Name = "tabConfiguration";
658:       this.tabConfiguration.Size = new System.Drawing.Size(1528, 1094);
```

```
659:        this.tabConfiguration.TabIndex = 0;
660:        this.tabConfiguration.Text = "Configuration";
661:        //
662:        // groupBoxInformation
663:        //
664:        this.groupBoxInformation.Controls.Add(this.labelInformation);
665:        this.groupBoxInformation.Location = new System.Drawing.Point(648, 680);
666:        this.groupBoxInformation.Name = "groupBoxInformation";
667:        this.groupBoxInformation.Size = new System.Drawing.Size(640, 392);
668:        this.groupBoxInformation.TabIndex = 7;
669:        this.groupBoxInformation.TabStop = false;
670:        this.groupBoxInformation.Text = "Information";
671:        this.groupBoxInformation.Visible = false;
672:        //
673:        // labelInformation
674:        //
675:        this.labelInformation.Location = new System.Drawing.Point(8, 16);
676:        this.labelInformation.Name = "labelInformation";
677:        this.labelInformation.Size = new System.Drawing.Size(624, 368);
678:        this.labelInformation.TabIndex = 0;
679:        this.labelInformation.Text = "This window will contain information about the selected server/subscription";
680:        this.labelInformation.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
681:        //
682:        // groupBoxServer
683:        //
684:        this.groupBoxServer.Controls.Add(this.buttonServerServerNameSelect);
685:        this.groupBoxServer.Controls.Add(this.radioButtonServerConnectionTypeXml);
686:        this.groupBoxServer.Controls.Add(this.radioButtonServerConnectionTypeCom);
687:        this.groupBoxServer.Controls.Add(this.labelServerConnectionType);
688:        this.groupBoxServer.Controls.Add(this.textBoxServerOpcServerHost);
689:        this.groupBoxServer.Controls.Add(this.textBoxServerDefaultAreaGroups);
690:        this.groupBoxServer.Controls.Add(this.labelServerDefaultAreaGroups);
691:        this.groupBoxServer.Controls.Add(this.textBoxServerDefaultUnitId);
692:        this.groupBoxServer.Controls.Add(this.textBoxServerOpcServerName);
693:        this.groupBoxServer.Controls.Add(this.labelServerDefaultUnitId);
694:        this.groupBoxServer.Controls.Add(this.labelServerOpcServerHost);
695:        this.groupBoxServer.Controls.Add(this.labelServerOpcServerName);
696:        this.groupBoxServer.Controls.Add(this.groupBoxServerAcsAlarmCodes);
697:        this.groupBoxServer.Location = new System.Drawing.Point(0, 680);
698:        this.groupBoxServer.Name = "groupBoxServer";
699:        this.groupBoxServer.Size = new System.Drawing.Size(640, 392);
700:        this.groupBoxServer.TabIndex = 7;
701:        this.groupBoxServer.TabStop = false;
702:        this.groupBoxServer.Text = "Server information";
703:        this.groupBoxServer.Visible = false;
704:        //
705:        // buttonServerServerNameSelect
```

```
706:        //
707:        this.buttonServerServerNameSelect.FlatStyle = System.Windows.Forms.FlatStyle.System;
708:        this.buttonServerServerNameSelect.Location = new System.Drawing.Point(576, 17);
709:        this.buttonServerServerNameSelect.Name = "buttonServerServerNameSelect";
710:        this.buttonServerServerNameSelect.Size = new System.Drawing.Size(56, 23);
711:        this.buttonServerServerNameSelect.TabIndex = 12;
712:        this.buttonServerServerNameSelect.Text = "Select...";
713:        this.buttonServerServerNameSelect.Click += new System.EventHandler(this.buttonServerServerNameSelect_Click);
714:        this.buttonServerServerNameSelect.MouseEnter += new System.EventHandler(this.button_MouseEnter);
715:        this.buttonServerServerNameSelect.MouseLeave += new System.EventHandler(this.button_MouseLeave);
716:        //
717:        // radioButtonServerConnectionTypeXml
718:        //
719:        this.radioButtonServerConnectionTypeXml.Location = new System.Drawing.Point(216, 120);
720:        this.radioButtonServerConnectionTypeXml.Name = "radioButtonServerConnectionTypeXml";
721:        this.radioButtonServerConnectionTypeXml.Size = new System.Drawing.Size(48, 16);
722:        this.radioButtonServerConnectionTypeXml.TabIndex = 11;
723:        this.radioButtonServerConnectionTypeXml.Text = "XML";
724:        this.radioButtonServerConnectionTypeXml.CheckedChanged += new System.EventHandler(this.radioButton_CheckedChanged);
725:        //
726:        // radioButtonServerConnectionTypeCom
727:        //
728:        this.radioButtonServerConnectionTypeCom.Location = new System.Drawing.Point(144, 120);
729:        this.radioButtonServerConnectionTypeCom.Name = "radioButtonServerConnectionTypeCom";
730:        this.radioButtonServerConnectionTypeCom.Size = new System.Drawing.Size(56, 16);
731:        this.radioButtonServerConnectionTypeCom.TabIndex = 10;
732:        this.radioButtonServerConnectionTypeCom.Text = "COM";
733:        this.radioButtonServerConnectionTypeCom.CheckedChanged += new System.EventHandler(this.radioButton_CheckedChanged);
734:        //
735:        // labelServerConnectionType
736:        //
737:        this.labelServerConnectionType.Location = new System.Drawing.Point(8, 120);
738:        this.labelServerConnectionType.Name = "labelServerConnectionType";
739:        this.labelServerConnectionType.Size = new System.Drawing.Size(144, 16);
740:        this.labelServerConnectionType.TabIndex = 9;
741:        this.labelServerConnectionType.Text = "Connection Type:";
742:        //
743:        // textBoxServerOpcServerHost
744:        //
745:        this.textBoxServerOpcServerHost.Location = new System.Drawing.Point(144, 44);
746:        this.textBoxServerOpcServerHost.Name = "textBoxServerOpcServerHost";
747:        this.textBoxServerOpcServerHost.Size = new System.Drawing.Size(488, 20);
748:        this.textBoxServerOpcServerHost.TabIndex = 2;
749:        this.textBoxServerOpcServerHost.Text = "";
750:        this.textBoxServerOpcServerHost.TextChanged += new System.EventHandler(this.AnyTextChanged);
751:        //
752:        // textBoxServerDefaultAreaGroups
```

```
753:        //
754:        this.textBoxServerDefaultAreaGroups.Location = new System.Drawing.Point(144, 92);
755:        this.textBoxServerDefaultAreaGroups.Name = "textBoxServerDefaultAreaGroups";
756:        this.textBoxServerDefaultAreaGroups.Size = new System.Drawing.Size(488, 20);
757:        this.textBoxServerDefaultAreaGroups.TabIndex = 4;
758:        this.textBoxServerDefaultAreaGroups.Text = "";
759:        this.textBoxServerDefaultAreaGroups.TextChanged += new System.EventHandler(this.AnyTextChanged);
760:        //
761:        // labelServerDefaultAreaGroups
762:        //
763:        this.labelServerDefaultAreaGroups.Location = new System.Drawing.Point(8, 96);
764:        this.labelServerDefaultAreaGroups.Name = "labelServerDefaultAreaGroups";
765:        this.labelServerDefaultAreaGroups.Size = new System.Drawing.Size(144, 16);
766:        this.labelServerDefaultAreaGroups.TabIndex = 8;
767:        this.labelServerDefaultAreaGroups.Text = "ACS Default Area Groups:";
768:        this.labelServerDefaultAreaGroups.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
769:        //
770:        // textBoxServerDefaultUnitId
771:        //
772:        this.textBoxServerDefaultUnitId.Location = new System.Drawing.Point(144, 68);
773:        this.textBoxServerDefaultUnitId.Name = "textBoxServerDefaultUnitId";
774:        this.textBoxServerDefaultUnitId.Size = new System.Drawing.Size(488, 20);
775:        this.textBoxServerDefaultUnitId.TabIndex = 3;
776:        this.textBoxServerDefaultUnitId.Text = "";
777:        this.textBoxServerDefaultUnitId.TextChanged += new System.EventHandler(this.AnyTextChanged);
778:        //
779:        // textBoxServerOpcServerName
780:        //
781:        this.textBoxServerOpcServerName.Location = new System.Drawing.Point(144, 20);
782:        this.textBoxServerOpcServerName.Name = "textBoxServerOpcServerName";
783:        this.textBoxServerOpcServerName.Size = new System.Drawing.Size(424, 20);
784:        this.textBoxServerOpcServerName.TabIndex = 1;
785:        this.textBoxServerOpcServerName.Text = "";
786:        this.textBoxServerOpcServerName.Validating += new System.ComponentModel.CancelEventHandler(this.
              textBoxServerOpcServerName_Validating);
787:        this.textBoxServerOpcServerName.TextChanged += new System.EventHandler(this.AnyTextChanged);
788:        //
789:        // labelServerDefaultUnitId
790:        //
791:        this.labelServerDefaultUnitId.Location = new System.Drawing.Point(8, 72);
792:        this.labelServerDefaultUnitId.Name = "labelServerDefaultUnitId";
793:        this.labelServerDefaultUnitId.Size = new System.Drawing.Size(144, 16);
794:        this.labelServerDefaultUnitId.TabIndex = 2;
795:        this.labelServerDefaultUnitId.Text = "ACS Default Unit Id:";
796:        this.labelServerDefaultUnitId.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
797:        //
798:        // labelServerOpcServerHost
```

```
799:        //
800:        this.labelServerOpcServerHost.Location = new System.Drawing.Point(8, 48);
801:        this.labelServerOpcServerHost.Name = "labelServerOpcServerHost";
802:        this.labelServerOpcServerHost.Size = new System.Drawing.Size(144, 16);
803:        this.labelServerOpcServerHost.TabIndex = 1;
804:        this.labelServerOpcServerHost.Text = "OPC Server Host:";
805:        this.labelServerOpcServerHost.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
806:        //
807:        // labelServerOpcServerName
808:        //
809:        this.labelServerOpcServerName.Location = new System.Drawing.Point(8, 24);
810:        this.labelServerOpcServerName.Name = "labelServerOpcServerName";
811:        this.labelServerOpcServerName.Size = new System.Drawing.Size(144, 16);
812:        this.labelServerOpcServerName.TabIndex = 0;
813:        this.labelServerOpcServerName.Text = "OPC Server Name:";
814:        this.labelServerOpcServerName.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
815:        //
816:        // groupBoxServerAcsAlarmCodes
817:        //
818:        this.groupBoxServerAcsAlarmCodes.Controls.Add(this.listBoxServerAcsAlarmCodes);
819:        this.groupBoxServerAcsAlarmCodes.Controls.Add(this.buttonServerAcsAlarmCodesAddAlarmCode);
820:        this.groupBoxServerAcsAlarmCodes.Controls.Add(this.textBoxServerAcsAlarmCodesAlarmCode);
821:        this.groupBoxServerAcsAlarmCodes.Controls.Add(this.labelServerAcsAlarmCodesAlarmCode);
822:        this.groupBoxServerAcsAlarmCodes.Location = new System.Drawing.Point(8, 152);
823:        this.groupBoxServerAcsAlarmCodes.Name = "groupBoxServerAcsAlarmCodes";
824:        this.groupBoxServerAcsAlarmCodes.Size = new System.Drawing.Size(232, 232);
825:        this.groupBoxServerAcsAlarmCodes.TabIndex = 5;
826:        this.groupBoxServerAcsAlarmCodes.TabStop = false;
827:        this.groupBoxServerAcsAlarmCodes.Text = "ACS Alarm Codes";
828:        //
829:        // listBoxServerAcsAlarmCodes
830:        //
831:        this.listBoxServerAcsAlarmCodes.Location = new System.Drawing.Point(8, 72);
832:        this.listBoxServerAcsAlarmCodes.Name = "listBoxServerAcsAlarmCodes";
833:        this.listBoxServerAcsAlarmCodes.Size = new System.Drawing.Size(216, 147);
834:        this.listBoxServerAcsAlarmCodes.TabIndex = 7;
835:        this.listBoxServerAcsAlarmCodes.KeyUp += new System.Windows.Forms.KeyEventHandler(this.
              listBoxServerAcsAlarmCodes_KeyUp);
836:        //
837:        // buttonServerAcsAlarmCodesAddAlarmCode
838:        //
839:        this.buttonServerAcsAlarmCodesAddAlarmCode.FlatStyle = System.Windows.Forms.FlatStyle.System;
840:        this.buttonServerAcsAlarmCodesAddAlarmCode.Location = new System.Drawing.Point(8, 48);
841:        this.buttonServerAcsAlarmCodesAddAlarmCode.Name = "buttonServerAcsAlarmCodesAddAlarmCode";
842:        this.buttonServerAcsAlarmCodesAddAlarmCode.Size = new System.Drawing.Size(216, 23);
843:        this.buttonServerAcsAlarmCodesAddAlarmCode.TabIndex = 6;
844:        this.buttonServerAcsAlarmCodesAddAlarmCode.Text = "Add Alarm Code";
```

```
845:        this.buttonServerAcsAlarmCodesAddAlarmCode.Click += new System.EventHandler(this.
              buttonServerAcsAlarmCodesAddAlarmCode_Click);
846:        this.buttonServerAcsAlarmCodesAddAlarmCode.MouseEnter += new System.EventHandler(this.button_MouseEnter);
847:        this.buttonServerAcsAlarmCodesAddAlarmCode.MouseLeave += new System.EventHandler(this.button_MouseLeave);
848:        //
849:        // textBoxServerAcsAlarmCodesAlarmCode
850:        //
851:        this.textBoxServerAcsAlarmCodesAlarmCode.Location = new System.Drawing.Point(104, 20);
852:        this.textBoxServerAcsAlarmCodesAlarmCode.Name = "textBoxServerAcsAlarmCodesAlarmCode";
853:        this.textBoxServerAcsAlarmCodesAlarmCode.RightToLeft = System.Windows.Forms.RightToLeft.Yes;
854:        this.textBoxServerAcsAlarmCodesAlarmCode.Size = new System.Drawing.Size(120, 20);
855:        this.textBoxServerAcsAlarmCodesAlarmCode.TabIndex = 5;
856:        this.textBoxServerAcsAlarmCodesAlarmCode.Text = "";
857:        //
858:        // labelServerAcsAlarmCodesAlarmCode
859:        //
860:        this.labelServerAcsAlarmCodesAlarmCode.Location = new System.Drawing.Point(8, 24);
861:        this.labelServerAcsAlarmCodesAlarmCode.Name = "labelServerAcsAlarmCodesAlarmCode";
862:        this.labelServerAcsAlarmCodesAlarmCode.Size = new System.Drawing.Size(104, 16);
863:        this.labelServerAcsAlarmCodesAlarmCode.TabIndex = 12;
864:        this.labelServerAcsAlarmCodesAlarmCode.Text = "ACS Alarm Code:";
865:        this.labelServerAcsAlarmCodesAlarmCode.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
866:        //
867:        // groupBoxSubscription
868:        //
869:        this.groupBoxSubscription.Controls.Add(this.radioButtonSubscriptionEnabledOff);
870:        this.groupBoxSubscription.Controls.Add(this.radioButtonSubscriptionEnabledOn);
871:        this.groupBoxSubscription.Controls.Add(this.labelSubscriptionEnabled);
872:        this.groupBoxSubscription.Controls.Add(this.panelDaSubscription);
873:        this.groupBoxSubscription.Controls.Add(this.textBoxSubscriptionAcsAreaGroups);
874:        this.groupBoxSubscription.Controls.Add(this.textBoxSubscriptionAcsUnitId);
875:        this.groupBoxSubscription.Controls.Add(this.textBoxSubscriptionName);
876:        this.groupBoxSubscription.Controls.Add(this.labelSubscriptionAcsAreaGroups);
877:        this.groupBoxSubscription.Controls.Add(this.labelSubscriptionAcsUnitId);
878:        this.groupBoxSubscription.Controls.Add(this.labelSubscriptionName);
879:        this.groupBoxSubscription.Controls.Add(this.panelAeSubscription);
880:        this.groupBoxSubscription.Location = new System.Drawing.Point(256, 8);
881:        this.groupBoxSubscription.Name = "groupBoxSubscription";
882:        this.groupBoxSubscription.Size = new System.Drawing.Size(640, 392);
883:        this.groupBoxSubscription.TabIndex = 7;
884:        this.groupBoxSubscription.TabStop = false;
885:        this.groupBoxSubscription.Text = "Subscription information";
886:        this.groupBoxSubscription.Visible = false;
887:        //
888:        // radioButtonSubscriptionEnabledOff
889:        //
890:        this.radioButtonSubscriptionEnabledOff.Location = new System.Drawing.Point(232, 24);
```

```
891:        this.radioButtonSubscriptionEnabledOff.Name = "radioButtonSubscriptionEnabledOff";
892:        this.radioButtonSubscriptionEnabledOff.Size = new System.Drawing.Size(96, 16);
893:        this.radioButtonSubscriptionEnabledOff.TabIndex = 2;
894:        this.radioButtonSubscriptionEnabledOff.Text = "Off";
895:        //
896:        // radioButtonSubscriptionEnabledOn
897:        //
898:        this.radioButtonSubscriptionEnabledOn.Location = new System.Drawing.Point(144, 24);
899:        this.radioButtonSubscriptionEnabledOn.Name = "radioButtonSubscriptionEnabledOn";
900:        this.radioButtonSubscriptionEnabledOn.Size = new System.Drawing.Size(88, 16);
901:        this.radioButtonSubscriptionEnabledOn.TabIndex = 1;
902:        this.radioButtonSubscriptionEnabledOn.Text = "On";
903:        //
904:        // labelSubscriptionEnabled
905:        //
906:        this.labelSubscriptionEnabled.Location = new System.Drawing.Point(8, 24);
907:        this.labelSubscriptionEnabled.Name = "labelSubscriptionEnabled";
908:        this.labelSubscriptionEnabled.Size = new System.Drawing.Size(144, 16);
909:        this.labelSubscriptionEnabled.TabIndex = 9;
910:        this.labelSubscriptionEnabled.Text = "Enabled:";
911:        this.labelSubscriptionEnabled.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
912:        //
913:        // panelDaSubscription
914:        //
915:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionFormula);
916:        this.panelDaSubscription.Controls.Add(this.textBoxDaSubscriptionAlarmStateRangeLow);
917:        this.panelDaSubscription.Controls.Add(this.labelDaSubscriptionAlarmStateRangeDash);
918:        this.panelDaSubscription.Controls.Add(this.textBoxDaSubscriptionAlarmStateRangeHigh);
919:        this.panelDaSubscription.Controls.Add(this.textBoxDaSubscriptionAlarmStateSingleValue);
920:        this.panelDaSubscription.Controls.Add(this.comboBoxDaSubscriptionAlarmStateBooleanValue);
921:        this.panelDaSubscription.Controls.Add(this.radioButtonDaSubscriptionAlarmStateTypeRange);
922:        this.panelDaSubscription.Controls.Add(this.radioButtonDaSubscriptionAlarmStateTypeSingleValue);
923:        this.panelDaSubscription.Controls.Add(this.radioButtonDaSubscriptionAlarmStateTypeBoolean);
924:        this.panelDaSubscription.Controls.Add(this.comboBoxDaSubscriptionAcsAlarmCode);
925:        this.panelDaSubscription.Controls.Add(this.labelDaSubscriptionAlarmState);
926:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionItemSelect);
927:        this.panelDaSubscription.Controls.Add(this.textBoxDaSubscriptionItem);
928:        this.panelDaSubscription.Controls.Add(this.labelDaSubscriptionItem);
929:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionServer);
930:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionResult);
931:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionItemPath);
932:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionItemName);
933:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionValue);
934:        this.panelDaSubscription.Controls.Add(this.textBoxDaSubscriptionMessageFormat);
935:        this.panelDaSubscription.Controls.Add(this.labelDaSubscriptionMessageFormat);
936:        this.panelDaSubscription.Controls.Add(this.buttonDaSubscriptionSubscription);
937:        this.panelDaSubscription.Controls.Add(this.labelDaSubscriptionAcsAlarmCode);
```

```
938:        this.panelDaSubscription.Location = new System.Drawing.Point(8, 392);
939:        this.panelDaSubscription.Name = "panelDaSubscription";
940:        this.panelDaSubscription.Size = new System.Drawing.Size(624, 272);
941:        this.panelDaSubscription.TabIndex = 6;
942:        this.panelDaSubscription.Visible = false;
943:        //
944:        // textBoxDaSubscriptionAlarmStateRangeLow
945:        //
946:        this.textBoxDaSubscriptionAlarmStateRangeLow.Location = new System.Drawing.Point(136, 136);
947:        this.textBoxDaSubscriptionAlarmStateRangeLow.Name = "textBoxDaSubscriptionAlarmStateRangeLow";
948:        this.textBoxDaSubscriptionAlarmStateRangeLow.Size = new System.Drawing.Size(48, 20);
949:        this.textBoxDaSubscriptionAlarmStateRangeLow.TabIndex = 11;
950:        this.textBoxDaSubscriptionAlarmStateRangeLow.Text = "";
951:        this.textBoxDaSubscriptionAlarmStateRangeLow.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
952:        this.textBoxDaSubscriptionAlarmStateRangeLow.Validating += new System.ComponentModel.CancelEventHandler(this.
              textBoxDaSubscriptionAlarmStateRangeLow_Validating);
953:        //
954:        // labelDaSubscriptionAlarmStateRangeDash
955:        //
956:        this.labelDaSubscriptionAlarmStateRangeDash.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing
              .FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
957:        this.labelDaSubscriptionAlarmStateRangeDash.Location = new System.Drawing.Point(184, 136);
958:        this.labelDaSubscriptionAlarmStateRangeDash.Name = "labelDaSubscriptionAlarmStateRangeDash";
959:        this.labelDaSubscriptionAlarmStateRangeDash.Size = new System.Drawing.Size(16, 20);
960:        this.labelDaSubscriptionAlarmStateRangeDash.TabIndex = 33;
961:        this.labelDaSubscriptionAlarmStateRangeDash.Text = "-";
962:        this.labelDaSubscriptionAlarmStateRangeDash.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
963:        //
964:        // textBoxDaSubscriptionAlarmStateRangeHigh
965:        //
966:        this.textBoxDaSubscriptionAlarmStateRangeHigh.Location = new System.Drawing.Point(200, 136);
967:        this.textBoxDaSubscriptionAlarmStateRangeHigh.Name = "textBoxDaSubscriptionAlarmStateRangeHigh";
968:        this.textBoxDaSubscriptionAlarmStateRangeHigh.Size = new System.Drawing.Size(48, 20);
969:        this.textBoxDaSubscriptionAlarmStateRangeHigh.TabIndex = 12;
970:        this.textBoxDaSubscriptionAlarmStateRangeHigh.Text = "";
971:        this.textBoxDaSubscriptionAlarmStateRangeHigh.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
972:        this.textBoxDaSubscriptionAlarmStateRangeHigh.Validating += new System.ComponentModel.CancelEventHandler(this.
              textBoxDaSubscriptionAlarmStateRangeHigh_Validating);
973:        //
974:        // textBoxDaSubscriptionAlarmStateSingleValue
975:        //
976:        this.textBoxDaSubscriptionAlarmStateSingleValue.Location = new System.Drawing.Point(136, 136);
977:        this.textBoxDaSubscriptionAlarmStateSingleValue.Name = "textBoxDaSubscriptionAlarmStateSingleValue";
978:        this.textBoxDaSubscriptionAlarmStateSingleValue.Size = new System.Drawing.Size(48, 20);
979:        this.textBoxDaSubscriptionAlarmStateSingleValue.TabIndex = 11;
980:        this.textBoxDaSubscriptionAlarmStateSingleValue.Text = "";
981:        this.textBoxDaSubscriptionAlarmStateSingleValue.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
```

```
982:        this.textBoxDaSubscriptionAlarmStateSingleValue.Validating += new System.ComponentModel.CancelEventHandler(this.
              textBoxDaSubscriptionAlarmStateSingleValue_Validating);
983:        //
984:        // comboBoxDaSubscriptionAlarmStateBooleanValue
985:        //
986:        this.comboBoxDaSubscriptionAlarmStateBooleanValue.Items.AddRange(new object[] {
987:                                                    "TRUE",
988:                                                    "FALSE"});
989:        this.comboBoxDaSubscriptionAlarmStateBooleanValue.Location = new System.Drawing.Point(136, 136);
990:        this.comboBoxDaSubscriptionAlarmStateBooleanValue.Name = "comboBoxDaSubscriptionAlarmStateBooleanValue";
991:        this.comboBoxDaSubscriptionAlarmStateBooleanValue.Size = new System.Drawing.Size(112, 21);
992:        this.comboBoxDaSubscriptionAlarmStateBooleanValue.TabIndex = 11;
993:        this.comboBoxDaSubscriptionAlarmStateBooleanValue.Text = "Select TRUE/FALSE";
994:        //
995:        // radioButtonDaSubscriptionAlarmStateTypeRange
996:        //
997:        this.radioButtonDaSubscriptionAlarmStateTypeRange.Location = new System.Drawing.Point(312, 112);
998:        this.radioButtonDaSubscriptionAlarmStateTypeRange.Name = "radioButtonDaSubscriptionAlarmStateTypeRange";
999:        this.radioButtonDaSubscriptionAlarmStateTypeRange.Size = new System.Drawing.Size(72, 16);
1000:       this.radioButtonDaSubscriptionAlarmStateTypeRange.TabIndex = 10;
1001:       this.radioButtonDaSubscriptionAlarmStateTypeRange.Text = "Range  :";
1002:       this.radioButtonDaSubscriptionAlarmStateTypeRange.CheckedChanged += new System.EventHandler(this.
              radioButtonDaSubscriptionAlarmStateType_CheckedChanged);
1003:       //
1004:       // radioButtonDaSubscriptionAlarmStateTypeSingleValue
1005:       //
1006:       this.radioButtonDaSubscriptionAlarmStateTypeSingleValue.Location = new System.Drawing.Point(216, 112);
1007:       this.radioButtonDaSubscriptionAlarmStateTypeSingleValue.Name = "radioButtonDaSubscriptionAlarmStateTypeSingleValue";
1008:       this.radioButtonDaSubscriptionAlarmStateTypeSingleValue.Size = new System.Drawing.Size(88, 16);
1009:       this.radioButtonDaSubscriptionAlarmStateTypeSingleValue.TabIndex = 10;
1010:       this.radioButtonDaSubscriptionAlarmStateTypeSingleValue.Text = "Single Value";
1011:       this.radioButtonDaSubscriptionAlarmStateTypeSingleValue.CheckedChanged += new System.EventHandler(this.
              radioButtonDaSubscriptionAlarmStateType_CheckedChanged);
1012:       //
1013:       // radioButtonDaSubscriptionAlarmStateTypeBoolean
1014:       //
1015:       this.radioButtonDaSubscriptionAlarmStateTypeBoolean.Location = new System.Drawing.Point(136, 112);
1016:       this.radioButtonDaSubscriptionAlarmStateTypeBoolean.Name = "radioButtonDaSubscriptionAlarmStateTypeBoolean";
1017:       this.radioButtonDaSubscriptionAlarmStateTypeBoolean.Size = new System.Drawing.Size(64, 16);
1018:       this.radioButtonDaSubscriptionAlarmStateTypeBoolean.TabIndex = 10;
1019:       this.radioButtonDaSubscriptionAlarmStateTypeBoolean.Text = "Boolean";
1020:       this.radioButtonDaSubscriptionAlarmStateTypeBoolean.CheckedChanged += new System.EventHandler(this.
              radioButtonDaSubscriptionAlarmStateType_CheckedChanged);
1021:       //
1022:       // comboBoxDaSubscriptionAcsAlarmCode
1023:       //
1024:       this.comboBoxDaSubscriptionAcsAlarmCode.ItemHeight = 13;
```

```
1025:         this.comboBoxDaSubscriptionAcsAlarmCode.Location = new System.Drawing.Point(136, 163);
1026:         this.comboBoxDaSubscriptionAcsAlarmCode.Name = "comboBoxDaSubscriptionAcsAlarmCode";
1027:         this.comboBoxDaSubscriptionAcsAlarmCode.Size = new System.Drawing.Size(112, 21);
1028:         this.comboBoxDaSubscriptionAcsAlarmCode.TabIndex = 13;
1029:         this.comboBoxDaSubscriptionAcsAlarmCode.Text = "Select ACS Alarm Code";
1030:         this.comboBoxDaSubscriptionAcsAlarmCode.SelectionChangeCommitted += new System.EventHandler(this.
              comboBox_SelectionChangeCommitted);
1031:         //
1032:         // labelDaSubscriptionAlarmState
1033:         //
1034:         this.labelDaSubscriptionAlarmState.Location = new System.Drawing.Point(0, 112);
1035:         this.labelDaSubscriptionAlarmState.Name = "labelDaSubscriptionAlarmState";
1036:         this.labelDaSubscriptionAlarmState.Size = new System.Drawing.Size(144, 16);
1037:         this.labelDaSubscriptionAlarmState.TabIndex = 23;
1038:         this.labelDaSubscriptionAlarmState.Text = "Alarm State:";
1039:         this.labelDaSubscriptionAlarmState.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1040:         //
1041:         // buttonDaSubscriptionItemSelect
1042:         //
1043:         this.buttonDaSubscriptionItemSelect.FlatStyle = System.Windows.Forms.FlatStyle.System;
1044:         this.buttonDaSubscriptionItemSelect.Location = new System.Drawing.Point(376, 81);
1045:         this.buttonDaSubscriptionItemSelect.Name = "buttonDaSubscriptionItemSelect";
1046:         this.buttonDaSubscriptionItemSelect.Size = new System.Drawing.Size(56, 23);
1047:         this.buttonDaSubscriptionItemSelect.TabIndex = 9;
1048:         this.buttonDaSubscriptionItemSelect.Text = "Select...";
1049:         this.buttonDaSubscriptionItemSelect.Click += new System.EventHandler(this.buttonDaSubscriptionItemSelect_Click);
1050:         this.buttonDaSubscriptionItemSelect.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1051:         this.buttonDaSubscriptionItemSelect.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1052:         //
1053:         // textBoxDaSubscriptionItem
1054:         //
1055:         this.textBoxDaSubscriptionItem.Location = new System.Drawing.Point(136, 84);
1056:         this.textBoxDaSubscriptionItem.Name = "textBoxDaSubscriptionItem";
1057:         this.textBoxDaSubscriptionItem.Size = new System.Drawing.Size(232, 20);
1058:         this.textBoxDaSubscriptionItem.TabIndex = 8;
1059:         this.textBoxDaSubscriptionItem.Text = "";
1060:         this.textBoxDaSubscriptionItem.TextChanged += new System.EventHandler(this.AnyTextChanged);
1061:         //
1062:         // labelDaSubscriptionItem
1063:         //
1064:         this.labelDaSubscriptionItem.Location = new System.Drawing.Point(0, 88);
1065:         this.labelDaSubscriptionItem.Name = "labelDaSubscriptionItem";
1066:         this.labelDaSubscriptionItem.Size = new System.Drawing.Size(144, 16);
1067:         this.labelDaSubscriptionItem.TabIndex = 14;
1068:         this.labelDaSubscriptionItem.Text = "Item:";
1069:         this.labelDaSubscriptionItem.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1070:         //
```

```
1071:       // buttonDaSubscriptionServer
1072:       //
1073:       this.buttonDaSubscriptionServer.FlatStyle = System.Windows.Forms.FlatStyle.System;
1074:       this.buttonDaSubscriptionServer.Location = new System.Drawing.Point(216, 48);
1075:       this.buttonDaSubscriptionServer.Name = "buttonDaSubscriptionServer";
1076:       this.buttonDaSubscriptionServer.Size = new System.Drawing.Size(80, 23);
1077:       this.buttonDaSubscriptionServer.TabIndex = 6;
1078:       this.buttonDaSubscriptionServer.Text = "server";
1079:       this.buttonDaSubscriptionServer.Click += new System.EventHandler(this.buttonDaSubscriptionMessageFormatButton_Click);
1080:       this.buttonDaSubscriptionServer.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1081:       this.buttonDaSubscriptionServer.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1082:       //
1083:       // buttonDaSubscriptionResult
1084:       //
1085:       this.buttonDaSubscriptionResult.FlatStyle = System.Windows.Forms.FlatStyle.System;
1086:       this.buttonDaSubscriptionResult.Location = new System.Drawing.Point(136, 48);
1087:       this.buttonDaSubscriptionResult.Name = "buttonDaSubscriptionResult";
1088:       this.buttonDaSubscriptionResult.Size = new System.Drawing.Size(80, 23);
1089:       this.buttonDaSubscriptionResult.TabIndex = 5;
1090:       this.buttonDaSubscriptionResult.Text = "result";
1091:       this.buttonDaSubscriptionResult.Click += new System.EventHandler(this.buttonDaSubscriptionMessageFormatButton_Click);
1092:       this.buttonDaSubscriptionResult.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1093:       this.buttonDaSubscriptionResult.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1094:       //
1095:       // buttonDaSubscriptionItemPath
1096:       //
1097:       this.buttonDaSubscriptionItemPath.FlatStyle = System.Windows.Forms.FlatStyle.System;
1098:       this.buttonDaSubscriptionItemPath.Location = new System.Drawing.Point(296, 24);
1099:       this.buttonDaSubscriptionItemPath.Name = "buttonDaSubscriptionItemPath";
1100:       this.buttonDaSubscriptionItemPath.Size = new System.Drawing.Size(80, 23);
1101:       this.buttonDaSubscriptionItemPath.TabIndex = 4;
1102:       this.buttonDaSubscriptionItemPath.Text = "item path";
1103:       this.buttonDaSubscriptionItemPath.Click += new System.EventHandler(this.buttonDaSubscriptionMessageFormatButton_Click)
            ;
1104:       this.buttonDaSubscriptionItemPath.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1105:       this.buttonDaSubscriptionItemPath.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1106:       //
1107:       // buttonDaSubscriptionItemName
1108:       //
1109:       this.buttonDaSubscriptionItemName.FlatStyle = System.Windows.Forms.FlatStyle.System;
1110:       this.buttonDaSubscriptionItemName.Location = new System.Drawing.Point(216, 24);
1111:       this.buttonDaSubscriptionItemName.Name = "buttonDaSubscriptionItemName";
1112:       this.buttonDaSubscriptionItemName.Size = new System.Drawing.Size(80, 23);
1113:       this.buttonDaSubscriptionItemName.TabIndex = 3;
1114:       this.buttonDaSubscriptionItemName.Text = "item name";
1115:       this.buttonDaSubscriptionItemName.Click += new System.EventHandler(this.buttonDaSubscriptionMessageFormatButton_Click)
            ;
```

```
1116:       this.buttonDaSubscriptionItemName.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1117:       this.buttonDaSubscriptionItemName.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1118:       //
1119:       // buttonDaSubscriptionValue
1120:       //
1121:       this.buttonDaSubscriptionValue.FlatStyle = System.Windows.Forms.FlatStyle.System;
1122:       this.buttonDaSubscriptionValue.Location = new System.Drawing.Point(136, 24);
1123:       this.buttonDaSubscriptionValue.Name = "buttonDaSubscriptionValue";
1124:       this.buttonDaSubscriptionValue.Size = new System.Drawing.Size(80, 23);
1125:       this.buttonDaSubscriptionValue.TabIndex = 2;
1126:       this.buttonDaSubscriptionValue.Text = "value";
1127:       this.buttonDaSubscriptionValue.Click += new System.EventHandler(this.buttonDaSubscriptionMessageFormatButton_Click);
1128:       this.buttonDaSubscriptionValue.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1129:       this.buttonDaSubscriptionValue.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1130:       //
1131:       // textBoxDaSubscriptionMessageFormat
1132:       //
1133:       this.textBoxDaSubscriptionMessageFormat.Location = new System.Drawing.Point(136, 4);
1134:       this.textBoxDaSubscriptionMessageFormat.Name = "textBoxDaSubscriptionMessageFormat";
1135:       this.textBoxDaSubscriptionMessageFormat.Size = new System.Drawing.Size(488, 20);
1136:       this.textBoxDaSubscriptionMessageFormat.TabIndex = 1;
1137:       this.textBoxDaSubscriptionMessageFormat.Text = "";
1138:       this.textBoxDaSubscriptionMessageFormat.TextChanged += new System.EventHandler(this.AnyTextChanged);
1139:       //
1140:       // labelDaSubscriptionMessageFormat
1141:       //
1142:       this.labelDaSubscriptionMessageFormat.Location = new System.Drawing.Point(0, 8);
1143:       this.labelDaSubscriptionMessageFormat.Name = "labelDaSubscriptionMessageFormat";
1144:       this.labelDaSubscriptionMessageFormat.Size = new System.Drawing.Size(144, 16);
1145:       this.labelDaSubscriptionMessageFormat.TabIndex = 3;
1146:       this.labelDaSubscriptionMessageFormat.Text = "Message Format:";
1147:       this.labelDaSubscriptionMessageFormat.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1148:       //
1149:       // buttonDaSubscriptionSubscription
1150:       //
1151:       this.buttonDaSubscriptionSubscription.FlatStyle = System.Windows.Forms.FlatStyle.System;
1152:       this.buttonDaSubscriptionSubscription.Location = new System.Drawing.Point(296, 48);
1153:       this.buttonDaSubscriptionSubscription.Name = "buttonDaSubscriptionSubscription";
1154:       this.buttonDaSubscriptionSubscription.Size = new System.Drawing.Size(80, 23);
1155:       this.buttonDaSubscriptionSubscription.TabIndex = 7;
1156:       this.buttonDaSubscriptionSubscription.Text = "subscription";
1157:       this.buttonDaSubscriptionSubscription.Click += new System.EventHandler(this.
            buttonDaSubscriptionMessageFormatButton_Click);
1158:       this.buttonDaSubscriptionSubscription.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1159:       this.buttonDaSubscriptionSubscription.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1160:       //
1161:       // labelDaSubscriptionAcsAlarmCode
```

```
1162:         //
1163:         this.labelDaSubscriptionAcsAlarmCode.Location = new System.Drawing.Point(0, 168);
1164:         this.labelDaSubscriptionAcsAlarmCode.Name = "labelDaSubscriptionAcsAlarmCode";
1165:         this.labelDaSubscriptionAcsAlarmCode.Size = new System.Drawing.Size(144, 16);
1166:         this.labelDaSubscriptionAcsAlarmCode.TabIndex = 26;
1167:         this.labelDaSubscriptionAcsAlarmCode.Text = "ACS Alarm Code:";
1168:         this.labelDaSubscriptionAcsAlarmCode.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1169:         //
1170:         // textBoxSubscriptionAcsAreaGroups
1171:         //
1172:         this.textBoxSubscriptionAcsAreaGroups.Location = new System.Drawing.Point(144, 92);
1173:         this.textBoxSubscriptionAcsAreaGroups.Name = "textBoxSubscriptionAcsAreaGroups";
1174:         this.textBoxSubscriptionAcsAreaGroups.Size = new System.Drawing.Size(488, 20);
1175:         this.textBoxSubscriptionAcsAreaGroups.TabIndex = 5;
1176:         this.textBoxSubscriptionAcsAreaGroups.Text = "";
1177:         this.textBoxSubscriptionAcsAreaGroups.TextChanged += new System.EventHandler(this.AnyTextChanged);
1178:         //
1179:         // textBoxSubscriptionAcsUnitId
1180:         //
1181:         this.textBoxSubscriptionAcsUnitId.Location = new System.Drawing.Point(144, 68);
1182:         this.textBoxSubscriptionAcsUnitId.Name = "textBoxSubscriptionAcsUnitId";
1183:         this.textBoxSubscriptionAcsUnitId.Size = new System.Drawing.Size(488, 20);
1184:         this.textBoxSubscriptionAcsUnitId.TabIndex = 4;
1185:         this.textBoxSubscriptionAcsUnitId.Text = "";
1186:         this.textBoxSubscriptionAcsUnitId.TextChanged += new System.EventHandler(this.AnyTextChanged);
1187:         //
1188:         // textBoxSubscriptionName
1189:         //
1190:         this.textBoxSubscriptionName.Location = new System.Drawing.Point(144, 44);
1191:         this.textBoxSubscriptionName.Name = "textBoxSubscriptionName";
1192:         this.textBoxSubscriptionName.Size = new System.Drawing.Size(488, 20);
1193:         this.textBoxSubscriptionName.TabIndex = 3;
1194:         this.textBoxSubscriptionName.Text = "";
1195:         this.textBoxSubscriptionName.Validating += new System.ComponentModel.CancelEventHandler(this.
              textBoxSubscriptionName_Validating);
1196:         this.textBoxSubscriptionName.TextChanged += new System.EventHandler(this.AnyTextChanged);
1197:         //
1198:         // labelSubscriptionAcsAreaGroups
1199:         //
1200:         this.labelSubscriptionAcsAreaGroups.Location = new System.Drawing.Point(8, 96);
1201:         this.labelSubscriptionAcsAreaGroups.Name = "labelSubscriptionAcsAreaGroups";
1202:         this.labelSubscriptionAcsAreaGroups.Size = new System.Drawing.Size(144, 16);
1203:         this.labelSubscriptionAcsAreaGroups.TabIndex = 2;
1204:         this.labelSubscriptionAcsAreaGroups.Text = "ACS Area Groups:";
1205:         this.labelSubscriptionAcsAreaGroups.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1206:         //
1207:         // labelSubscriptionAcsUnitId
```

```
1208:       //
1209:       this.labelSubscriptionAcsUnitId.Location = new System.Drawing.Point(8, 72);
1210:       this.labelSubscriptionAcsUnitId.Name = "labelSubscriptionAcsUnitId";
1211:       this.labelSubscriptionAcsUnitId.Size = new System.Drawing.Size(144, 16);
1212:       this.labelSubscriptionAcsUnitId.TabIndex = 1;
1213:       this.labelSubscriptionAcsUnitId.Text = "ACS Unit Id:";
1214:       this.labelSubscriptionAcsUnitId.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1215:       //
1216:       // labelSubscriptionName
1217:       //
1218:       this.labelSubscriptionName.Location = new System.Drawing.Point(8, 48);
1219:       this.labelSubscriptionName.Name = "labelSubscriptionName";
1220:       this.labelSubscriptionName.Size = new System.Drawing.Size(144, 16);
1221:       this.labelSubscriptionName.TabIndex = 0;
1222:       this.labelSubscriptionName.Text = "Subscription Name:";
1223:       this.labelSubscriptionName.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1224:       //
1225:       // panelAeSubscription
1226:       //
1227:       this.panelAeSubscription.Controls.Add(this.buttonAeSubscriptionFormula);
1228:       this.panelAeSubscription.Controls.Add(this.buttonAeSubscriptionSubscription);
1229:       this.panelAeSubscription.Controls.Add(this.buttonAeSubscriptionServer);
1230:       this.panelAeSubscription.Controls.Add(this.buttonAeSubscriptionCondition);
1231:       this.panelAeSubscription.Controls.Add(this.buttonAeSubscriptionMessage);
1232:       this.panelAeSubscription.Controls.Add(this.buttonAeSubscriptionSeverity);
1233:       this.panelAeSubscription.Controls.Add(this.buttonAeSubscriptionSource);
1234:       this.panelAeSubscription.Controls.Add(this.textBoxAeSubscriptionMessageFormat);
1235:       this.panelAeSubscription.Controls.Add(this.groupBoxSeverityMappings);
1236:       this.panelAeSubscription.Controls.Add(this.groupBoxFilter);
1237:       this.panelAeSubscription.Controls.Add(this.labelAeSubscriptionMessageFormat);
1238:       this.panelAeSubscription.Location = new System.Drawing.Point(8, 112);
1239:       this.panelAeSubscription.Name = "panelAeSubscription";
1240:       this.panelAeSubscription.Size = new System.Drawing.Size(624, 272);
1241:       this.panelAeSubscription.TabIndex = 6;
1242:       this.panelAeSubscription.Visible = false;
1243:       //
1244:       // buttonAeSubscriptionSubscription
1245:       //
1246:       this.buttonAeSubscriptionSubscription.FlatStyle = System.Windows.Forms.FlatStyle.System;
1247:       this.buttonAeSubscriptionSubscription.Location = new System.Drawing.Point(296, 48);
1248:       this.buttonAeSubscriptionSubscription.Name = "buttonAeSubscriptionSubscription";
1249:       this.buttonAeSubscriptionSubscription.Size = new System.Drawing.Size(80, 23);
1250:       this.buttonAeSubscriptionSubscription.TabIndex = 7;
1251:       this.buttonAeSubscriptionSubscription.Text = "subscription";
1252:       this.buttonAeSubscriptionSubscription.Click += new System.EventHandler(this.
            buttonAeSubscriptionMessageFormatButton_Click);
1253:       this.buttonAeSubscriptionSubscription.MouseEnter += new System.EventHandler(this.button_MouseEnter);
```

```
1254:         this.buttonAeSubscriptionSubscription.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1255:         //
1256:         // buttonAeSubscriptionServer
1257:         //
1258:         this.buttonAeSubscriptionServer.FlatStyle = System.Windows.Forms.FlatStyle.System;
1259:         this.buttonAeSubscriptionServer.Location = new System.Drawing.Point(216, 48);
1260:         this.buttonAeSubscriptionServer.Name = "buttonAeSubscriptionServer";
1261:         this.buttonAeSubscriptionServer.Size = new System.Drawing.Size(80, 23);
1262:         this.buttonAeSubscriptionServer.TabIndex = 6;
1263:         this.buttonAeSubscriptionServer.Text = "server";
1264:         this.buttonAeSubscriptionServer.Click += new System.EventHandler(this.buttonAeSubscriptionMessageFormatButton_Click);
1265:         this.buttonAeSubscriptionServer.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1266:         this.buttonAeSubscriptionServer.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1267:         //
1268:         // buttonAeSubscriptionCondition
1269:         //
1270:         this.buttonAeSubscriptionCondition.FlatStyle = System.Windows.Forms.FlatStyle.System;
1271:         this.buttonAeSubscriptionCondition.Location = new System.Drawing.Point(136, 48);
1272:         this.buttonAeSubscriptionCondition.Name = "buttonAeSubscriptionCondition";
1273:         this.buttonAeSubscriptionCondition.Size = new System.Drawing.Size(80, 23);
1274:         this.buttonAeSubscriptionCondition.TabIndex = 5;
1275:         this.buttonAeSubscriptionCondition.Text = "condition";
1276:         this.buttonAeSubscriptionCondition.Click += new System.EventHandler(this.buttonAeSubscriptionMessageFormatButton_Click
              );
1277:         this.buttonAeSubscriptionCondition.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1278:         this.buttonAeSubscriptionCondition.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1279:         //
1280:         // buttonAeSubscriptionMessage
1281:         //
1282:         this.buttonAeSubscriptionMessage.FlatStyle = System.Windows.Forms.FlatStyle.System;
1283:         this.buttonAeSubscriptionMessage.Location = new System.Drawing.Point(296, 24);
1284:         this.buttonAeSubscriptionMessage.Name = "buttonAeSubscriptionMessage";
1285:         this.buttonAeSubscriptionMessage.Size = new System.Drawing.Size(80, 23);
1286:         this.buttonAeSubscriptionMessage.TabIndex = 4;
1287:         this.buttonAeSubscriptionMessage.Text = "message";
1288:         this.buttonAeSubscriptionMessage.Click += new System.EventHandler(this.buttonAeSubscriptionMessageFormatButton_Click);
1289:         this.buttonAeSubscriptionMessage.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1290:         this.buttonAeSubscriptionMessage.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1291:         //
1292:         // buttonAeSubscriptionSeverity
1293:         //
1294:         this.buttonAeSubscriptionSeverity.FlatStyle = System.Windows.Forms.FlatStyle.System;
1295:         this.buttonAeSubscriptionSeverity.Location = new System.Drawing.Point(216, 24);
1296:         this.buttonAeSubscriptionSeverity.Name = "buttonAeSubscriptionSeverity";
1297:         this.buttonAeSubscriptionSeverity.Size = new System.Drawing.Size(80, 23);
1298:         this.buttonAeSubscriptionSeverity.TabIndex = 3;
1299:         this.buttonAeSubscriptionSeverity.Text = "severity";
```

```
1300:         this.buttonAeSubscriptionSeverity.Click += new System.EventHandler(this.buttonAeSubscriptionMessageFormatButton_Click)
                ;
1301:         this.buttonAeSubscriptionSeverity.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1302:         this.buttonAeSubscriptionSeverity.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1303:         //
1304:         // buttonAeSubscriptionSource
1305:         //
1306:         this.buttonAeSubscriptionSource.FlatStyle = System.Windows.Forms.FlatStyle.System;
1307:         this.buttonAeSubscriptionSource.Location = new System.Drawing.Point(136, 24);
1308:         this.buttonAeSubscriptionSource.Name = "buttonAeSubscriptionSource";
1309:         this.buttonAeSubscriptionSource.Size = new System.Drawing.Size(80, 23);
1310:         this.buttonAeSubscriptionSource.TabIndex = 2;
1311:         this.buttonAeSubscriptionSource.Text = "source";
1312:         this.buttonAeSubscriptionSource.Click += new System.EventHandler(this.buttonAeSubscriptionMessageFormatButton_Click);
1313:         this.buttonAeSubscriptionSource.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1314:         this.buttonAeSubscriptionSource.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1315:         //
1316:         // textBoxAeSubscriptionMessageFormat
1317:         //
1318:         this.textBoxAeSubscriptionMessageFormat.Location = new System.Drawing.Point(136, 4);
1319:         this.textBoxAeSubscriptionMessageFormat.Name = "textBoxAeSubscriptionMessageFormat";
1320:         this.textBoxAeSubscriptionMessageFormat.Size = new System.Drawing.Size(488, 20);
1321:         this.textBoxAeSubscriptionMessageFormat.TabIndex = 1;
1322:         this.textBoxAeSubscriptionMessageFormat.Text = "";
1323:         this.textBoxAeSubscriptionMessageFormat.TextChanged += new System.EventHandler(this.AnyTextChanged);
1324:         //
1325:         // groupBoxSeverityMappings
1326:         //
1327:         this.groupBoxSeverityMappings.Controls.Add(this.listBoxAeSubscriptionSeverityMappings);
1328:         this.groupBoxSeverityMappings.Controls.Add(this.buttonAeSubscriptionSeverityMappingAddMapping);
1329:         this.groupBoxSeverityMappings.Controls.Add(this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode);
1330:         this.groupBoxSeverityMappings.Controls.Add(this.textBoxAeSubscriptionSeverityMappingsRange);
1331:         this.groupBoxSeverityMappings.Controls.Add(this.labelAeSubscriptionSeverityMappingsAcsAlarmCode);
1332:         this.groupBoxSeverityMappings.Controls.Add(this.labelAeSubscriptionSeverityMappingsRange);
1333:         this.groupBoxSeverityMappings.Location = new System.Drawing.Point(392, 80);
1334:         this.groupBoxSeverityMappings.Name = "groupBoxSeverityMappings";
1335:         this.groupBoxSeverityMappings.Size = new System.Drawing.Size(232, 192);
1336:         this.groupBoxSeverityMappings.TabIndex = 9;
1337:         this.groupBoxSeverityMappings.TabStop = false;
1338:         this.groupBoxSeverityMappings.Text = "Severity Mappings";
1339:         //
1340:         // listBoxAeSubscriptionSeverityMappings
1341:         //
1342:         this.listBoxAeSubscriptionSeverityMappings.Location = new System.Drawing.Point(8, 89);
1343:         this.listBoxAeSubscriptionSeverityMappings.Name = "listBoxAeSubscriptionSeverityMappings";
1344:         this.listBoxAeSubscriptionSeverityMappings.Size = new System.Drawing.Size(216, 95);
1345:         this.listBoxAeSubscriptionSeverityMappings.TabIndex = 4;
```

```
1346:         this.listBoxAeSubscriptionSeverityMappings.KeyUp += new System.Windows.Forms.KeyEventHandler(this.
                listBoxAeSubscriptionSeverityMappings_KeyUp);
1347:         //
1348:         // buttonAeSubscriptionSeverityMappingAddMapping
1349:         //
1350:         this.buttonAeSubscriptionSeverityMappingAddMapping.FlatStyle = System.Windows.Forms.FlatStyle.System;
1351:         this.buttonAeSubscriptionSeverityMappingAddMapping.Location = new System.Drawing.Point(8, 64);
1352:         this.buttonAeSubscriptionSeverityMappingAddMapping.Name = "buttonAeSubscriptionSeverityMappingAddMapping";
1353:         this.buttonAeSubscriptionSeverityMappingAddMapping.Size = new System.Drawing.Size(216, 23);
1354:         this.buttonAeSubscriptionSeverityMappingAddMapping.TabIndex = 3;
1355:         this.buttonAeSubscriptionSeverityMappingAddMapping.Text = "Add Mapping";
1356:         this.buttonAeSubscriptionSeverityMappingAddMapping.Click += new System.EventHandler(this.
                buttonAeSubscriptionSeverityMappingAddMapping_Click);
1357:         this.buttonAeSubscriptionSeverityMappingAddMapping.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1358:         this.buttonAeSubscriptionSeverityMappingAddMapping.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1359:         //
1360:         // comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode
1361:         //
1362:         this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.ItemHeight = 13;
1363:         this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.Location = new System.Drawing.Point(80, 40);
1364:         this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.Name = "comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode";
1365:         this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.Size = new System.Drawing.Size(144, 21);
1366:         this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.TabIndex = 2;
1367:         this.comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.Text = "Select ACS Alarm Code";
1368:         //
1369:         // textBoxAeSubscriptionSeverityMappingsRange
1370:         //
1371:         this.textBoxAeSubscriptionSeverityMappingsRange.Location = new System.Drawing.Point(8, 40);
1372:         this.textBoxAeSubscriptionSeverityMappingsRange.Name = "textBoxAeSubscriptionSeverityMappingsRange";
1373:         this.textBoxAeSubscriptionSeverityMappingsRange.RightToLeft = System.Windows.Forms.RightToLeft.No;
1374:         this.textBoxAeSubscriptionSeverityMappingsRange.Size = new System.Drawing.Size(56, 20);
1375:         this.textBoxAeSubscriptionSeverityMappingsRange.TabIndex = 1;
1376:         this.textBoxAeSubscriptionSeverityMappingsRange.Text = "";
1377:         this.textBoxAeSubscriptionSeverityMappingsRange.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
1378:         //
1379:         // labelAeSubscriptionSeverityMappingsAcsAlarmCode
1380:         //
1381:         this.labelAeSubscriptionSeverityMappingsAcsAlarmCode.Location = new System.Drawing.Point(80, 24);
1382:         this.labelAeSubscriptionSeverityMappingsAcsAlarmCode.Name = "labelAeSubscriptionSeverityMappingsAcsAlarmCode";
1383:         this.labelAeSubscriptionSeverityMappingsAcsAlarmCode.Size = new System.Drawing.Size(104, 16);
1384:         this.labelAeSubscriptionSeverityMappingsAcsAlarmCode.TabIndex = 12;
1385:         this.labelAeSubscriptionSeverityMappingsAcsAlarmCode.Text = "ACS Alarm Code:";
1386:         this.labelAeSubscriptionSeverityMappingsAcsAlarmCode.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1387:         //
1388:         // labelAeSubscriptionSeverityMappingsRange
1389:         //
1390:         this.labelAeSubscriptionSeverityMappingsRange.Location = new System.Drawing.Point(8, 24);
```

```
1391:         this.labelAeSubscriptionSeverityMappingsRange.Name = "labelAeSubscriptionSeverityMappingsRange";
1392:         this.labelAeSubscriptionSeverityMappingsRange.Size = new System.Drawing.Size(88, 16);
1393:         this.labelAeSubscriptionSeverityMappingsRange.TabIndex = 11;
1394:         this.labelAeSubscriptionSeverityMappingsRange.Text = "Range:";
1395:         this.labelAeSubscriptionSeverityMappingsRange.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1396:         //
1397:         // groupBoxFilter
1398:         //
1399:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterLowestSeverityInfo);
1400:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterHighestSeverityInfo);
1401:         this.groupBoxFilter.Controls.Add(this.buttonAeSubscriptionFilterEventCatIdsSelect);
1402:         this.groupBoxFilter.Controls.Add(this.buttonAeSubscriptionFilterAreasSelect);
1403:         this.groupBoxFilter.Controls.Add(this.buttonAeSubscriptionFilterSourcesSelect);
1404:         this.groupBoxFilter.Controls.Add(this.radioButtonAeSubscriptionFilterEnabledOff);
1405:         this.groupBoxFilter.Controls.Add(this.radioButtonAeSubscriptionFilterEnabledOn);
1406:         this.groupBoxFilter.Controls.Add(this.comboBoxAeSubscriptionFilterEventType);
1407:         this.groupBoxFilter.Controls.Add(this.textBoxAeSubscriptionFilterHighestSeverity);
1408:         this.groupBoxFilter.Controls.Add(this.textBoxAeSubscriptionFilterLowestSeverity);
1409:         this.groupBoxFilter.Controls.Add(this.textBoxAeSubscriptionFilterEventCatIds);
1410:         this.groupBoxFilter.Controls.Add(this.textBoxAeSubscriptionFilterAreas);
1411:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterHighestSeverity);
1412:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterLowestSeverity);
1413:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterEventCatIds);
1414:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterEventType);
1415:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterAreas);
1416:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterEnabled);
1417:         this.groupBoxFilter.Controls.Add(this.textBoxAeSubscriptionFilterSources);
1418:         this.groupBoxFilter.Controls.Add(this.labelAeSubscriptionFilterSources);
1419:         this.groupBoxFilter.Location = new System.Drawing.Point(0, 80);
1420:         this.groupBoxFilter.Name = "groupBoxFilter";
1421:         this.groupBoxFilter.Size = new System.Drawing.Size(384, 192);
1422:         this.groupBoxFilter.TabIndex = 8;
1423:         this.groupBoxFilter.TabStop = false;
1424:         this.groupBoxFilter.Text = "Filter";
1425:         //
1426:         // labelAeSubscriptionFilterLowestSeverityInfo
1427:         //
1428:         this.labelAeSubscriptionFilterLowestSeverityInfo.Location = new System.Drawing.Point(192, 168);
1429:         this.labelAeSubscriptionFilterLowestSeverityInfo.Name = "labelAeSubscriptionFilterLowestSeverityInfo";
1430:         this.labelAeSubscriptionFilterLowestSeverityInfo.Size = new System.Drawing.Size(144, 16);
1431:         this.labelAeSubscriptionFilterLowestSeverityInfo.TabIndex = 13;
1432:         this.labelAeSubscriptionFilterLowestSeverityInfo.Text = "      0 = No Lowest Severity";
1433:         this.labelAeSubscriptionFilterLowestSeverityInfo.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1434:         //
1435:         // labelAeSubscriptionFilterHighestSeverityInfo
1436:         //
1437:         this.labelAeSubscriptionFilterHighestSeverityInfo.Location = new System.Drawing.Point(192, 144);
```

```
1438:        this.labelAeSubscriptionFilterHighestSeverityInfo.Name = "labelAeSubscriptionFilterHighestSeverityInfo";
1439:        this.labelAeSubscriptionFilterHighestSeverityInfo.Size = new System.Drawing.Size(144, 16);
1440:        this.labelAeSubscriptionFilterHighestSeverityInfo.TabIndex = 12;
1441:        this.labelAeSubscriptionFilterHighestSeverityInfo.Text = "1000 = No Highest Severity";
1442:        this.labelAeSubscriptionFilterHighestSeverityInfo.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1443:        //
1444:        // buttonAeSubscriptionFilterEventCatIdsSelect
1445:        //
1446:        this.buttonAeSubscriptionFilterEventCatIdsSelect.FlatStyle = System.Windows.Forms.FlatStyle.System;
1447:        this.buttonAeSubscriptionFilterEventCatIdsSelect.Location = new System.Drawing.Point(320, 113);
1448:        this.buttonAeSubscriptionFilterEventCatIdsSelect.Name = "buttonAeSubscriptionFilterEventCatIdsSelect";
1449:        this.buttonAeSubscriptionFilterEventCatIdsSelect.Size = new System.Drawing.Size(56, 23);
1450:        this.buttonAeSubscriptionFilterEventCatIdsSelect.TabIndex = 9;
1451:        this.buttonAeSubscriptionFilterEventCatIdsSelect.Text = "Select...";
1452:        this.buttonAeSubscriptionFilterEventCatIdsSelect.Click += new System.EventHandler(this.
              buttonAeSubscriptionFilterEventCatIdsSelect_Click);
1453:        this.buttonAeSubscriptionFilterEventCatIdsSelect.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1454:        this.buttonAeSubscriptionFilterEventCatIdsSelect.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1455:        //
1456:        // buttonAeSubscriptionFilterAreasSelect
1457:        //
1458:        this.buttonAeSubscriptionFilterAreasSelect.FlatStyle = System.Windows.Forms.FlatStyle.System;
1459:        this.buttonAeSubscriptionFilterAreasSelect.Location = new System.Drawing.Point(320, 65);
1460:        this.buttonAeSubscriptionFilterAreasSelect.Name = "buttonAeSubscriptionFilterAreasSelect";
1461:        this.buttonAeSubscriptionFilterAreasSelect.Size = new System.Drawing.Size(56, 23);
1462:        this.buttonAeSubscriptionFilterAreasSelect.TabIndex = 6;
1463:        this.buttonAeSubscriptionFilterAreasSelect.Text = "Select...";
1464:        this.buttonAeSubscriptionFilterAreasSelect.Click += new System.EventHandler(this.
              buttonAeSubscriptionFilterAreasSelect_Click);
1465:        this.buttonAeSubscriptionFilterAreasSelect.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1466:        this.buttonAeSubscriptionFilterAreasSelect.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1467:        //
1468:        // buttonAeSubscriptionFilterSourcesSelect
1469:        //
1470:        this.buttonAeSubscriptionFilterSourcesSelect.FlatStyle = System.Windows.Forms.FlatStyle.System;
1471:        this.buttonAeSubscriptionFilterSourcesSelect.Location = new System.Drawing.Point(320, 40);
1472:        this.buttonAeSubscriptionFilterSourcesSelect.Name = "buttonAeSubscriptionFilterSourcesSelect";
1473:        this.buttonAeSubscriptionFilterSourcesSelect.Size = new System.Drawing.Size(56, 23);
1474:        this.buttonAeSubscriptionFilterSourcesSelect.TabIndex = 4;
1475:        this.buttonAeSubscriptionFilterSourcesSelect.Text = "Select...";
1476:        this.buttonAeSubscriptionFilterSourcesSelect.Click += new System.EventHandler(this.
              buttonAeSubscriptionFilterSourcesSelect_Click);
1477:        this.buttonAeSubscriptionFilterSourcesSelect.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1478:        this.buttonAeSubscriptionFilterSourcesSelect.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1479:        //
1480:        // radioButtonAeSubscriptionFilterEnabledOff
1481:        //
```

```
1482:       this.radioButtonAeSubscriptionFilterEnabledOff.Location = new System.Drawing.Point(224, 24);
1483:       this.radioButtonAeSubscriptionFilterEnabledOff.Name = "radioButtonAeSubscriptionFilterEnabledOff";
1484:       this.radioButtonAeSubscriptionFilterEnabledOff.Size = new System.Drawing.Size(96, 16);
1485:       this.radioButtonAeSubscriptionFilterEnabledOff.TabIndex = 2;
1486:       this.radioButtonAeSubscriptionFilterEnabledOff.Text = "Off";
1487:       this.radioButtonAeSubscriptionFilterEnabledOff.CheckedChanged += new System.EventHandler(this.
              radioButton_CheckedChanged);
1488:       //
1489:       // radioButtonAeSubscriptionFilterEnabledOn
1490:       //
1491:       this.radioButtonAeSubscriptionFilterEnabledOn.Location = new System.Drawing.Point(136, 24);
1492:       this.radioButtonAeSubscriptionFilterEnabledOn.Name = "radioButtonAeSubscriptionFilterEnabledOn";
1493:       this.radioButtonAeSubscriptionFilterEnabledOn.Size = new System.Drawing.Size(88, 16);
1494:       this.radioButtonAeSubscriptionFilterEnabledOn.TabIndex = 1;
1495:       this.radioButtonAeSubscriptionFilterEnabledOn.Text = "On";
1496:       this.radioButtonAeSubscriptionFilterEnabledOn.CheckedChanged += new System.EventHandler(this.
              radioButton_CheckedChanged);
1497:       //
1498:       // comboBoxAeSubscriptionFilterEventType
1499:       //
1500:       this.comboBoxAeSubscriptionFilterEventType.ItemHeight = 13;
1501:       this.comboBoxAeSubscriptionFilterEventType.Items.AddRange(new object[] {
1502:                                                 "All",
1503:                                                 "Condition",
1504:                                                 "Simple",
1505:                                                 "Tracking"});
1506:       this.comboBoxAeSubscriptionFilterEventType.Location = new System.Drawing.Point(136, 91);
1507:       this.comboBoxAeSubscriptionFilterEventType.Name = "comboBoxAeSubscriptionFilterEventType";
1508:       this.comboBoxAeSubscriptionFilterEventType.Size = new System.Drawing.Size(121, 21);
1509:       this.comboBoxAeSubscriptionFilterEventType.TabIndex = 7;
1510:       this.comboBoxAeSubscriptionFilterEventType.SelectionChangeCommitted += new System.EventHandler(this.
              comboBox_SelectionChangeCommitted);
1511:       //
1512:       // textBoxAeSubscriptionFilterHighestSeverity
1513:       //
1514:       this.textBoxAeSubscriptionFilterHighestSeverity.Location = new System.Drawing.Point(136, 140);
1515:       this.textBoxAeSubscriptionFilterHighestSeverity.Name = "textBoxAeSubscriptionFilterHighestSeverity";
1516:       this.textBoxAeSubscriptionFilterHighestSeverity.RightToLeft = System.Windows.Forms.RightToLeft.No;
1517:       this.textBoxAeSubscriptionFilterHighestSeverity.Size = new System.Drawing.Size(48, 20);
1518:       this.textBoxAeSubscriptionFilterHighestSeverity.TabIndex = 10;
1519:       this.textBoxAeSubscriptionFilterHighestSeverity.Text = "";
1520:       this.textBoxAeSubscriptionFilterHighestSeverity.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
1521:       this.textBoxAeSubscriptionFilterHighestSeverity.TextChanged += new System.EventHandler(this.AnyTextChanged);
1522:       //
1523:       // textBoxAeSubscriptionFilterLowestSeverity
1524:       //
1525:       this.textBoxAeSubscriptionFilterLowestSeverity.Location = new System.Drawing.Point(136, 164);
```

```
1526:         this.textBoxAeSubscriptionFilterLowestSeverity.Name = "textBoxAeSubscriptionFilterLowestSeverity";
1527:         this.textBoxAeSubscriptionFilterLowestSeverity.RightToLeft = System.Windows.Forms.RightToLeft.No;
1528:         this.textBoxAeSubscriptionFilterLowestSeverity.Size = new System.Drawing.Size(48, 20);
1529:         this.textBoxAeSubscriptionFilterLowestSeverity.TabIndex = 11;
1530:         this.textBoxAeSubscriptionFilterLowestSeverity.Text = "";
1531:         this.textBoxAeSubscriptionFilterLowestSeverity.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
1532:         this.textBoxAeSubscriptionFilterLowestSeverity.TextChanged += new System.EventHandler(this.AnyTextChanged);
1533:         //
1534:         // textBoxAeSubscriptionFilterEventCatIds
1535:         //
1536:         this.textBoxAeSubscriptionFilterEventCatIds.Location = new System.Drawing.Point(136, 116);
1537:         this.textBoxAeSubscriptionFilterEventCatIds.Name = "textBoxAeSubscriptionFilterEventCatIds";
1538:         this.textBoxAeSubscriptionFilterEventCatIds.Size = new System.Drawing.Size(176, 20);
1539:         this.textBoxAeSubscriptionFilterEventCatIds.TabIndex = 8;
1540:         this.textBoxAeSubscriptionFilterEventCatIds.Text = "";
1541:         this.textBoxAeSubscriptionFilterEventCatIds.TextChanged += new System.EventHandler(this.AnyTextChanged);
1542:         //
1543:         // textBoxAeSubscriptionFilterAreas
1544:         //
1545:         this.textBoxAeSubscriptionFilterAreas.Location = new System.Drawing.Point(136, 68);
1546:         this.textBoxAeSubscriptionFilterAreas.Name = "textBoxAeSubscriptionFilterAreas";
1547:         this.textBoxAeSubscriptionFilterAreas.Size = new System.Drawing.Size(176, 20);
1548:         this.textBoxAeSubscriptionFilterAreas.TabIndex = 5;
1549:         this.textBoxAeSubscriptionFilterAreas.Text = "";
1550:         this.textBoxAeSubscriptionFilterAreas.TextChanged += new System.EventHandler(this.AnyTextChanged);
1551:         //
1552:         // labelAeSubscriptionFilterHighestSeverity
1553:         //
1554:         this.labelAeSubscriptionFilterHighestSeverity.Location = new System.Drawing.Point(8, 144);
1555:         this.labelAeSubscriptionFilterHighestSeverity.Name = "labelAeSubscriptionFilterHighestSeverity";
1556:         this.labelAeSubscriptionFilterHighestSeverity.Size = new System.Drawing.Size(136, 16);
1557:         this.labelAeSubscriptionFilterHighestSeverity.TabIndex = 10;
1558:         this.labelAeSubscriptionFilterHighestSeverity.Text = "Highest Severity:";
1559:         this.labelAeSubscriptionFilterHighestSeverity.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1560:         //
1561:         // labelAeSubscriptionFilterLowestSeverity
1562:         //
1563:         this.labelAeSubscriptionFilterLowestSeverity.Location = new System.Drawing.Point(8, 168);
1564:         this.labelAeSubscriptionFilterLowestSeverity.Name = "labelAeSubscriptionFilterLowestSeverity";
1565:         this.labelAeSubscriptionFilterLowestSeverity.Size = new System.Drawing.Size(136, 16);
1566:         this.labelAeSubscriptionFilterLowestSeverity.TabIndex = 9;
1567:         this.labelAeSubscriptionFilterLowestSeverity.Text = "Lowest Severity:";
1568:         this.labelAeSubscriptionFilterLowestSeverity.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1569:         //
1570:         // labelAeSubscriptionFilterEventCatIds
1571:         //
1572:         this.labelAeSubscriptionFilterEventCatIds.Location = new System.Drawing.Point(8, 120);
```

```
1573:         this.labelAeSubscriptionFilterEventCatIds.Name = "labelAeSubscriptionFilterEventCatIds";
1574:         this.labelAeSubscriptionFilterEventCatIds.Size = new System.Drawing.Size(136, 16);
1575:         this.labelAeSubscriptionFilterEventCatIds.TabIndex = 8;
1576:         this.labelAeSubscriptionFilterEventCatIds.Text = "Event Category Ids:";
1577:         this.labelAeSubscriptionFilterEventCatIds.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1578:         //
1579:         // labelAeSubscriptionFilterEventType
1580:         //
1581:         this.labelAeSubscriptionFilterEventType.Location = new System.Drawing.Point(8, 96);
1582:         this.labelAeSubscriptionFilterEventType.Name = "labelAeSubscriptionFilterEventType";
1583:         this.labelAeSubscriptionFilterEventType.Size = new System.Drawing.Size(136, 16);
1584:         this.labelAeSubscriptionFilterEventType.TabIndex = 7;
1585:         this.labelAeSubscriptionFilterEventType.Text = "Event Type:";
1586:         this.labelAeSubscriptionFilterEventType.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1587:         //
1588:         // labelAeSubscriptionFilterAreas
1589:         //
1590:         this.labelAeSubscriptionFilterAreas.Location = new System.Drawing.Point(8, 72);
1591:         this.labelAeSubscriptionFilterAreas.Name = "labelAeSubscriptionFilterAreas";
1592:         this.labelAeSubscriptionFilterAreas.Size = new System.Drawing.Size(136, 16);
1593:         this.labelAeSubscriptionFilterAreas.TabIndex = 6;
1594:         this.labelAeSubscriptionFilterAreas.Text = "Areas:";
1595:         this.labelAeSubscriptionFilterAreas.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1596:         //
1597:         // labelAeSubscriptionFilterEnabled
1598:         //
1599:         this.labelAeSubscriptionFilterEnabled.Location = new System.Drawing.Point(8, 24);
1600:         this.labelAeSubscriptionFilterEnabled.Name = "labelAeSubscriptionFilterEnabled";
1601:         this.labelAeSubscriptionFilterEnabled.Size = new System.Drawing.Size(136, 16);
1602:         this.labelAeSubscriptionFilterEnabled.TabIndex = 4;
1603:         this.labelAeSubscriptionFilterEnabled.Text = "Enabled:";
1604:         this.labelAeSubscriptionFilterEnabled.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1605:         //
1606:         // textBoxAeSubscriptionFilterSources
1607:         //
1608:         this.textBoxAeSubscriptionFilterSources.Location = new System.Drawing.Point(136, 44);
1609:         this.textBoxAeSubscriptionFilterSources.Name = "textBoxAeSubscriptionFilterSources";
1610:         this.textBoxAeSubscriptionFilterSources.Size = new System.Drawing.Size(176, 20);
1611:         this.textBoxAeSubscriptionFilterSources.TabIndex = 3;
1612:         this.textBoxAeSubscriptionFilterSources.Text = "";
1613:         this.textBoxAeSubscriptionFilterSources.TextChanged += new System.EventHandler(this.AnyTextChanged);
1614:         //
1615:         // labelAeSubscriptionFilterSources
1616:         //
1617:         this.labelAeSubscriptionFilterSources.Location = new System.Drawing.Point(8, 48);
1618:         this.labelAeSubscriptionFilterSources.Name = "labelAeSubscriptionFilterSources";
1619:         this.labelAeSubscriptionFilterSources.Size = new System.Drawing.Size(136, 16);
```

```
1620:       this.labelAeSubscriptionFilterSources.TabIndex = 5;
1621:       this.labelAeSubscriptionFilterSources.Text = "Sources:";
1622:       this.labelAeSubscriptionFilterSources.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1623:       //
1624:       // labelAeSubscriptionMessageFormat
1625:       //
1626:       this.labelAeSubscriptionMessageFormat.Location = new System.Drawing.Point(0, 8);
1627:       this.labelAeSubscriptionMessageFormat.Name = "labelAeSubscriptionMessageFormat";
1628:       this.labelAeSubscriptionMessageFormat.Size = new System.Drawing.Size(144, 16);
1629:       this.labelAeSubscriptionMessageFormat.TabIndex = 3;
1630:       this.labelAeSubscriptionMessageFormat.Text = "Message Format:";
1631:       this.labelAeSubscriptionMessageFormat.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
1632:       //
1633:       // buttonAddDaSubscription
1634:       //
1635:       this.buttonAddDaSubscription.Enabled = false;
1636:       this.buttonAddDaSubscription.FlatStyle = System.Windows.Forms.FlatStyle.System;
1637:       this.buttonAddDaSubscription.Location = new System.Drawing.Point(128, 376);
1638:       this.buttonAddDaSubscription.Name = "buttonAddDaSubscription";
1639:       this.buttonAddDaSubscription.Size = new System.Drawing.Size(120, 23);
1640:       this.buttonAddDaSubscription.TabIndex = 6;
1641:       this.buttonAddDaSubscription.Text = "Add DA Subscription";
1642:       this.buttonAddDaSubscription.Click += new System.EventHandler(this.buttonAddDaSubscription_Click);
1643:       this.buttonAddDaSubscription.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1644:       this.buttonAddDaSubscription.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1645:       //
1646:       // buttonAddAeSubscription
1647:       //
1648:       this.buttonAddAeSubscription.Enabled = false;
1649:       this.buttonAddAeSubscription.FlatStyle = System.Windows.Forms.FlatStyle.System;
1650:       this.buttonAddAeSubscription.Location = new System.Drawing.Point(8, 376);
1651:       this.buttonAddAeSubscription.Name = "buttonAddAeSubscription";
1652:       this.buttonAddAeSubscription.Size = new System.Drawing.Size(120, 23);
1653:       this.buttonAddAeSubscription.TabIndex = 5;
1654:       this.buttonAddAeSubscription.Text = "Add A&&E Subscription";
1655:       this.buttonAddAeSubscription.Click += new System.EventHandler(this.buttonAddAeSubscription_Click);
1656:       this.buttonAddAeSubscription.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1657:       this.buttonAddAeSubscription.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1658:       //
1659:       // buttonAddServer
1660:       //
1661:       this.buttonAddServer.FlatStyle = System.Windows.Forms.FlatStyle.System;
1662:       this.buttonAddServer.Location = new System.Drawing.Point(8, 352);
1663:       this.buttonAddServer.Name = "buttonAddServer";
1664:       this.buttonAddServer.Size = new System.Drawing.Size(240, 24);
1665:       this.buttonAddServer.TabIndex = 4;
1666:       this.buttonAddServer.Text = "Add Server";
```

```
1667:         this.buttonAddServer.Click += new System.EventHandler(this.buttonAddServer_Click);
1668:         this.buttonAddServer.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1669:         this.buttonAddServer.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1670:         //
1671:         // buttonRemoveSelected
1672:         //
1673:         this.buttonRemoveSelected.Enabled = false;
1674:         this.buttonRemoveSelected.FlatStyle = System.Windows.Forms.FlatStyle.System;
1675:         this.buttonRemoveSelected.Location = new System.Drawing.Point(8, 328);
1676:         this.buttonRemoveSelected.Name = "buttonRemoveSelected";
1677:         this.buttonRemoveSelected.Size = new System.Drawing.Size(240, 23);
1678:         this.buttonRemoveSelected.TabIndex = 3;
1679:         this.buttonRemoveSelected.Text = "Remove Selected";
1680:         this.buttonRemoveSelected.Click += new System.EventHandler(this.buttonRemoveSelected_Click);
1681:         this.buttonRemoveSelected.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1682:         this.buttonRemoveSelected.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1683:         //
1684:         // groupBoxServerList
1685:         //
1686:         this.groupBoxServerList.Controls.Add(this.treeViewServers);
1687:         this.groupBoxServerList.Location = new System.Drawing.Point(8, 8);
1688:         this.groupBoxServerList.Name = "groupBoxServerList";
1689:         this.groupBoxServerList.Size = new System.Drawing.Size(240, 312);
1690:         this.groupBoxServerList.TabIndex = 2;
1691:         this.groupBoxServerList.TabStop = false;
1692:         this.groupBoxServerList.Text = "Servers";
1693:         //
1694:         // treeViewServers
1695:         //
1696:         this.treeViewServers.ImageList = this.imageListTreeView;
1697:         this.treeViewServers.Location = new System.Drawing.Point(8, 16);
1698:         this.treeViewServers.Name = "treeViewServers";
1699:         this.treeViewServers.Size = new System.Drawing.Size(224, 288);
1700:         this.treeViewServers.TabIndex = 0;
1701:         this.treeViewServers.AfterSelect += new System.Windows.Forms.TreeViewEventHandler(this.treeViewServers_AfterSelect);
1702:         //
1703:         // imageListTreeView
1704:         //
1705:         this.imageListTreeView.ImageSize = new System.Drawing.Size(16, 16);
1706:         this.imageListTreeView.ImageStream = ((System.Windows.Forms.ImageListStreamer)(resources.GetObject("imageListTreeView.
              ImageStream")));
1707:         this.imageListTreeView.TransparentColor = System.Drawing.Color.Transparent;
1708:         //
1709:         // pictureBoxConfigurationLogo
1710:         //
1711:         this.pictureBoxConfigurationLogo.Image = ((System.Drawing.Image)(resources.GetObject("pictureBoxConfigurationLogo.
              Image")));
```

```
1712:       this.pictureBoxConfigurationLogo.Location = new System.Drawing.Point(768, 408);
1713:       this.pictureBoxConfigurationLogo.Name = "pictureBoxConfigurationLogo";
1714:       this.pictureBoxConfigurationLogo.Size = new System.Drawing.Size(128, 16);
1715:       this.pictureBoxConfigurationLogo.TabIndex = 8;
1716:       this.pictureBoxConfigurationLogo.TabStop = false;
1717:       //
1718:       // tabService
1719:       //
1720:       this.tabService.Controls.Add(this.buttonServiceStopService);
1721:       this.tabService.Controls.Add(this.buttonServiceStartService);
1722:       this.tabService.Controls.Add(this.progressBarService);
1723:       this.tabService.Controls.Add(this.labelServiceStatus);
1724:       this.tabService.Controls.Add(this.groupBoxServiceInformation);
1725:       this.tabService.Controls.Add(this.pictureBoxServiceLogo);
1726:       this.tabService.Location = new System.Drawing.Point(4, 22);
1727:       this.tabService.Name = "tabService";
1728:       this.tabService.Size = new System.Drawing.Size(1528, 1094);
1729:       this.tabService.TabIndex = 1;
1730:       this.tabService.Text = "Service";
1731:       //
1732:       // buttonServiceStopService
1733:       //
1734:       this.buttonServiceStopService.Enabled = false;
1735:       this.buttonServiceStopService.FlatStyle = System.Windows.Forms.FlatStyle.System;
1736:       this.buttonServiceStopService.Location = new System.Drawing.Point(152, 152);
1737:       this.buttonServiceStopService.Name = "buttonServiceStopService";
1738:       this.buttonServiceStopService.Size = new System.Drawing.Size(136, 23);
1739:       this.buttonServiceStopService.TabIndex = 5;
1740:       this.buttonServiceStopService.Text = "Stop Service";
1741:       this.buttonServiceStopService.Click += new System.EventHandler(this.buttonServiceStopService_Click);
1742:       //
1743:       // buttonServiceStartService
1744:       //
1745:       this.buttonServiceStartService.Enabled = false;
1746:       this.buttonServiceStartService.FlatStyle = System.Windows.Forms.FlatStyle.System;
1747:       this.buttonServiceStartService.Location = new System.Drawing.Point(8, 152);
1748:       this.buttonServiceStartService.Name = "buttonServiceStartService";
1749:       this.buttonServiceStartService.Size = new System.Drawing.Size(136, 23);
1750:       this.buttonServiceStartService.TabIndex = 4;
1751:       this.buttonServiceStartService.Text = "Start Service";
1752:       this.buttonServiceStartService.Click += new System.EventHandler(this.buttonServiceStartService_Click);
1753:       //
1754:       // progressBarService
1755:       //
1756:       this.progressBarService.Location = new System.Drawing.Point(760, 8);
1757:       this.progressBarService.Name = "progressBarService";
1758:       this.progressBarService.Size = new System.Drawing.Size(136, 16);
```

```
1759:         this.progressBarService.TabIndex = 3;
1760:         this.progressBarService.Visible = false;
1761:         //
1762:         // labelServiceStatus
1763:         //
1764:         this.labelServiceStatus.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Bold,
                System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
1765:         this.labelServiceStatus.Location = new System.Drawing.Point(8, 8);
1766:         this.labelServiceStatus.Name = "labelServiceStatus";
1767:         this.labelServiceStatus.Size = new System.Drawing.Size(888, 16);
1768:         this.labelServiceStatus.TabIndex = 2;
1769:         this.labelServiceStatus.Text = "Please wait while fetching information...";
1770:         this.labelServiceStatus.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
1771:         //
1772:         // groupBoxServiceInformation
1773:         //
1774:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceServiceStatusData);
1775:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceServiceTypeData);
1776:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceServiceNameData);
1777:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceDisplayNameData);
1778:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceServiceStatus);
1779:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceServiceType);
1780:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceServiceName);
1781:         this.groupBoxServiceInformation.Controls.Add(this.labelServiceDisplayName);
1782:         this.groupBoxServiceInformation.Location = new System.Drawing.Point(8, 24);
1783:         this.groupBoxServiceInformation.Name = "groupBoxServiceInformation";
1784:         this.groupBoxServiceInformation.Size = new System.Drawing.Size(888, 120);
1785:         this.groupBoxServiceInformation.TabIndex = 1;
1786:         this.groupBoxServiceInformation.TabStop = false;
1787:         this.groupBoxServiceInformation.Text = "Service Information";
1788:         //
1789:         // labelServiceServiceStatusData
1790:         //
1791:         this.labelServiceServiceStatusData.Location = new System.Drawing.Point(96, 96);
1792:         this.labelServiceServiceStatusData.Name = "labelServiceServiceStatusData";
1793:         this.labelServiceServiceStatusData.Size = new System.Drawing.Size(784, 16);
1794:         this.labelServiceServiceStatusData.TabIndex = 7;
1795:         this.labelServiceServiceStatusData.Text = "N/A";
1796:         //
1797:         // labelServiceServiceTypeData
1798:         //
1799:         this.labelServiceServiceTypeData.Location = new System.Drawing.Point(96, 72);
1800:         this.labelServiceServiceTypeData.Name = "labelServiceServiceTypeData";
1801:         this.labelServiceServiceTypeData.Size = new System.Drawing.Size(784, 16);
1802:         this.labelServiceServiceTypeData.TabIndex = 6;
1803:         this.labelServiceServiceTypeData.Text = "N/A";
1804:         //
```

```
1805:       // labelServiceServiceNameData
1806:       //
1807:       this.labelServiceServiceNameData.Location = new System.Drawing.Point(96, 48);
1808:       this.labelServiceServiceNameData.Name = "labelServiceServiceNameData";
1809:       this.labelServiceServiceNameData.Size = new System.Drawing.Size(784, 16);
1810:       this.labelServiceServiceNameData.TabIndex = 5;
1811:       this.labelServiceServiceNameData.Text = "N/A";
1812:       //
1813:       // labelServiceDisplayNameData
1814:       //
1815:       this.labelServiceDisplayNameData.Location = new System.Drawing.Point(96, 24);
1816:       this.labelServiceDisplayNameData.Name = "labelServiceDisplayNameData";
1817:       this.labelServiceDisplayNameData.Size = new System.Drawing.Size(784, 16);
1818:       this.labelServiceDisplayNameData.TabIndex = 4;
1819:       this.labelServiceDisplayNameData.Text = "N/A";
1820:       //
1821:       // labelServiceServiceStatus
1822:       //
1823:       this.labelServiceServiceStatus.Location = new System.Drawing.Point(8, 96);
1824:       this.labelServiceServiceStatus.Name = "labelServiceServiceStatus";
1825:       this.labelServiceServiceStatus.Size = new System.Drawing.Size(88, 16);
1826:       this.labelServiceServiceStatus.TabIndex = 3;
1827:       this.labelServiceServiceStatus.Text = "Service Status:";
1828:       //
1829:       // labelServiceServiceType
1830:       //
1831:       this.labelServiceServiceType.Location = new System.Drawing.Point(8, 72);
1832:       this.labelServiceServiceType.Name = "labelServiceServiceType";
1833:       this.labelServiceServiceType.Size = new System.Drawing.Size(88, 16);
1834:       this.labelServiceServiceType.TabIndex = 2;
1835:       this.labelServiceServiceType.Text = "Service Type:";
1836:       //
1837:       // labelServiceServiceName
1838:       //
1839:       this.labelServiceServiceName.Location = new System.Drawing.Point(8, 48);
1840:       this.labelServiceServiceName.Name = "labelServiceServiceName";
1841:       this.labelServiceServiceName.Size = new System.Drawing.Size(88, 16);
1842:       this.labelServiceServiceName.TabIndex = 1;
1843:       this.labelServiceServiceName.Text = "Service Name:";
1844:       //
1845:       // labelServiceDisplayName
1846:       //
1847:       this.labelServiceDisplayName.Location = new System.Drawing.Point(8, 24);
1848:       this.labelServiceDisplayName.Name = "labelServiceDisplayName";
1849:       this.labelServiceDisplayName.Size = new System.Drawing.Size(88, 16);
1850:       this.labelServiceDisplayName.TabIndex = 0;
1851:       this.labelServiceDisplayName.Text = "Display Name:";
```

```
1852:       //
1853:       // pictureBoxServiceLogo
1854:       //
1855:       this.pictureBoxServiceLogo.Image = ((System.Drawing.Image)(resources.GetObject("pictureBoxServiceLogo.Image")));
1856:       this.pictureBoxServiceLogo.Location = new System.Drawing.Point(768, 408);
1857:       this.pictureBoxServiceLogo.Name = "pictureBoxServiceLogo";
1858:       this.pictureBoxServiceLogo.Size = new System.Drawing.Size(128, 16);
1859:       this.pictureBoxServiceLogo.TabIndex = 0;
1860:       this.pictureBoxServiceLogo.TabStop = false;
1861:       //
1862:       // imageListMenu
1863:       //
1864:       this.imageListMenu.ImageSize = new System.Drawing.Size(16, 16);
1865:       this.imageListMenu.ImageStream = ((System.Windows.Forms.ImageListStreamer)(resources.GetObject("imageListMenu.
            ImageStream")));
1866:       this.imageListMenu.TransparentColor = System.Drawing.Color.Transparent;
1867:       //
1868:       // menuImage
1869:       //
1870:       this.menuImage.ImageList = this.imageListMenu;
1871:       //
1872:       // openFileDialog
1873:       //
1874:       this.openFileDialog.FileName = "listener.cfg";
1875:       this.openFileDialog.Filter = "Configuration files (*.cfg)|*.cfg|All files (*.*)|*.*";
1876:       //
1877:       // saveFileDialog
1878:       //
1879:       this.saveFileDialog.FileName = "listener.cfg";
1880:       this.saveFileDialog.Filter = "Configuration files (*.cfg)|*.cfg|All files (*.*)|*.*";
1881:       //
1882:       // buttonAeSubscriptionFormula
1883:       //
1884:       this.buttonAeSubscriptionFormula.FlatStyle = System.Windows.Forms.FlatStyle.System;
1885:       this.buttonAeSubscriptionFormula.Location = new System.Drawing.Point(544, 24);
1886:       this.buttonAeSubscriptionFormula.Name = "buttonAeSubscriptionFormula";
1887:       this.buttonAeSubscriptionFormula.Size = new System.Drawing.Size(80, 23);
1888:       this.buttonAeSubscriptionFormula.TabIndex = 10;
1889:       this.buttonAeSubscriptionFormula.Text = "formula";
1890:       this.buttonAeSubscriptionFormula.Click += new System.EventHandler(this.buttonAeSubscriptionMessageFormatButton_Click);
1891:       this.buttonAeSubscriptionFormula.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1892:       this.buttonAeSubscriptionFormula.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1893:       //
1894:       // buttonDaSubscriptionFormula
1895:       //
1896:       this.buttonDaSubscriptionFormula.FlatStyle = System.Windows.Forms.FlatStyle.System;
1897:       this.buttonDaSubscriptionFormula.Location = new System.Drawing.Point(544, 24);
```

```
1898:       this.buttonDaSubscriptionFormula.Name = "buttonDaSubscriptionFormula";
1899:       this.buttonDaSubscriptionFormula.Size = new System.Drawing.Size(80, 23);
1900:       this.buttonDaSubscriptionFormula.TabIndex = 34;
1901:       this.buttonDaSubscriptionFormula.Text = "formula";
1902:       this.buttonDaSubscriptionFormula.Click += new System.EventHandler(this.buttonDaSubscriptionMessageFormatButton_Click);
1903:       this.buttonDaSubscriptionFormula.MouseEnter += new System.EventHandler(this.button_MouseEnter);
1904:       this.buttonDaSubscriptionFormula.MouseLeave += new System.EventHandler(this.button_MouseLeave);
1905:       //
1906:       // View
1907:       //
1908:       this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
1909:       this.ClientSize = new System.Drawing.Size(904, 481);
1910:       this.Controls.Add(this.statusBar);
1911:       this.Controls.Add(this.tabControl);
1912:       this.MaximizeBox = false;
1913:       this.Menu = this.mainMenu1;
1914:       this.Name = "View";
1915:       this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
1916:       this.Text = "OPC Listener Configuration";
1917:       this.Load += new System.EventHandler(this.View_Load);
1918:       ((System.ComponentModel.ISupportInitialize)(this.statusBarPanelLeft)).EndInit();
1919:       ((System.ComponentModel.ISupportInitialize)(this.statusBarPanelRight)).EndInit();
1920:       this.tabControl.ResumeLayout(false);
1921:       this.tabConfiguration.ResumeLayout(false);
1922:       this.groupBoxInformation.ResumeLayout(false);
1923:       this.groupBoxServer.ResumeLayout(false);
1924:       this.groupBoxServerAcsAlarmCodes.ResumeLayout(false);
1925:       this.groupBoxSubscription.ResumeLayout(false);
1926:       this.panelDaSubscription.ResumeLayout(false);
1927:       this.panelAeSubscription.ResumeLayout(false);
1928:       this.groupBoxSeverityMappings.ResumeLayout(false);
1929:       this.groupBoxFilter.ResumeLayout(false);
1930:       this.groupBoxServerList.ResumeLayout(false);
1931:       this.tabService.ResumeLayout(false);
1932:       this.groupBoxServiceInformation.ResumeLayout(false);
1933:       this.ResumeLayout(false);
1934:
1935:     }
1936:     #endregion
1937:
1938:     #region Main Method
1939:     /// <summary>
1940:     /// The main entry point for the application.
1941:     /// </summary>
1942:     [STAThread]
1943:     static void Main()
1944:     {
```

```
1945:       Application.Run(new View());
1946:     }
1947:
1948:     #endregion
1949:
1950:     #region On Load
1951:
1952:     private void View_Load(object sender, System.EventArgs e)
1953:     {
1954:       // Ensure all panels are in the correct places
1955:       panelDaSubscription.Bounds = panelAeSubscription.Bounds;
1956:       groupBoxServer.Bounds = groupBoxSubscription.Bounds;
1957:       groupBoxInformation.Bounds = groupBoxSubscription.Bounds;
1958:       groupBoxInformation.Visible = true;
1959:       // Add the root node to the tree view
1960:       treeViewServers.Nodes.Add("Servers");
1961:       delegateUpdateProgress = new DelegateUpdateProgress(this.UpdateProgress);
1962:       delegateInitProgress = new DelegateInitProgress(this.InitProgress);
1963:       openFileDialog.InitialDirectory = System.AppDomain.CurrentDomain.BaseDirectory + "config\\";
1964:       saveFileDialog.InitialDirectory = System.AppDomain.CurrentDomain.BaseDirectory + "config\\";
1965:       ModifyRegistry myRegistry = new ModifyRegistry();
1966:       myRegistry.SubKey = REGISTRY_SUB_KEY;
1967:       string language = myRegistry.Read(REGISTRY_VALUE);
1968:       if (language == null)
1969:         LoadNewLanguage(LANGUAGE_ENGLISH);
1970:       else
1971:         LoadNewLanguage(language);
1972:     }
1973:
1974:     #endregion
1975:
1976:     #region View Switchers
1977:     //-----------------------------------------------------------------------------
1978:     // View Switchers
1979:     //
1980:     // Methods which affect the look of the application user interface
1981:     //-----------------------------------------------------------------------------
1982:
1983:     public void SwitchToInformationView()
1984:     {
1985:       groupBoxSubscription.Visible = false;
1986:       groupBoxServer.Visible = false;
1987:       groupBoxInformation.Visible = true;
1988:
1989:       buttonRemoveSelected.Enabled = false;
1990:       buttonAddAeSubscription.Enabled = false;
1991:       buttonAddDaSubscription.Enabled = false;
```

```
1992:
1993:       Update();
1994:     }
1995:
1996:     public void SwitchToServerView()
1997:     {
1998:       groupBoxInformation.Visible = false;
1999:       groupBoxSubscription.Visible = false;
2000:       groupBoxServer.Visible = true;
2001:
2002:       buttonRemoveSelected.Enabled = true;
2003:       buttonAddAeSubscription.Enabled = true;
2004:       buttonAddDaSubscription.Enabled = true;
2005:
2006:       Update();
2007:     }
2008:
2009:     public void SwitchToAeSubscriptionView()
2010:     {
2011:       groupBoxInformation.Visible = false;
2012:       groupBoxServer.Visible = false;
2013:       groupBoxSubscription.Visible = true;
2014:       panelDaSubscription.Visible = false;
2015:       panelAeSubscription.Visible = true;
2016:
2017:       buttonRemoveSelected.Enabled = true;
2018:       buttonAddAeSubscription.Enabled = true;
2019:       buttonAddDaSubscription.Enabled = true;
2020:
2021:       Update();
2022:     }
2023:
2024:     public void SwitchToDaSubscriptionView()
2025:     {
2026:       groupBoxInformation.Visible = false;
2027:       groupBoxServer.Visible = false;
2028:       groupBoxSubscription.Visible = true;
2029:       panelAeSubscription.Visible = false;
2030:       panelDaSubscription.Visible = true;
2031:
2032:       buttonRemoveSelected.Enabled = true;
2033:       buttonAddAeSubscription.Enabled = true;
2034:       buttonAddDaSubscription.Enabled = true;
2035:
2036:       Update();
2037:     }
2038:
```

```
2039:     public void SwitchToAlarmTypeBooleanView()
2040:     {
2041:       textBoxDaSubscriptionAlarmStateSingleValue.Visible = false;
2042:       textBoxDaSubscriptionAlarmStateRangeLow.Visible = false;
2043:       textBoxDaSubscriptionAlarmStateRangeHigh.Visible = false;
2044:       labelDaSubscriptionAlarmStateRangeDash.Visible = false;
2045:       comboBoxDaSubscriptionAlarmStateBooleanValue.Visible = true;
2046:       radioButtonDaSubscriptionAlarmStateTypeBoolean.Checked = true;
2047:       comboBoxDaSubscriptionAlarmStateBooleanValue.SelectedIndex = 0;
2048:     }
2049:
2050:     public void SwitchToAlarmTypeSingleValueView()
2051:     {
2052:       comboBoxDaSubscriptionAlarmStateBooleanValue.Visible = false;
2053:       textBoxDaSubscriptionAlarmStateRangeLow.Visible = false;
2054:       textBoxDaSubscriptionAlarmStateRangeHigh.Visible = false;
2055:       labelDaSubscriptionAlarmStateRangeDash.Visible = false;
2056:       textBoxDaSubscriptionAlarmStateSingleValue.Visible = true;
2057:       radioButtonDaSubscriptionAlarmStateTypeSingleValue.Checked = true;
2058:       textBoxDaSubscriptionAlarmStateSingleValue.Text = "";
2059:     }
2060:
2061:     public void SwitchToAlarmTypeRangeView()
2062:     {
2063:       comboBoxDaSubscriptionAlarmStateBooleanValue.Visible = false;
2064:       textBoxDaSubscriptionAlarmStateSingleValue.Visible = false;
2065:       textBoxDaSubscriptionAlarmStateRangeLow.Visible = true;
2066:       textBoxDaSubscriptionAlarmStateRangeHigh.Visible = true;
2067:       labelDaSubscriptionAlarmStateRangeDash.Visible = true;
2068:       radioButtonDaSubscriptionAlarmStateTypeRange.Checked = true;
2069:       textBoxDaSubscriptionAlarmStateRangeLow.Text = "";
2070:       textBoxDaSubscriptionAlarmStateRangeHigh.Text = "";
2071:     }
2072:     #endregion
2073:
2074:     #region View Updaters
2075:     //-----------------------------------------------------------------------
2076:     // View Updaters
2077:     //
2078:     // Methods used by the controller to manipulate the look of the application
2079:     // user interface
2080:     //-----------------------------------------------------------------------
2081:     public void UpdateTreeView(object[] servers)
2082:     {
2083:       //Remove all the current servers
2084:       treeViewServers.Nodes[0].Nodes.Clear();
2085:
```

```
2086:       foreach (OpcServerNode node1 in servers)
2087:       {
2088:         node1.ImageIndex = 1;
2089:         node1.SelectedImageIndex = 1;
2090:         foreach (OpcSubscriptionNode node2 in node1.Nodes)
2091:         {
2092:           if (node2 is OpcAeSubscriptionNode)
2093:           {
2094:             node2.ImageIndex = 2;
2095:             node2.SelectedImageIndex = 2;
2096:           }
2097:           else if (node2 is OpcDaSubscriptionNode)
2098:           {
2099:             node2.ImageIndex = 3;
2100:             node2.SelectedImageIndex = 3;
2101:           }
2102:         }
2103:         // Add the server
2104:         treeViewServers.Nodes[0].Nodes.Add(node1);
2105:       }
2106:       ShowAllNodesInTree();
2107:     }
2108:
2109:     public void ShowAllNodesInTree()
2110:     {
2111:       treeViewServers.ExpandAll();
2112:       Update();
2113:     }
2114:
2115:     private void InitProgress(int max)
2116:     {
2117:
2118:       progressBarService.Maximum = max;
2119:       progressBarService.Value = 0;
2120:
2121:     }
2122:
2123:     private void UpdateProgress(int newValue)
2124:     {
2125:
2126:       progressBarService.Value = newValue;
2127:       if (newValue == progressBarService.Maximum)
2128:       {
2129:         switch (lastAction)
2130:         {
2131:           case ServiceAction.Started:
2132:             buttonServiceStartService.Enabled = false;
```

```
2133:             buttonServiceStopService.Enabled = true;
2134:             break;
2135:           case ServiceAction.Stopped:
2136:             buttonServiceStartService.Enabled = true;
2137:             buttonServiceStopService.Enabled = false;
2138:             break;
2139:         }
2140:         serviceActionInProcess = false;
2141:         progressBarService.Visible = false;
2142:         progressBarService.Value = 0;
2143:         tabControl_SelectedIndexChanged(null, null);
2144:       }
2145:     }
2146:
2147:     public void UpdateDaSubscriptionItem(string item)
2148:     {
2149:       textBoxDaSubscriptionItem.Text = item;
2150:     }
2151:
2152:     public void UpdateAeSubscriptionFilterSources(string sources)
2153:     {
2154:       textBoxAeSubscriptionFilterSources.Text = sources;
2155:     }
2156:
2157:     public void UpdateAeSubscriptionFilterAreas(string areas)
2158:     {
2159:       textBoxAeSubscriptionFilterAreas.Text = areas;
2160:     }
2161:
2162:     public void UpdateAeSubscriptionFilterEventCatIds(string ids)
2163:     {
2164:       textBoxAeSubscriptionFilterEventCatIds.Text = ids;
2165:     }
2166:
2167:     public void UpdateOpcServerNameRemote(string serverName, string host)
2168:     {
2169:       textBoxServerOpcServerName.Text = serverName;
2170:       textBoxServerOpcServerHost.Text = host;
2171:     }
2172:
2173:     #endregion
2174:
2175:     #region Utility Methods
2176:     //-----------------------------------------------------------------------
2177:     // Utility Methods
2178:     //-----------------------------------------------------------------------
2179:     private OpcServerNode GetSelectedServer()
```

```
2180:     {
2181:       if (treeViewServers.SelectedNode is OpcServerNode)
2182:         return (OpcServerNode) treeViewServers.SelectedNode;
2183:       else if (treeViewServers.SelectedNode is OpcSubscriptionNode)
2184:         return (OpcServerNode) treeViewServers.SelectedNode.Parent;
2185:       return null;
2186:     }
2187:
2188:     private void SaveValuesInModel()
2189:     {
2190:       if (lastSelectedNode != null)
2191:       {
2192:         if (lastSelectedNode is OpcServerNode)
2193:         {
2194:           OpcServerNode node = (OpcServerNode) lastSelectedNode;
2195:           node.OpcServerName = textBoxServerOpcServerName.Text;
2196:           node.OpcServerHost = textBoxServerOpcServerHost.Text;
2197:           node.AcsDefaultUnitId = textBoxServerDefaultUnitId.Text;
2198:           node.AcsDefaultAreaGroups = textBoxServerDefaultAreaGroups.Text;
2199:           node.UseCom = radioButtonServerConnectionTypeCom.Checked;
2200:           node.AcsAlarmCodes = new string[listBoxServerAcsAlarmCodes.Items.Count];
2201:           int count = 0;
2202:           foreach (string code in listBoxServerAcsAlarmCodes.Items)
2203:             node.AcsAlarmCodes[count++] = code;
2204:         }
2205:         else if (lastSelectedNode is OpcAeSubscriptionNode)
2206:         {
2207:           OpcAeSubscriptionNode node = (OpcAeSubscriptionNode) lastSelectedNode;
2208:           node.Enabled = radioButtonSubscriptionEnabledOn.Checked;
2209:           node.SubscriptionName = textBoxSubscriptionName.Text;
2210:           node.AcsUnitId = textBoxSubscriptionAcsUnitId.Text;
2211:           node.AcsAreaGroups = textBoxSubscriptionAcsAreaGroups.Text;
2212:           node.MessageFormat = textBoxAeSubscriptionMessageFormat.Text;
2213:           node.FilterEnabled = radioButtonAeSubscriptionFilterEnabledOn.Checked;
2214:           node.Sources = textBoxAeSubscriptionFilterSources.Text;
2215:           node.Areas = textBoxAeSubscriptionFilterAreas.Text;
2216:           node.EventType = comboBoxAeSubscriptionFilterEventType.SelectedItem.ToString();
2217:           node.CategoryIds = textBoxAeSubscriptionFilterEventCatIds.Text;
2218:           node.LowestSeverity = textBoxAeSubscriptionFilterLowestSeverity.Text;
2219:           node.HighestSeverity = textBoxAeSubscriptionFilterHighestSeverity.Text;
2220:           node.SeverityMappings = new OpcSeverityMapping[listBoxAeSubscriptionSeverityMappings.Items.Count];
2221:           int count = 0;
2222:           foreach (OpcSeverityMapping mapping in listBoxAeSubscriptionSeverityMappings.Items)
2223:             node.SeverityMappings[count++] = mapping;
2224:         }
2225:         else if (lastSelectedNode is OpcDaSubscriptionNode)
2226:         {
```

```
2227:           OpcDaSubscriptionNode node = (OpcDaSubscriptionNode) lastSelectedNode;
2228:           node.Enabled = radioButtonSubscriptionEnabledOn.Checked;
2229:           node.SubscriptionName = textBoxSubscriptionName.Text;
2230:                     node.AcsUnitId = textBoxSubscriptionAcsUnitId.Text;
2231:           node.AcsAreaGroups = textBoxSubscriptionAcsAreaGroups.Text;
2232:           node.MessageFormat = textBoxDaSubscriptionMessageFormat.Text;
2233:           node.Item = textBoxDaSubscriptionItem.Text;
2234:           if (radioButtonDaSubscriptionAlarmStateTypeBoolean.Checked)
2235:           {
2236:             node.AlarmStateType = AlarmStateTypes.BOOLEAN;
2237:             node.AlarmState = comboBoxDaSubscriptionAlarmStateBooleanValue.SelectedItem.ToString();
2238:           }
2239:           else if (radioButtonDaSubscriptionAlarmStateTypeSingleValue.Checked)
2240:           {
2241:             node.AlarmStateType = AlarmStateTypes.SINGLE;
2242:             node.AlarmState = textBoxDaSubscriptionAlarmStateSingleValue.Text;
2243:           }
2244:           else if (radioButtonDaSubscriptionAlarmStateTypeRange.Checked)
2245:           {
2246:             node.AlarmStateType = AlarmStateTypes.RANGE;
2247:             node.AlarmState = textBoxDaSubscriptionAlarmStateRangeLow.Text + " to " +
                  textBoxDaSubscriptionAlarmStateRangeHigh.Text;
2248:           }
2249:           if (comboBoxDaSubscriptionAcsAlarmCode.SelectedItem != null)
2250:             node.AcsAlarmCode = comboBoxDaSubscriptionAcsAlarmCode.SelectedItem.ToString();
2251:         }
2252:       }
2253:     }
2254:
2255:     private void LoadValuesFromModel()
2256:     {
2257:       if (lastSelectedNode != null)
2258:       {
2259:         bool tempDirty = dirty;
2260:         if (lastSelectedNode is OpcServerNode)
2261:         {
2262:           OpcServerNode node = (OpcServerNode) lastSelectedNode;
2263:           textBoxServerOpcServerName.Text = node.OpcServerName;
2264:           textBoxServerOpcServerHost.Text = node.OpcServerHost;
2265:           textBoxServerDefaultUnitId.Text = node.AcsDefaultUnitId;
2266:           textBoxServerDefaultAreaGroups.Text = node.AcsDefaultAreaGroups;
2267:           if (node.UseCom)
2268:             radioButtonServerConnectionTypeCom.Checked = true;
2269:           else
2270:             radioButtonServerConnectionTypeXml.Checked = true;
2271:           listBoxServerAcsAlarmCodes.Items.Clear();
2272:           listBoxServerAcsAlarmCodes.Items.AddRange(node.AcsAlarmCodes);
```

```
2273:         }
2274:         else if (lastSelectedNode is OpcAeSubscriptionNode)
2275:         {
2276:           OpcAeSubscriptionNode node = (OpcAeSubscriptionNode) lastSelectedNode;
2277:           if (node.Enabled)
2278:             radioButtonSubscriptionEnabledOn.Checked = true;
2279:           else
2280:             radioButtonSubscriptionEnabledOff.Checked = true;
2281:           textBoxSubscriptionName.Text = node.SubscriptionName;
2282:           textBoxSubscriptionAcsUnitId.Text = node.AcsUnitId;
2283:           textBoxSubscriptionAcsAreaGroups.Text = node.AcsAreaGroups;
2284:           textBoxAeSubscriptionMessageFormat.Text = node.MessageFormat;
2285:           if (node.FilterEnabled)
2286:             radioButtonAeSubscriptionFilterEnabledOn.Checked = true;
2287:           else
2288:             radioButtonAeSubscriptionFilterEnabledOff.Checked = true;
2289:                     textBoxAeSubscriptionFilterSources.Text = node.Sources;
2290:           textBoxAeSubscriptionFilterAreas.Text = node.Areas;
2291:           if (node.EventType != null)
2292:             comboBoxAeSubscriptionFilterEventType.SelectedItem = node.EventType;
2293:           else
2294:             comboBoxAeSubscriptionFilterEventType.SelectedIndex = 0;
2295:           textBoxAeSubscriptionFilterEventCatIds.Text = node.CategoryIds;
2296:           textBoxAeSubscriptionFilterLowestSeverity.Text = node.LowestSeverity;
2297:           textBoxAeSubscriptionFilterHighestSeverity.Text = node.HighestSeverity;
2298:
2299:           // Populate combobox
2300:           OpcServerNode parent = (OpcServerNode) node.Parent;
2301:           comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.Items.Clear();
2302:           comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.Items.AddRange(parent.AcsAlarmCodes);
2303:           comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.SelectedItem = "NONE";
2304:
2305:           listBoxAeSubscriptionSeverityMappings.Items.Clear();
2306:           listBoxAeSubscriptionSeverityMappings.Items.AddRange(node.SeverityMappings);
2307:         }
2308:         else if (lastSelectedNode is OpcDaSubscriptionNode)
2309:         {
2310:           OpcDaSubscriptionNode node = (OpcDaSubscriptionNode) lastSelectedNode;
2311:           if (node.Enabled)
2312:             radioButtonSubscriptionEnabledOn.Checked = true;
2313:           else
2314:             radioButtonSubscriptionEnabledOff.Checked = true;
2315:           textBoxSubscriptionName.Text = node.SubscriptionName;
2316:           textBoxSubscriptionAcsUnitId.Text = node.AcsUnitId;
2317:           textBoxSubscriptionAcsAreaGroups.Text = node.AcsAreaGroups;
2318:           textBoxDaSubscriptionMessageFormat.Text = node.MessageFormat;
2319:           textBoxDaSubscriptionItem.Text = node.Item;
```

```
2320:
2321:           switch (node.AlarmStateType)
2322:           {
2323:             case AlarmStateTypes.BOOLEAN:
2324:               SwitchToAlarmTypeBooleanView();
2325:               comboBoxDaSubscriptionAlarmStateBooleanValue.SelectedItem = node.AlarmState;
2326:               break;
2327:             case AlarmStateTypes.SINGLE:
2328:               SwitchToAlarmTypeSingleValueView();
2329:               textBoxDaSubscriptionAlarmStateSingleValue.Text = node.AlarmState;
2330:               break;
2331:             case AlarmStateTypes.RANGE:
2332:               SwitchToAlarmTypeRangeView();
2333:               string low = node.AlarmState.Substring(0, node.AlarmState.IndexOf(" to ")).Trim();
2334:               string high = node.AlarmState.Substring(node.AlarmState.IndexOf(" to ")+4, node.AlarmState.Length-(node.
                    AlarmState.IndexOf(" to ")+4)).Trim();
2335:               textBoxDaSubscriptionAlarmStateRangeLow.Text = low;
2336:               textBoxDaSubscriptionAlarmStateRangeHigh.Text = high;
2337:               break;
2338:           }
2339:
2340:           // Populate combobox
2341:                     OpcServerNode parent = (OpcServerNode) node.Parent;
2342:           comboBoxDaSubscriptionAcsAlarmCode.Items.Clear();
2343:           comboBoxDaSubscriptionAcsAlarmCode.Items.AddRange(parent.AcsAlarmCodes);
2344:
2345:           if (node.AcsAlarmCode != null)
2346:             comboBoxDaSubscriptionAcsAlarmCode.SelectedItem = node.AcsAlarmCode;
2347:           else
2348:             comboBoxDaSubscriptionAcsAlarmCode.SelectedItem = "NONE";
2349:         }
2350:         SetDirty(tempDirty);
2351:       }
2352:     }
2353:
2354:     private void WorkerThreadFunction()
2355:     {
2356:       serviceActionInProcess = true;
2357:       UpdateProgressThread thread = new UpdateProgressThread(this, time);
2358:       thread.Run();
2359:     }
2360:
2361:     private void SetDirty(bool v)
2362:     {
2363:       dirty = v;
2364:       menuItemSave.Enabled = v;
2365:     }
```

```
2366:
2367:     private bool PromptForSaveIfNeeded()
2368:     {
2369:       if (dirty)
2370:       {
2371:         DialogResult res = MessageBox.Show("Save current configuration?", "Save changes", MessageBoxButtons.YesNoCancel);
2372:         if (res == DialogResult.Yes)
2373:         {
2374:           SaveValuesInModel();
2375:           if (currentConfigurationPath == null)
2376:           {
2377:             if (saveFileDialog.ShowDialog() == DialogResult.OK)
2378:             {
2379:               try
2380:               {
2381:                 controller.SaveConfiguration(saveFileDialog.FileName);
2382:                 currentConfigurationPath = saveFileDialog.FileName;
2383:                 SetDirty(false);
2384:                 return true;
2385:               }
2386:               catch (XmlException)
2387:               {
2388:                 MessageBox.Show("Error while writing configuration file!", "Error");
2389:               }
2390:             }
2391:             else
2392:               return false;
2393:           }
2394:           else
2395:           {
2396:             try
2397:             {
2398:               controller.SaveConfiguration(currentConfigurationPath);
2399:               SetDirty(false);
2400:             }
2401:             catch (XmlException)
2402:             {
2403:               MessageBox.Show("Error while writing configuration file!", "Error");
2404:             }
2405:             return true;
2406:           }
2407:         }
2408:         else if (res == DialogResult.No)
2409:         {
2410:           return true;
2411:         }
2412:         else
```

```
2413:         {
2414:           return false;
2415:         }
2416:       }
2417:       return true;
2418:     }
2419:
2420:     private void LoadNewLanguage(string path)
2421:     {
2422:       try
2423:       {
2424:         PropertyFileHelper pfh = new PropertyFileHelper(HOME_DIRECTORY + path);
2425:         this.Text = ReplacePatterns(pfh, pfh.GetProperty("PROGRAM_TITLE"));
2426:         AboutBox.ProgramName = this.Text;
2427:         DEFAULT_STATUS_BAR_TEXT = ReplacePatterns(pfh, this.Text);
2428:         statusBarPanelLeft.Text = DEFAULT_STATUS_BAR_TEXT;
2429:         menuItemFile.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_FILE"));
2430:         menuItemNew.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_FILE_NEW"));
2431:         menuItemOpen.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_FILE_OPEN"));
2432:         openFileDialog.Title = menuItemOpen.Text.Replace("&", "");
2433:         menuItemSave.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_FILE_SAVE"));
2434:         menuItemSaveAs.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_FILE_SAVE_AS"));
2435:         saveFileDialog.Title = menuItemSaveAs.Text.Replace("&", "");
2436:         menuItemExit.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_FILE_EXIT"));
2437:         menuItemLanguage.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_LANGUAGE"));
2438:         menuItemHelp.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_HELP"));
2439:         menuItemAbout.Text = ReplacePatterns(pfh, pfh.GetProperty("MENU_HELP_ABOUT"));
2440:         AboutBox.Title = menuItemAbout.Text.Replace("&", "");
2441:         groupBoxServerList.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_SERVERS"));
2442:         treeViewServers.Nodes[0].Text = groupBoxServerList.Text;
2443:         groupBoxInformation.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_INFORMATION"));
2444:         groupBoxServer.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_SERVER"));
2445:         groupBoxServerAcsAlarmCodes.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_ACS_ALARM_CODES"));
2446:         groupBoxSubscription.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_SUBSCRIPTION"));
2447:         groupBoxFilter.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_FILTER"));
2448:         groupBoxSeverityMappings.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_SEVERITY_MAPPINGS"));
2449:         groupBoxServiceInformation.Text = ReplacePatterns(pfh, pfh.GetProperty("GROUPBOX_SERVICE_INFORMATION"));
2450:         buttonRemoveSelected.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_REMOVE_SELECTED"));
2451:         buttonAddServer.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_ADD_SERVER"));
2452:         buttonAddAeSubscription.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_ADD_AE_SUBSCRIPTION"));
2453:         buttonAddDaSubscription.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_ADD_DA_SUBSCRIPTION"));
2454:         buttonServerServerNameSelect.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_SELECT"));
2455:         buttonAeSubscriptionFilterSourcesSelect.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_SELECT"));
2456:         buttonAeSubscriptionFilterAreasSelect.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_SELECT"));
2457:         buttonAeSubscriptionFilterEventCatIdsSelect.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_SELECT"));
2458:         buttonDaSubscriptionItemSelect.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_SELECT"));
2459:         buttonServerAcsAlarmCodesAddAlarmCode.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_ADD_ALARM_CODE"));
```

```
2460:         buttonAeSubscriptionSource.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_SOURCE"));
2461:         buttonAeSubscriptionSeverity.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_SEVERITY"));
2462:         buttonAeSubscriptionMessage.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_MESSAGE"));
2463:         buttonAeSubscriptionCondition.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_CONDITION"));
2464:         buttonAeSubscriptionServer.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_SERVER"));
2465:         buttonAeSubscriptionSubscription.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_SUBSCRIPTION"));
2466:         buttonAeSubscriptionFormula.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_FORMULA"));
2467:         buttonDaSubscriptionValue.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_VALUE"));
2468:         buttonDaSubscriptionItemName.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_ITEM_NAME"));
2469:         buttonDaSubscriptionItemPath.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_ITEM_PATH"));
2470:         buttonDaSubscriptionResult.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_RESULT"));
2471:         buttonDaSubscriptionServer.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_SERVER"));
2472:         buttonDaSubscriptionSubscription.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_SUBSCRIPTION"));
2473:         buttonDaSubscriptionFormula.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_MESSAGE_FORMAT_FORMULA"));
2474:         buttonAeSubscriptionSeverityMappingAddMapping.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_ADD_MAPPING"));
2475:         buttonServiceStartService.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_START_SERVICE"));
2476:               buttonServiceStopService.Text = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_STOP_SERVICE"));
2477:
2478:         SelectItemBox.BUTTON_LIST_SERVERS_TEXT = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_SELECT_ITEM_BOX_LIST_SERVERS")
               );
2479:         SelectItemBox.BUTTON_OK_TEXT = ReplacePatterns(pfh, pfh.GetProperty("BUTTON_SELECT_ITEM_BOX_OK"));
2480:
2481:         tabControl.TabPages[0].Text = ReplacePatterns(pfh, pfh.GetProperty("TAB_CONFIGURATION"));
2482:         tabControl.TabPages[1].Text = ReplacePatterns(pfh, pfh.GetProperty("TAB_SERVICE"));
2483:         radioButtonSubscriptionEnabledOn.Text = ReplacePatterns(pfh, pfh.GetProperty("RADIO_BUTTON_ON"));
2484:         radioButtonAeSubscriptionFilterEnabledOn.Text = ReplacePatterns(pfh, pfh.GetProperty("RADIO_BUTTON_ON"));
2485:         radioButtonSubscriptionEnabledOff.Text = ReplacePatterns(pfh, pfh.GetProperty("RADIO_BUTTON_OFF"));
2486:         radioButtonAeSubscriptionFilterEnabledOff.Text = ReplacePatterns(pfh, pfh.GetProperty("RADIO_BUTTON_OFF"));
2487:         radioButtonDaSubscriptionAlarmStateTypeBoolean.Text = ReplacePatterns(pfh, pfh.GetProperty("RADIO_BUTTON_BOOLEAN"));
2488:         radioButtonDaSubscriptionAlarmStateTypeSingleValue.Text = ReplacePatterns(pfh, pfh.GetProperty("
               RADIO_BUTTON_SINGLE_VALUE"));
2489:               radioButtonDaSubscriptionAlarmStateTypeRange.Text = ReplacePatterns(pfh, pfh.GetProperty("RADIO_BUTTON_RANGE
                 "));
2490:
2491:         labelInformation.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_INFORMATION_MESSAGE"));
2492:         labelServerOpcServerName.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVER_OPC_SERVER_NAME"));
2493:               labelServerOpcServerHost.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVER_OPC_SERVER_HOST"));
2494:         labelServerDefaultUnitId.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVER_ACS_DEFAULT_UNIT_ID"));
2495:         labelServerDefaultAreaGroups.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVER_ACS_DEFAULT_AREA_GROUPS"));
2496:         labelServerConnectionType.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVER_CONNECTION_TYPE"));
2497:         labelServerAcsAlarmCodesAlarmCode.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVER_ACS_ALARM_CODE"));
2498:         labelSubscriptionEnabled.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_ENABLED"));
2499:         labelSubscriptionName.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_SUBSCRIPTION_NAME"));
2500:         labelSubscriptionAcsUnitId.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_ACS_UNIT_ID"));
2501:         labelSubscriptionAcsAreaGroups.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_ACS_AREA_GROUPS"));
2502:         labelAeSubscriptionMessageFormat.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_MESSAGE_FORMAT"));
2503:         labelDaSubscriptionMessageFormat.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_MESSAGE_FORMAT"));
```

```
2504:         labelAeSubscriptionFilterEnabled.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_FILTER_ENABLED"));
2505:         labelAeSubscriptionFilterSources.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_FILTER_SOURCES"));
2506:         labelAeSubscriptionFilterAreas.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_FILTER_AREAS"));
2507:         labelAeSubscriptionFilterEventType.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_FILTER_EVENT_TYPE
               "));
2508:         labelAeSubscriptionFilterEventCatIds.Text = ReplacePatterns(pfh, pfh.GetProperty("
               LABEL_SUBSCRIPTION_FILTER_EVENT_CAT_IDS"));
2509:         labelAeSubscriptionFilterHighestSeverity.Text = ReplacePatterns(pfh, pfh.GetProperty("
               LABEL_SUBSCRIPTION_FILTER_HIGHEST_SEVERITY"));
2510:         labelAeSubscriptionFilterHighestSeverityInfo.Text = ReplacePatterns(pfh, pfh.GetProperty("
               LABEL_SUBSCRIPTION_FILTER_HIGHEST_SEVERITY_INFO"));
2511:         labelAeSubscriptionFilterLowestSeverity.Text = ReplacePatterns(pfh, pfh.GetProperty("
               LABEL_SUBSCRIPTION_FILTER_LOWEST_SEVERITY"));
2512:         labelAeSubscriptionFilterLowestSeverityInfo.Text = ReplacePatterns(pfh, pfh.GetProperty("
               LABEL_SUBSCRIPTION_FILTER_LOWEST_SEVERITY_INFO"));
2513:         labelAeSubscriptionSeverityMappingsRange.Text = ReplacePatterns(pfh, pfh.GetProperty("
               LABEL_SUBSCRIPTION_SEVERITY_MAPPING_RANGE"));
2514:         labelAeSubscriptionSeverityMappingsAcsAlarmCode.Text = ReplacePatterns(pfh, pfh.GetProperty("
               LABEL_SUBSCRIPTION_SEVERITY_MAPPING_ACS_ALARM_CODE"));
2515:         labelDaSubscriptionItem.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_ITEM"));
2516:         labelDaSubscriptionAlarmState.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_ALARM_STATE"));
2517:         labelDaSubscriptionAcsAlarmCode.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SUBSCRIPTION_ACS_ALARM_CODE"));
2518:                 labelServiceDisplayName.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_DISPLAY_NAME"));
2519:         labelServiceServiceName.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_SERVICE_NAME"));
2520:         labelServiceServiceType.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_SERVICE_TYPE"));
2521:         labelServiceServiceStatus.Text = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_SERVICE_STATUS"));
2522:
2523:         SERVICE_RUNNING = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_RUNNING"));
2524:         SERVICE_STOPPED = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_STOPPED"));
2525:         SERVICE_FETCHING_INFO = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_FETCHING_INFO"));
2526:         SERVICE_NOT_INSTALLED = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SERVICE_NOT_INSTALLED"));
2527:
2528:         SelectItemBox.TITLE_SERVER = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SELECT_ITEM_BOX_TITLE_SERVER"));
2529:         SelectItemBox.TITLE_OTHER = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SELECT_ITEM_BOX_TITLE_OTHER"));
2530:         SelectItemBox.LABEL_HOST = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SELECT_ITEM_BOX_HOST"));
2531:         SelectItemBox.LABEL_SOURCES = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SELECT_ITEM_BOX_SOURCES"));
2532:         SelectItemBox.LABEL_AREAS = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SELECT_ITEM_BOX_AREAS"));
2533:         SelectItemBox.LABEL_EVENT_CAT_IDS = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SELECT_ITEM_BOX_EVENT_CAT_IDS"));
2534:         SelectItemBox.LABEL_ITEM = ReplacePatterns(pfh, pfh.GetProperty("LABEL_SELECT_ITEM_BOX_ITEM"));
2535:
2536:         Controller.DEFAULT_SERVER_SERVER_NAME = ReplacePatterns(pfh, pfh.GetProperty("DEFAULT_SERVER_SERVER_NAME"));
2537:         Controller.DEFAULT_SUBSCRIPTION_SUBSCRIPTION_NAME = ReplacePatterns(pfh, pfh.GetProperty("
               DEFAULT_SUBSCRIPTION_SUBSCRIPTION_NAME"));
2538:               Controller.DEFAULT_AE_SUBSCRIPTION_MESSAGE_FORMAT = ReplacePatterns(pfh, pfh.GetProperty("
                 DEFAULT_AE_SUBSCRIPTION_MESSAGE_FORMAT"));
2539:         Controller.DEFAULT_DA_SUBSCRIPTION_MESSAGE_FORMAT = ReplacePatterns(pfh, pfh.GetProperty("
               DEFAULT_DA_SUBSCRIPTION_MESSAGE_FORMAT"));
```

```
2540:         if (tabControl.SelectedIndex==1)
2541:           tabControl_SelectedIndexChanged(null, null);
2542:
2543:         STATUS_BAR_REMOVE_SERVER = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_REMOVE_SERVER"));
2544:         STATUS_BAR_REMOVE_SUBSCRIPTION = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_REMOVE_SUBSCRIPTION"));
2545:         STATUS_BAR_ADD_SERVER = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_SERVER"));
2546:         STATUS_BAR_ADD_AE_SUBSCRIPTION = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_AE_SUBSCRIPTION"));
2547:         STATUS_BAR_ADD_DA_SUBSCRIPTION = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_DA_SUBSCRIPTION"));
2548:         STATUS_BAR_ADD_ACS_ALARM_CODE = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_ACS_ALARM_CODE"));
2549:         STATUS_BAR_ADD_SEVERITY_MAPPING = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_SEVERITY_MAPPING"));
2550:         STATUS_BAR_ADD_MESSAGE_FORMAT_SOURCE = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_MESSAGE_FORMAT_SOURCE"))
               ;
2551:         STATUS_BAR_ADD_MESSAGE_FORMAT_SEVERITY = ReplacePatterns(pfh, pfh.GetProperty("
               STATUS_BAR_ADD_MESSAGE_FORMAT_SEVERITY"));
2552:         STATUS_BAR_ADD_MESSAGE_FORMAT_MESSAGE = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_MESSAGE_FORMAT_MESSAGE"
               ));
2553:         STATUS_BAR_ADD_MESSAGE_FORMAT_CONDITION = ReplacePatterns(pfh, pfh.GetProperty("
               STATUS_BAR_ADD_MESSAGE_FORMAT_CONDITION"));
2554:         STATUS_BAR_ADD_MESSAGE_FORMAT_VALUE = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_MESSAGE_FORMAT_VALUE"));
2555:         STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_NAME = ReplacePatterns(pfh, pfh.GetProperty("
               STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_NAME"));
2556:         STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_PATH = ReplacePatterns(pfh, pfh.GetProperty("
               STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_PATH"));
2557:         STATUS_BAR_ADD_MESSAGE_FORMAT_RESULT = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_MESSAGE_FORMAT_RESULT"))
               ;
2558:         STATUS_BAR_ADD_MESSAGE_FORMAT_SERVER = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_MESSAGE_FORMAT_SERVER"))
               ;
2559:         STATUS_BAR_ADD_MESSAGE_FORMAT_SUBSCRIPTION = ReplacePatterns(pfh, pfh.GetProperty("
               STATUS_BAR_ADD_MESSAGE_FORMAT_SUBSCRIPTION"));
2560:         STATUS_BAR_ADD_MESSAGE_FORMAT_FORMULA = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_ADD_MESSAGE_FORMAT_FORMULA"
               ));
2561:         STATUS_BAR_SELECT_SERVER = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_SELECT_SERVER"));
2562:         STATUS_BAR_SELECT_SOURCES = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_SELECT_SOURCES"));
2563:         STATUS_BAR_SELECT_AREAS = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_SELECT_AREAS"));
2564:         STATUS_BAR_SELECT_EVENT_CAT_IDS = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_SELECT_EVENT_CAT_IDS"));
2565:         STATUS_BAR_SELECT_ITEM = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_SELECT_ITEM"));
2566:         STATUS_BAR_START_SERVICE = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_START_SERVICE"));
2567:         STATUS_BAR_STOP_SERVICE = ReplacePatterns(pfh, pfh.GetProperty("STATUS_BAR_STOP_SERVICE"));
2568:
2569:         ERROR_TITLE = ReplacePatterns(pfh, pfh.GetProperty("ERROR_TITLE"));
2570:         SelectItemBox.ERROR_TITLE = ERROR_TITLE;
2571:         SelectItemBox.ERROR_MESSAGE_SELECT_ITEM = ReplacePatterns(pfh, pfh.GetProperty("ERROR_MESSAGE_SELECT_ITEM"));
2572:         ERROR_MESSAGE_OPEN_FILE = ReplacePatterns(pfh, pfh.GetProperty("ERROR_MESSAGE_OPEN_FILE"));
2573:         ERROR_MESSAGE_SAVE_FILE = ReplacePatterns(pfh, pfh.GetProperty("ERROR_MESSAGE_SAVE_FILE"));
2574:
2575:         if (path.Equals(LANGUAGE_ENGLISH))
2576:         {
```

```
2577:           menuItemDanish.Checked = false;
2578:           menuItemEnglish.Checked = true;
2579:         }
2580:         else if (path.Equals(LANGUAGE_DANISH))
2581:         {
2582:           menuItemEnglish.Checked = false;
2583:           menuItemDanish.Checked = true;
2584:         }
2585:
2586:         Update();
2587:       }
2588:       catch (Exception e)
2589:       {
2590:         MessageBox.Show(e.Message, "Error");
2591:       }
2592:     }
2593:
2594:     private string ReplacePatterns(PropertyFileHelper pfh, string text)
2595:     {
2596:       int index = text.IndexOf("%");
2597:       int index2;
2598:
2599:       if (index != -1)
2600:       {
2601:         // Percent sign
2602:         if (text.Substring(index, 2).Equals("%%"))
2603:         {
2604:           return text.Substring(0, index+1) + ReplacePatterns(pfh, text.Substring(index+2));
2605:         }
2606:         // pattern
2607:         else
2608:         {
2609:           index2 = text.IndexOf("%", index+1);
2610:           if (index2 != -1)
2611:           {
2612:             return text.Substring(0, index) + pfh.GetProperty(text.Substring(index+1, (index2-index)-1)) + ReplacePatterns(
                    pfh, text.Substring(index2+1));
2613:           }
2614:           else
2615:           {
2616:             throw new Exception("Invalid value");
2617:           }
2618:         }
2619:       }
2620:       return text;
2621:     }
2622:
```

```
2623:     #endregion
2624:
2625:     #region Event Handlers
2626:     //-----------------------------------------------------------------------------
2627:     // Event Handlers
2628:     //
2629:     // Methods which handle incoming events (mouse clicks, etc.)
2630:     //-----------------------------------------------------------------------------
2631:
2632:     #region Button Clicks
2633:
2634:     private void buttonRemoveSelected_Click(object sender, System.EventArgs e)
2635:     {
2636:       SaveValuesInModel();
2637:       if (treeViewServers.SelectedNode != treeViewServers.Nodes[0])
2638:       {
2639:         treeViewServers.Focus();
2640:         String msg = "";
2641:         if (treeViewServers.SelectedNode is OpcServerNode)
2642:           msg = "Really remove server '" + treeViewServers.SelectedNode.Text + "' and all its subscriptions?";
2643:         else if (treeViewServers.SelectedNode is OpcSubscriptionNode)
2644:           msg = "Really remove subscription '" + treeViewServers.SelectedNode.Text + "'?";
2645:         DialogResult res = MessageBox.Show(msg, "Remove...", MessageBoxButtons.YesNo);
2646:         if (res == DialogResult.Yes)
2647:         {
2648:           TreeNode parent = treeViewServers.SelectedNode.Parent;
2649:           int index = parent.Nodes.Count - 1;
2650:           bool lastSubscription = false;
2651:           if (parent.Nodes.IndexOf(treeViewServers.SelectedNode)==index)
2652:           {
2653:             index --;
2654:             lastSubscription = true;
2655:           }
2656:           if (!lastSubscription)
2657:           {
2658:             if (treeViewServers.SelectedNode is OpcServerNode)
2659:               controller.RemoveServer((OpcServerNode) treeViewServers.SelectedNode);
2660:             else if (treeViewServers.SelectedNode is OpcSubscriptionNode)
2661:               controller.RemoveSubscription((OpcSubscriptionNode) treeViewServers.SelectedNode, (OpcServerNode) parent);
2662:           }
2663:           else
2664:           {
2665:             if (treeViewServers.SelectedNode is OpcServerNode)
2666:               controller.RemoveServer((OpcServerNode) treeViewServers.SelectedNode);
2667:             else if (treeViewServers.SelectedNode is OpcSubscriptionNode)
2668:               controller.RemoveSubscription((OpcSubscriptionNode) treeViewServers.SelectedNode, (OpcServerNode) parent);
2669:             if (index > 0)
```

```
2670:                 treeViewServers.SelectedNode = parent.Nodes[index];
2671:               treeViewServers.Focus();
2672:             }
2673:             SetDirty(true);
2674:           }
2675:         }
2676:       }
2677:
2678:       private void buttonAddServer_Click(object sender, System.EventArgs e)
2679:       {
2680:         OpcServerNode server = controller.AddServer();
2681:         if (server != null)
2682:         {
2683:           treeViewServers.SelectedNode = server;
2684:           SwitchToServerView();
2685:           textBoxServerOpcServerName.Focus();
2686:           textBoxServerOpcServerName.SelectAll();
2687:           SetDirty(true);
2688:         }
2689:       }
2690:
2691:       private void buttonAddAeSubscription_Click(object sender, System.EventArgs e)
2692:       {
2693:         OpcServerNode server = GetSelectedServer();
2694:         if (server != null)
2695:         {
2696:           OpcSubscriptionNode subscription = controller.AddAeSubscription(server);
2697:           if (subscription != null)
2698:           {
2699:             treeViewServers.SelectedNode = subscription;
2700:             SwitchToAeSubscriptionView();
2701:             textBoxSubscriptionName.Focus();
2702:             textBoxSubscriptionName.SelectAll();
2703:             SetDirty(true);
2704:           }
2705:
2706:         }
2707:       }
2708:
2709:       private void buttonAddDaSubscription_Click(object sender, System.EventArgs e)
2710:       {
2711:         OpcServerNode server = GetSelectedServer();
2712:         if (server != null)
2713:         {
2714:           OpcSubscriptionNode subscription = controller.AddDaSubscription(server);
2715:           if (subscription != null)
2716:           {
```

```
2717:             treeViewServers.SelectedNode = subscription;
2718:             SwitchToDaSubscriptionView();
2719:             textBoxSubscriptionName.Focus();
2720:             textBoxSubscriptionName.SelectAll();
2721:             SetDirty(true);
2722:           }
2723:       }
2724:     }
2725:
2726:     private void buttonServerAcsAlarmCodesAddAlarmCode_Click(object sender, System.EventArgs e)
2727:     {
2728:       if (!textBoxServerAcsAlarmCodesAlarmCode.Text.Equals(""))
2729:       {
2730:         listBoxServerAcsAlarmCodes.Items.Add(textBoxServerAcsAlarmCodesAlarmCode.Text);
2731:         textBoxServerAcsAlarmCodesAlarmCode.Focus();
2732:         textBoxServerAcsAlarmCodesAlarmCode.SelectAll();
2733:         SetDirty(true);
2734:       }
2735:     }
2736:
2737:     private void buttonAeSubscriptionSeverityMappingAddMapping_Click(object sender, System.EventArgs e)
2738:     {
2739:       if (controller.ValidOpcRange(textBoxAeSubscriptionSeverityMappingsRange.Text))
2740:       {
2741:         OpcSeverityMapping mapping = new OpcSeverityMapping(textBoxAeSubscriptionSeverityMappingsRange.Text,
2742:           comboBoxAeSubscriptionSeverityMappingsAcsAlarmCode.SelectedItem.ToString());
2743:         listBoxAeSubscriptionSeverityMappings.Items.Add(mapping);
2744:         textBoxAeSubscriptionSeverityMappingsRange.Focus();
2745:         textBoxAeSubscriptionSeverityMappingsRange.SelectAll();
2746:         SetDirty(true);
2747:       }
2748:       else
2749:       {
2750:         MessageBox.Show("Please enter a valid severity range!", "Error");
2751:         textBoxAeSubscriptionSeverityMappingsRange.Focus();
2752:         textBoxAeSubscriptionSeverityMappingsRange.SelectAll();
2753:       }
2754:     }
2755:
2756:     private void buttonAeSubscriptionMessageFormatButton_Click(object sender, System.EventArgs e)
2757:     {
2758:       int index = textBoxAeSubscriptionMessageFormat.SelectionStart;
2759:       int cursorPos;
2760:       bool normalSelect = true;
2761:       String text = textBoxAeSubscriptionMessageFormat.Text.Substring(0, index);
2762:
2763:       if (sender.Equals(buttonAeSubscriptionSource))
```

```
2764:         text += MESSAGE_FORMAT_SOURCE;
2765:       else if (sender.Equals(buttonAeSubscriptionSeverity))
2766:         text += MESSAGE_FORMAT_SEVERITY;
2767:       else if (sender.Equals(buttonAeSubscriptionMessage))
2768:         text += MESSAGE_FORMAT_MESSAGE;
2769:       else if (sender.Equals(buttonAeSubscriptionCondition))
2770:         text += MESSAGE_FORMAT_CONDITION;
2771:       else if (sender.Equals(buttonAeSubscriptionServer))
2772:         text += MESSAGE_FORMAT_SERVER;
2773:       else if (sender.Equals(buttonAeSubscriptionSubscription))
2774:         text += MESSAGE_FORMAT_SUBSCRIPTION;
2775:       else if (sender.Equals(buttonAeSubscriptionFormula))
2776:       {
2777:         text += MESSAGE_FORMAT_FORMULA;
2778:         normalSelect = false;
2779:       }
2780:
2781:       cursorPos = text.Length;
2782:       text += textBoxAeSubscriptionMessageFormat.Text.Substring(index);
2783:       textBoxAeSubscriptionMessageFormat.Text = text;
2784:
2785:       if (normalSelect)
2786:       {
2787:         textBoxAeSubscriptionMessageFormat.SelectionStart = cursorPos;
2788:       }
2789:       else
2790:       {
2791:         textBoxAeSubscriptionMessageFormat.SelectionStart = cursorPos - 2;
2792:       }
2793:       textBoxAeSubscriptionMessageFormat.Focus();
2794:       SetDirty(true);
2795:     }
2796:
2797:     private void buttonDaSubscriptionMessageFormatButton_Click(object sender, System.EventArgs e)
2798:     {
2799:       int index = textBoxDaSubscriptionMessageFormat.SelectionStart;
2800:       int toBeDeleted = textBoxDaSubscriptionMessageFormat.SelectionLength;
2801:       int cursorPos;
2802:       bool normalSelect = true;
2803:       String text = textBoxDaSubscriptionMessageFormat.Text.Substring(0, index);
2804:
2805:       if (sender.Equals(buttonDaSubscriptionValue))
2806:         text += MESSAGE_FORMAT_VALUE;
2807:       else if (sender.Equals(buttonDaSubscriptionItemName))
2808:         text += MESSAGE_FORMAT_ITEM_NAME;
2809:       else if (sender.Equals(buttonDaSubscriptionItemPath))
2810:         text += MESSAGE_FORMAT_ITEM_PATH;
```

```
2811:       else if (sender.Equals(buttonDaSubscriptionResult))
2812:         text += MESSAGE_FORMAT_RESULT;
2813:       else if (sender.Equals(buttonDaSubscriptionServer))
2814:         text += MESSAGE_FORMAT_SERVER;
2815:       else if (sender.Equals(buttonDaSubscriptionSubscription))
2816:         text += MESSAGE_FORMAT_SUBSCRIPTION;
2817:       else if (sender.Equals(buttonDaSubscriptionFormula))
2818:       {
2819:         text += MESSAGE_FORMAT_FORMULA;
2820:         normalSelect = false;
2821:       }
2822:
2823:       cursorPos = text.Length;
2824:       text += textBoxDaSubscriptionMessageFormat.Text.Substring(index+toBeDeleted);
2825:       textBoxDaSubscriptionMessageFormat.Text = text;
2826:
2827:       if (normalSelect)
2828:       {
2829:         textBoxDaSubscriptionMessageFormat.SelectionStart = cursorPos;
2830:       }
2831:       else
2832:       {
2833:         textBoxDaSubscriptionMessageFormat.SelectionStart = cursorPos - 2;
2834:       }
2835:       textBoxDaSubscriptionMessageFormat.Focus();
2836:       SetDirty(true);
2837:     }
2838:
2839:     private void buttonServiceStartService_Click(object sender, System.EventArgs e)
2840:     {
2841:       time = 5;
2842:       progressBarService.Visible = true;
2843:       Thread thread = new Thread(new ThreadStart(this.WorkerThreadFunction));
2844:       buttonServiceStartService.Enabled = false;
2845:       lastAction = ServiceAction.Started;
2846:       thread.Start();
2847:       ServiceController sc = new ServiceController(SERVICE_NAME);
2848:       try
2849:       {
2850:         sc.Start();
2851:         labelServiceStatus.Text = "Service is starting...";
2852:         Refresh();
2853:       }
2854:       catch (InvalidOperationException)
2855:       {
2856:         MessageBox.Show("An error occurred while starting the service!", "Error");
2857:         thread = null;
```

```
2858:         progressBarService.Visible = false;
2859:         progressBarService.Value = 0;
2860:       }
2861:     }
2862:
2863:     private void buttonServiceStopService_Click(object sender, System.EventArgs e)
2864:     {
2865:       time = 20;
2866:       progressBarService.Visible = true;
2867:       Thread thread = new Thread(new ThreadStart(this.WorkerThreadFunction));
2868:       buttonServiceStopService.Enabled = false;
2869:       lastAction = ServiceAction.Stopped;
2870:       thread.Start();
2871:       ServiceController sc = new ServiceController(SERVICE_NAME);
2872:       try
2873:       {
2874:         sc.Stop();
2875:         labelServiceStatus.Text = "Service is stopping...";
2876:         Refresh();
2877:       }
2878:       catch (InvalidOperationException)
2879:       {
2880:         MessageBox.Show("An error occurred while stopping the service!", "Error");
2881:         thread = null;
2882:         progressBarService.Visible = false;
2883:         progressBarService.Value = 0;
2884:       }
2885:     }
2886:
2887:     private void buttonDaSubscriptionItemSelect_Click(object sender, System.EventArgs e)
2888:     {
2889:       OpcServerNode server = (OpcServerNode) lastSelectedNode.Parent;
2890:       SelectItemBox sib = new SelectItemBox(this, server.OpcServerName, server.OpcServerHost, ((server.UseCom) ?
            ConnectionType.COM : ConnectionType.XML), Item.ITEM, null);
2891:       if (sib.InitSelectItemBox())
2892:         sib.ShowDialog(this);
2893:       textBoxDaSubscriptionItem.Focus();
2894:       textBoxDaSubscriptionItem.SelectAll();
2895:     }
2896:
2897:     private void buttonAeSubscriptionFilterSourcesSelect_Click(object sender, System.EventArgs e)
2898:     {
2899:       OpcServerNode server = (OpcServerNode) lastSelectedNode.Parent;
2900:       SelectItemBox sib = new SelectItemBox(this, server.OpcServerName, server.OpcServerHost, ((server.UseCom) ?
            ConnectionType.COM : ConnectionType.XML), Item.SOURCES, null);
2901:       if (sib.InitSelectItemBox())
2902:         sib.ShowDialog(this);
```

```
2903:       textBoxAeSubscriptionFilterSources.Focus();
2904:       textBoxAeSubscriptionFilterSources.SelectAll();
2905:     }
2906:
2907:     private void buttonAeSubscriptionFilterAreasSelect_Click(object sender, System.EventArgs e)
2908:     {
2909:       OpcServerNode server = (OpcServerNode) lastSelectedNode.Parent;
2910:       SelectItemBox sib = new SelectItemBox(this, server.OpcServerName, server.OpcServerHost, ((server.UseCom) ?
            ConnectionType.COM : ConnectionType.XML), Item.AREAS, null);
2911:       if (sib.InitSelectItemBox())
2912:         sib.ShowDialog(this);
2913:       textBoxAeSubscriptionFilterAreas.Focus();
2914:       textBoxAeSubscriptionFilterAreas.SelectAll();
2915:     }
2916:
2917:     private void buttonAeSubscriptionFilterEventCatIdsSelect_Click(object sender, System.EventArgs e)
2918:     {
2919:       OpcServerNode server = (OpcServerNode) lastSelectedNode.Parent;
2920:       SelectItemBox sib = new SelectItemBox(this, server.OpcServerName, server.OpcServerHost, ((server.UseCom) ?
            ConnectionType.COM : ConnectionType.XML), Item.CAT_IDS, null);
2921:       if (sib.InitSelectItemBox())
2922:         sib.ShowDialog(this);
2923:       textBoxAeSubscriptionFilterEventCatIds.Focus();
2924:       textBoxAeSubscriptionFilterEventCatIds.SelectAll();
2925:     }
2926:
2927:     private void buttonServerServerNameSelect_Click(object sender, System.EventArgs e)
2928:     {
2929:       OpcServerNode server = (OpcServerNode) lastSelectedNode;
2930:       SelectItemBox sib = new SelectItemBox(this, server.OpcServerName, server.OpcServerHost, ((server.UseCom) ?
            ConnectionType.COM : ConnectionType.XML), Item.SERVER, textBoxServerOpcServerHost.Text);
2931:       if (sib.InitSelectItemBox())
2932:         sib.ShowDialog(this);
2933:       textBoxServerOpcServerName.Focus();
2934:       textBoxServerOpcServerName.SelectAll();
2935:     }
2936:
2937:     #endregion
2938:
2939:     #region List/Tree-View Event Handlers
2940:
2941:     private void treeViewServers_AfterSelect(object sender, System.Windows.Forms.TreeViewEventArgs e)
2942:     {
2943:       SaveValuesInModel();
2944:       if (treeViewServers.SelectedNode is OpcServerNode)
2945:         SwitchToServerView();
2946:       else if (treeViewServers.SelectedNode is OpcAeSubscriptionNode)
```

```
2947:         SwitchToAeSubscriptionView();
2948:       else if (treeViewServers.SelectedNode is OpcDaSubscriptionNode)
2949:         SwitchToDaSubscriptionView();
2950:       else
2951:         SwitchToInformationView();
2952:
2953:       lastSelectedNode = treeViewServers.SelectedNode;
2954:       LoadValuesFromModel();
2955:     }
2956:
2957:     private void listBoxServerAcsAlarmCodes_KeyUp(object sender, System.Windows.Forms.KeyEventArgs e)
2958:     {
2959:       // If DEL is pressed while an item is selected
2960:       if (e.KeyCode==Keys.Delete && listBoxServerAcsAlarmCodes.SelectedItem != null && !listBoxServerAcsAlarmCodes.
          SelectedItem.Equals("NONE"))
2961:       {
2962:         string selectedItem = (string) listBoxServerAcsAlarmCodes.SelectedItem;
2963:         int index = listBoxServerAcsAlarmCodes.SelectedIndex;
2964:         if (listBoxServerAcsAlarmCodes.Items.Count==(index+1))
2965:           listBoxServerAcsAlarmCodes.Items.RemoveAt(index--);
2966:         else
2967:           listBoxServerAcsAlarmCodes.Items.RemoveAt(index);
2968:
2969:         // Set all DA subscriptions using the deleted value for alarm state to NONE instead
2970:         OpcServerNode server = (OpcServerNode) lastSelectedNode;
2971:         foreach (OpcSubscriptionNode sub in server.Nodes)
2972:         {
2973:           if (sub is OpcDaSubscriptionNode)
2974:           {
2975:             OpcDaSubscriptionNode daSub = (OpcDaSubscriptionNode) sub;
2976:             if (daSub.AcsAlarmCode.Equals(selectedItem))
2977:               daSub.AcsAlarmCode = "NONE";
2978:           }
2979:         }
2980:
2981:         listBoxServerAcsAlarmCodes.SelectedIndex = index;
2982:         listBoxServerAcsAlarmCodes.Focus();
2983:       }
2984:     }
2985:
2986:     private void listBoxAeSubscriptionSeverityMappings_KeyUp(object sender, System.Windows.Forms.KeyEventArgs e)
2987:     {
2988:       // If DEL is pressed while an item is selected
2989:       if (e.KeyCode==Keys.Delete && listBoxAeSubscriptionSeverityMappings.SelectedItem != null)
2990:       {
2991:         int index = listBoxAeSubscriptionSeverityMappings.SelectedIndex;
2992:         if (listBoxAeSubscriptionSeverityMappings.Items.Count==(index+1))
```

```
2993:           listBoxAeSubscriptionSeverityMappings.Items.RemoveAt(index--);
2994:         else
2995:           listBoxAeSubscriptionSeverityMappings.Items.RemoveAt(index);
2996:
2997:         listBoxAeSubscriptionSeverityMappings.SelectedIndex = index;
2998:         listBoxAeSubscriptionSeverityMappings.Focus();
2999:       }
3000:     }
3001:
3002:     #endregion
3003:
3004:     #region Tab Control Event Handlers
3005:
3006:     private void tabControl_SelectedIndexChanged(object sender, System.EventArgs e)
3007:     {
3008:       if (tabControl.SelectedIndex == 1 && !serviceActionInProcess)
3009:       {
3010:         labelServiceStatus.Text = SERVICE_FETCHING_INFO;
3011:         tabControl.Refresh();
3012:
3013:         // Check the state of the service
3014:         ServiceController sc = new ServiceController(SERVICE_NAME);
3015:         try
3016:         {
3017:           labelServiceDisplayNameData.Text = sc.DisplayName;
3018:           labelServiceServiceNameData.Text = sc.ServiceName;
3019:           labelServiceServiceTypeData.Text = sc.ServiceType.ToString();
3020:           labelServiceServiceStatusData.Text = sc.Status.ToString();
3021:           if (sc.Status == ServiceControllerStatus.Running)
3022:           {
3023:             labelServiceStatus.Text = SERVICE_RUNNING;
3024:             buttonServiceStartService.Enabled = false;
3025:             buttonServiceStopService.Enabled = true;
3026:           }
3027:           else
3028:           {
3029:             labelServiceStatus.Text = SERVICE_STOPPED;
3030:             buttonServiceStartService.Enabled = true;
3031:             buttonServiceStopService.Enabled = false;
3032:           }
3033:         }
3034:         catch (InvalidOperationException)
3035:         {
3036:           labelServiceStatus.Text = SERVICE_NOT_INSTALLED;
3037:           labelServiceDisplayNameData.Text = "";
3038:           labelServiceServiceNameData.Text = "";
3039:           labelServiceServiceTypeData.Text = "";
```

```
3040:             labelServiceServiceStatusData.Text = "";
3041:             buttonServiceStartService.Enabled = false;
3042:             buttonServiceStopService.Enabled = false;
3043:           }
3044:           Refresh();
3045:         }
3046:       }
3047:
3048:       #endregion
3049:
3050:       #region Menu Event Handlers
3051:
3052:       private void menuItemNew_Click(object sender, System.EventArgs e)
3053:       {
3054:         SaveValuesInModel();
3055:
3056:         if (PromptForSaveIfNeeded())
3057:         {
3058:           textBoxServerOpcServerName.CausesValidation = false;
3059:           textBoxSubscriptionName.CausesValidation = false;
3060:           controller.ClearServers();
3061:           currentConfigurationPath = null;
3062:           SwitchToInformationView();
3063:           tabControl.SelectedIndex = 0;
3064:           tabControl.Refresh();
3065:           tabControl.Focus();
3066:           textBoxServerOpcServerName.CausesValidation = true;
3067:           textBoxSubscriptionName.CausesValidation = true;
3068:           SetDirty(false);
3069:         }
3070:       }
3071:
3072:       private void menuItemOpen_Click(object sender, System.EventArgs e)
3073:       {
3074:         SaveValuesInModel();
3075:
3076:         if (PromptForSaveIfNeeded())
3077:         {
3078:           if (openFileDialog.ShowDialog() == DialogResult.OK)
3079:           {
3080:             try
3081:             {
3082:               textBoxServerOpcServerName.CausesValidation = false;
3083:               textBoxSubscriptionName.CausesValidation = false;
3084:               controller.LoadConfiguration(openFileDialog.FileName);
3085:               currentConfigurationPath = openFileDialog.FileName;
3086:               tabControl.SelectedIndex = 0;
```

```
3087:             treeViewServers.SelectedNode = treeViewServers.Nodes[0];
3088:             treeViewServers.Focus();
3089:             textBoxServerOpcServerName.CausesValidation = true;
3090:             textBoxSubscriptionName.CausesValidation = true;
3091:             tabControl.Refresh();
3092:             SetDirty(false);
3093:           }
3094:           catch (ECorruptedConfigFileException ex)
3095:           {
3096:             MessageBox.Show(ERROR_MESSAGE_OPEN_FILE + "\n\nException: " + ex.Message, ERROR_TITLE);
3097:           }
3098:         }
3099:       }
3100:     }
3101:
3102:     private void menuItemSaveAs_Click(object sender, System.EventArgs e)
3103:     {
3104:       SaveValuesInModel();
3105:       if (saveFileDialog.ShowDialog() == DialogResult.OK)
3106:       {
3107:         try
3108:         {
3109:           controller.SaveConfiguration(saveFileDialog.FileName);
3110:           currentConfigurationPath = saveFileDialog.FileName;
3111:           SetDirty(false);
3112:         }
3113:         catch (XmlException ex)
3114:         {
3115:           MessageBox.Show(ERROR_MESSAGE_SAVE_FILE + "\n\nException: " + ex.Message, ERROR_TITLE);
3116:         }
3117:       }
3118:     }
3119:
3120:     private void menuItemSave_Click(object sender, System.EventArgs e)
3121:     {
3122:       SaveValuesInModel();
3123:       if (currentConfigurationPath == null)
3124:       {
3125:         menuItemSaveAs_Click(null, null);
3126:       }
3127:       else
3128:       {
3129:         try
3130:         {
3131:           controller.SaveConfiguration(currentConfigurationPath);
3132:           SetDirty(false);
3133:         }
```

```
3134:         catch (XmlException ex)
3135:         {
3136:           MessageBox.Show(ERROR_MESSAGE_SAVE_FILE + "\n\nException: " + ex.Message, ERROR_TITLE);
3137:         }
3138:       }
3139:
3140:     }
3141:
3142:     private void menuItemExit_Click(object sender, System.EventArgs e)
3143:     {
3144:       Close();
3145:     }
3146:
3147:     private void menuItemAbout_Click(object sender, System.EventArgs e)
3148:     {
3149:       AboutBox ab = new AboutBox();
3150:       ab.ShowDialog(this);
3151:     }
3152:
3153:     private void menuItemEnglish_Click(object sender, System.EventArgs e)
3154:     {
3155:       LoadNewLanguage(LANGUAGE_ENGLISH);
3156:       ModifyRegistry myRegistry = new ModifyRegistry();
3157:       myRegistry.SubKey = REGISTRY_SUB_KEY;
3158:             myRegistry.Write(REGISTRY_VALUE, LANGUAGE_ENGLISH);
3159:     }
3160:     private void menuItemDanish_Click(object sender, System.EventArgs e)
3161:     {
3162:       LoadNewLanguage(LANGUAGE_DANISH);
3163:       ModifyRegistry myRegistry = new ModifyRegistry();
3164:       myRegistry.SubKey = REGISTRY_SUB_KEY;
3165:       myRegistry.Write(REGISTRY_VALUE, LANGUAGE_DANISH);
3166:     }
3167:
3168:     #endregion
3169:
3170:     #region Textbox Event Handlers
3171:
3172:     private void AnyTextChanged(object sender, System.EventArgs e)
3173:     {
3174:       SetDirty(true);
3175:     }
3176:
3177:     #endregion
3178:
3179:     #region Combobox Event Handlers
3180:
```

```
3181:     private void comboBox_SelectionChangeCommitted(object sender, System.EventArgs e)
3182:     {
3183:       SetDirty(true);
3184:     }
3185:
3186:     #endregion
3187:
3188:     #region Radiobutton Event Handlers
3189:
3190:     private void radioButton_CheckedChanged(object sender, System.EventArgs e)
3191:     {
3192:       SetDirty(true);
3193:     }
3194:
3195:     private void radioButtonDaSubscriptionAlarmStateType_CheckedChanged(object sender, System.EventArgs e)
3196:     {
3197:       if (radioButtonDaSubscriptionAlarmStateTypeBoolean.Checked)
3198:         SwitchToAlarmTypeBooleanView();
3199:       else if (radioButtonDaSubscriptionAlarmStateTypeSingleValue.Checked)
3200:         SwitchToAlarmTypeSingleValueView();
3201:       else if (radioButtonDaSubscriptionAlarmStateTypeRange.Checked)
3202:         SwitchToAlarmTypeRangeView();
3203:     }
3204:
3205:     #endregion
3206:
3207:     #region Validation Event Handlers
3208:
3209:     private void textBoxServerOpcServerName_Validating(object sender, System.ComponentModel.CancelEventArgs e)
3210:     {
3211:       SaveValuesInModel();
3212:       if (textBoxServerOpcServerName.Text.Equals(""))
3213:       {
3214:         MessageBox.Show("Please enter a server name!", "Notification");
3215:         textBoxServerOpcServerName.SelectAll();
3216:         e.Cancel = true;
3217:       }
3218:       else if (controller.ServerNameInUse(textBoxServerOpcServerName.Text))
3219:       {
3220:         MessageBox.Show("Server name already exists, please change it!", "Notification");
3221:         textBoxServerOpcServerName.SelectAll();
3222:         e.Cancel = true;
3223:       }
3224:       else
3225:       {
3226:         treeViewServers.Update();
3227:       }
```

```
3228:     }
3229:
3230:     private void textBoxSubscriptionName_Validating(object sender, System.ComponentModel.CancelEventArgs e)
3231:     {
3232:       SaveValuesInModel();
3233:       OpcServerNode server = (OpcServerNode) lastSelectedNode.Parent;
3234:       if (textBoxSubscriptionName.Text.Equals(""))
3235:       {
3236:         MessageBox.Show("Please enter a subscription name!", "Notification");
3237:         textBoxSubscriptionName.SelectAll();
3238:         e.Cancel = true;
3239:       }
3240:       else if (controller.SubscriptionNameInUse(textBoxSubscriptionName.Text, server))
3241:       {
3242:         MessageBox.Show("Subscription name already exists, please change it!", "Notification");
3243:         textBoxSubscriptionName.SelectAll();
3244:         e.Cancel = true;
3245:       }
3246:       else
3247:       {
3248:         treeViewServers.Update();
3249:       }
3250:     }
3251:
3252:     private void textBoxDaSubscriptionAlarmStateSingleValue_Validating(object sender, System.ComponentModel.CancelEventArgs
         e)
3253:     {
3254:       SaveValuesInModel();
3255:       if (textBoxDaSubscriptionAlarmStateSingleValue.Text.Equals(""))
3256:       {
3257:         textBoxDaSubscriptionAlarmStateSingleValue.Text = "0";
3258:       }
3259:       else if (!controller.ValidNumber(textBoxDaSubscriptionAlarmStateSingleValue.Text))
3260:       {
3261:         MessageBox.Show("Please enter a valid value!", "Notification");
3262:         textBoxDaSubscriptionAlarmStateSingleValue.SelectAll();
3263:         e.Cancel = true;
3264:       }
3265:     }
3266:
3267:     private void textBoxDaSubscriptionAlarmStateRangeLow_Validating(object sender, System.ComponentModel.CancelEventArgs e)
3268:     {
3269:       SaveValuesInModel();
3270:       if (textBoxDaSubscriptionAlarmStateRangeLow.Text.Equals(""))
3271:       {
3272:         textBoxDaSubscriptionAlarmStateRangeLow.Text = "0";
3273:       }
```

```
3274:       else if (!controller.ValidNumber(textBoxDaSubscriptionAlarmStateRangeLow.Text))
3275:       {
3276:         MessageBox.Show("Please enter a valid value!", "Notification");
3277:         textBoxDaSubscriptionAlarmStateRangeLow.SelectAll();
3278:         e.Cancel = true;
3279:       }
3280:     }
3281:
3282:     private void textBoxDaSubscriptionAlarmStateRangeHigh_Validating(object sender, System.ComponentModel.CancelEventArgs e)
3283:     {
3284:       SaveValuesInModel();
3285:       if (textBoxDaSubscriptionAlarmStateRangeHigh.Text.Equals(""))
3286:       {
3287:         textBoxDaSubscriptionAlarmStateRangeHigh.Text = "0";
3288:       }
3289:       else if (!controller.ValidNumber(textBoxDaSubscriptionAlarmStateRangeHigh.Text))
3290:       {
3291:         MessageBox.Show("Please enter a valid value!", "Notification");
3292:         textBoxDaSubscriptionAlarmStateRangeHigh.SelectAll();
3293:         e.Cancel = true;
3294:       }
3295:     }
3296:
3297:     #endregion
3298:
3299:     #region MouseOver Event Handlers
3300:
3301:     private void button_MouseEnter(object sender, System.EventArgs e)
3302:     {
3303:       if (sender.Equals(buttonRemoveSelected))
3304:       {
3305:         if (lastSelectedNode is OpcServerNode)
3306:           statusBarPanelLeft.Text = STATUS_BAR_REMOVE_SERVER;
3307:         else if (lastSelectedNode is OpcSubscriptionNode)
3308:           statusBarPanelLeft.Text = STATUS_BAR_REMOVE_SUBSCRIPTION;
3309:       }
3310:       else if (sender.Equals(buttonAddServer))
3311:         statusBarPanelLeft.Text = STATUS_BAR_ADD_SERVER;
3312:       else if (sender.Equals(buttonAddAeSubscription))
3313:         statusBarPanelLeft.Text = STATUS_BAR_ADD_AE_SUBSCRIPTION;
3314:       else if (sender.Equals(buttonAddDaSubscription))
3315:         statusBarPanelLeft.Text = STATUS_BAR_ADD_DA_SUBSCRIPTION;
3316:       else if (sender.Equals(buttonServerServerNameSelect))
3317:         statusBarPanelLeft.Text = STATUS_BAR_SELECT_SERVER;
3318:       else if (sender.Equals(buttonServerAcsAlarmCodesAddAlarmCode))
3319:         statusBarPanelLeft.Text = STATUS_BAR_ADD_ACS_ALARM_CODE;
3320:       else if (sender.Equals(buttonAeSubscriptionSource))
```

```
3321:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_SOURCE;
3322:       else if (sender.Equals(buttonAeSubscriptionSeverity))
3323:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_SEVERITY;
3324:       else if (sender.Equals(buttonAeSubscriptionMessage))
3325:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_MESSAGE;
3326:       else if (sender.Equals(buttonAeSubscriptionCondition))
3327:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_CONDITION;
3328:       else if (sender.Equals(buttonAeSubscriptionServer))
3329:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_SERVER;
3330:       else if (sender.Equals(buttonAeSubscriptionSubscription))
3331:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_SUBSCRIPTION;
3332:       else if (sender.Equals(buttonAeSubscriptionFormula))
3333:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_FORMULA;
3334:       else if (sender.Equals(buttonAeSubscriptionFilterSourcesSelect))
3335:         statusBarPanelLeft.Text = STATUS_BAR_SELECT_SOURCES;
3336:       else if (sender.Equals(buttonAeSubscriptionFilterAreasSelect))
3337:         statusBarPanelLeft.Text = STATUS_BAR_SELECT_AREAS;
3338:       else if (sender.Equals(buttonAeSubscriptionFilterEventCatIdsSelect))
3339:         statusBarPanelLeft.Text = STATUS_BAR_SELECT_EVENT_CAT_IDS;
3340:       else if (sender.Equals(buttonAeSubscriptionSeverityMappingAddMapping))
3341:         statusBarPanelLeft.Text = STATUS_BAR_ADD_SEVERITY_MAPPING;
3342:       else if (sender.Equals(buttonDaSubscriptionValue))
3343:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_VALUE;
3344:       else if (sender.Equals(buttonDaSubscriptionItemName))
3345:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_NAME;
3346:       else if (sender.Equals(buttonDaSubscriptionItemPath))
3347:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_ITEM_PATH;
3348:       else if (sender.Equals(buttonDaSubscriptionResult))
3349:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_RESULT;
3350:       else if (sender.Equals(buttonDaSubscriptionServer))
3351:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_SERVER;
3352:       else if (sender.Equals(buttonDaSubscriptionSubscription))
3353:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_SUBSCRIPTION;
3354:       else if (sender.Equals(buttonDaSubscriptionFormula))
3355:         statusBarPanelLeft.Text = STATUS_BAR_ADD_MESSAGE_FORMAT_FORMULA;
3356:       else if (sender.Equals(buttonDaSubscriptionItemSelect))
3357:         statusBarPanelLeft.Text = STATUS_BAR_SELECT_ITEM;
3358:       else if (sender.Equals(buttonServiceStartService))
3359:         statusBarPanelLeft.Text = STATUS_BAR_START_SERVICE;
3360:       else if (sender.Equals(buttonServiceStopService))
3361:         statusBarPanelLeft.Text = STATUS_BAR_STOP_SERVICE;
3362:     }
3363:
3364:     private void button_MouseLeave(object sender, System.EventArgs e)
3365:     {
3366:       statusBarPanelLeft.Text = DEFAULT_STATUS_BAR_TEXT;
3367:     }
```

```
3368:
3369:     #endregion
3370:
3371:     #region Overridden Event Handlers
3372:
3373:     protected override void OnClosing(CancelEventArgs e)
3374:     {
3375:       if (!e.Cancel)
3376:         if (!PromptForSaveIfNeeded())
3377:           e.Cancel = true;
3378:       base.OnClosing (e);
3379:     }
3380:
3381:     #endregion
3382:
3383:     #endregion
3384:
3385:
3386:   }
3387:
3388:   /// <summary>
3389:   /// Class used to run updates of the progress bar as a seperate thread
3390:   /// </summary>
3391:   public class UpdateProgressThread
3392:   {
3393:     private View view;
3394:     private int time;
3395:
3396:     public UpdateProgressThread(View view, int time)
3397:     {
3398:       this.view = view;
3399:       this.time = time;
3400:     }
3401:
3402:     public void Run()
3403:     {
3404:       view.Invoke(view.delegateInitProgress, new Object[] {time});
3405:       for (int i=1; i<=time; i++)
3406:       {
3407:         Thread.Sleep(1000);
3408:         view.Invoke(view.delegateUpdateProgress, new Object[] {i});
3409:       }
3410:     }
3411:   }
3412: }
3413:
```