ELEC-E8113

Information systems in industry

OPC UA teamwork

Author:	Versio:	Date:
Ilkka Seilonen	1.0	5.9.2022

Table of contents

1 Introduction

In this OPC UA (Unified Architecture) assignment you are supposed to learn about OPC UA technology and communication of automation data between automation systems and information systems. You are doing this though implementing a simple OPC UA server application for communicating data of an example process to an imaginary client in a particularly requested form.

1.1 Automation data

The example process is an educational process available at the laboratory of Aalto EEA. In this exercise we consider the data available from a part of the boiler sub-process (see figure 1). The sub-process contains a boiler, four sensors, three valves and a pump. The automation application controlling the process runs on a PLC. It contains a PID controller controlling water level in the boiler. Data form the automation is available through an OPC Classic server and an OPC UA wrapper. The application was developed by a former employee several years ago. Your job now is to develop the communication with support for selected standards requested by an imaginary client.

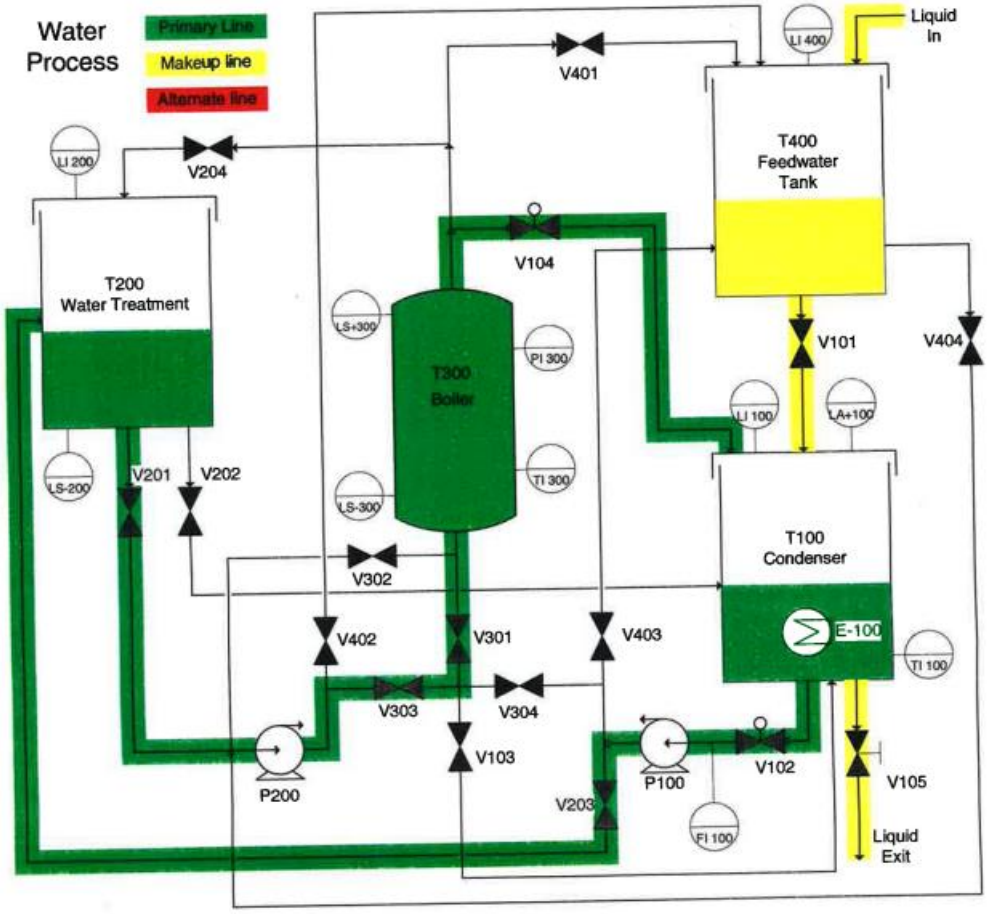

Figure 1. Diagram of the example process. Boiler is in the middle.

1.2 Existing OPC UA server application

The data needed in this exercise is available through a simple test application called DemoServer. The application is available in Mycourses website in folder Materials / Teamwork / OPC UA. You can download and install the application, and use it in as a part of your exercise. You might need to modify the application (e.g. paths) to fit your own computer.

The DemoServer application is an OPC UA server. You can access it with UaExpert (see figure 2). The data available at the DemoServer application is organized as a flat list of variables. This is quite typical in OPC Classic based systems. The values of the variables are arbitrary and there is mostly no application logic connected to them. You can change the values through UaExpert.

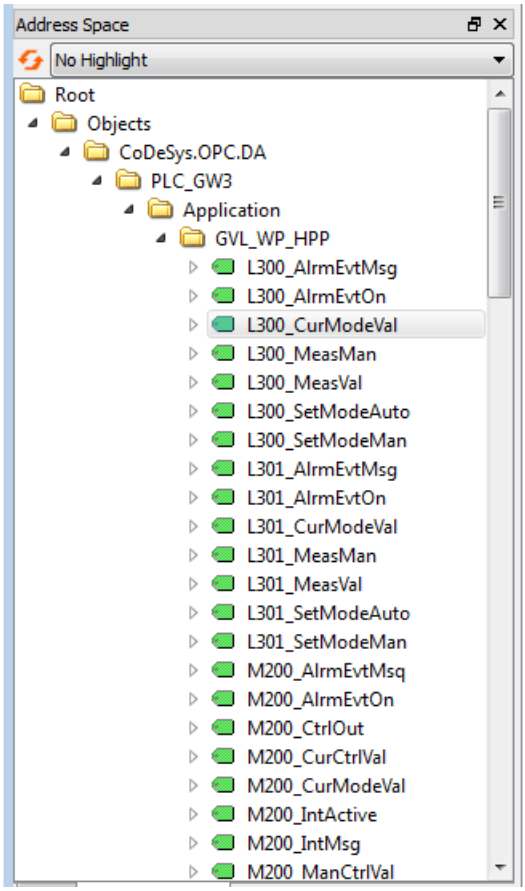

Figure 2. Part of data available from the DemoServer application

1.3 New OPC UA server application

You are supposed to develop a chained OPC UA server application which communicates with the DemoServer application and provides its data in another form. The chained server pattern is illustrated in figure 3. Let's call the new application AppServer for now. The new form of data at the AppServer application has to conform to a few specifications outlined below. Regardless the different forms of data the values of the variables at the both servers should be the same.

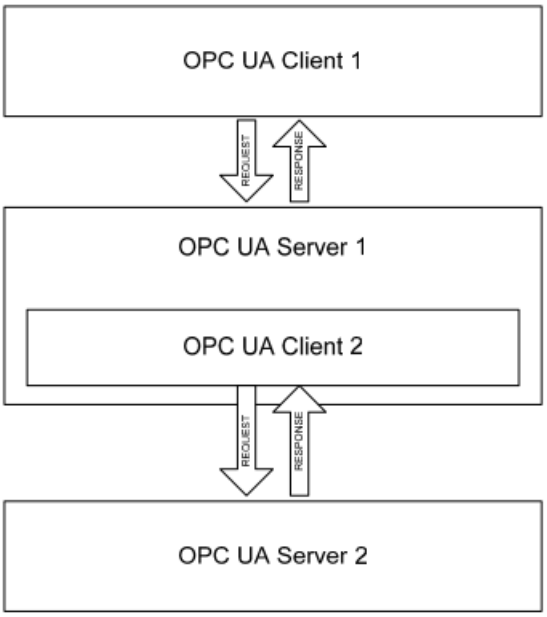

Figure 3. Chained OPC UA server pattern

The representation of data at the AppServer application should fulfil the following requirements:

1. The data about all other tags than PIC300 should follow the OPC UA for Process Automation Devices Companion Specification, release 1.00. You have to utilize the PADIMType type.
2. The data about the tag PIC300 should follow the OPC UA for Programmable Logic Controllers base on IEC61131-3, release 1.00. You can utilize the CtrlConfigurationType type.
3. The variables whose name contain SetModeAuto or SetModeMan should be provided as OPC UA methods according to the OPC UA Specification, Part 3: Address Space Model, release 1.04.
4. The variables who have the nature of alarms (e.g. name contains “Alrm”) should be provided only as OPC UA alarms (and not as variables) according to the OPC UA Specification Part 9: Alarms & Conditions, release 1.04.

The application logic of the AppServer is required to implement the following functionality:

1. Reading value of any variable. The values should be same at DemoServer and AppServer.
2. Writing value of any variable. The values should be same at both servers.
3. Subscribing and unsubscribing to any variable. The values should be same at both servers.
4. Subscribing and unsubscribing to any alarm. The alarms at AppServer should follow the same application logic than respective variables at DemoServer.

5. Calling any method with simple application logic. It is possible to call all methods with names SetModeAuto and SetModeMan. The value of the variable CurModeVal is set to either AUTO or MANUAL according these calls.

2 Implementation tools

2.1 Overview

This exercise has to be done with your own computer. A laptop with Windows operating system is recommended but Linux might be possible as well.

You will need a few software tools in order to do the exercise. You have to install them yourself to your computer. The required tools are listed below and explained in more details in subsequent chapters.

- Java SE Development Kit (JDK), version 8, or later version compatible with Prosys SDK.
- Eclipse IDE for Java Developers, version 2018-12 or later.
- Prosys OPC UA Java SDK, version 4.9.0. (available from Prosys)
- UaExpert, version 1.5.0 or later.
- Example programs DemoServer and AppServer (available at Mycourses).

2.2 Java and Eclipse

This exercise is programmed with Java programming language. In order to do this you will need Java SE Development Kit (JDK), version 8 or later version compatible with Prosys SDK. It is available for download at Oracle website (see chapter 3). OpenJDK should also be fine in Linux.

It is proposed that you use the Eclipse IDE (integrated development environment) to do the required programming and debugging. You will need Eclipse IDE for Java (or Java EE) Developers (see figure 4). Eclipse is available for download at Eclipse website (see chapter 3). Download the 2018-12 version or later. Documentation of Eclipse IDE is also available at the website. Alternatively you can use another IDE for Java if you wish.

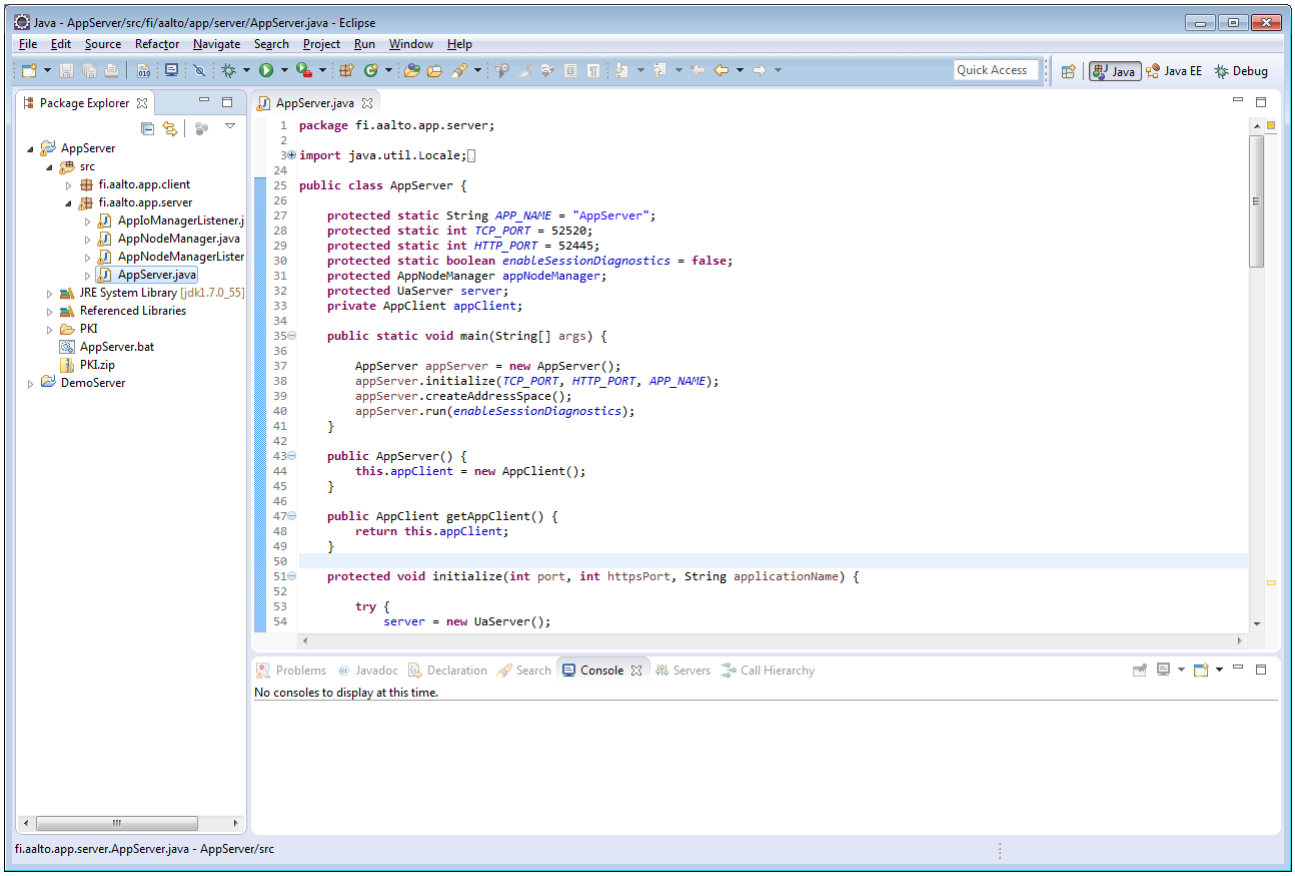

Figure 4.Eclipse IDE for Java Developers

2.3 Prosys OPC UA Java SDK

Prosys OPC UA Java SDK is a software library you are going to use to develop the AppServer in this exercise. It is a Java library you can use with Eclipse. The example programs were developed with it. The library can be obtained from Prosys though a request.

Prosys OPC UA Java SDK download contains an OPC UA library and a few additional libraries. The most essential library is the actual SDK which can be used to develop OPC UA client and server applications. You can study the tutorials, documentation and sample programs in order to learn how the SDK API is used for designing an OPC UA application. Javadoc documentation of the SDK API is included in the Prosys OPC UA Java SDK download (see figure 5). The SDK also contains a few useful tutorials. It is very much recommended to study them.

In order to implement the requested new address space for your application you will need to use the code generation tools of the SDK. You have to read its documentation provided with the SDK. Information model files for the requested AppServer application are available in the AppServer project. You will need them in code generation.

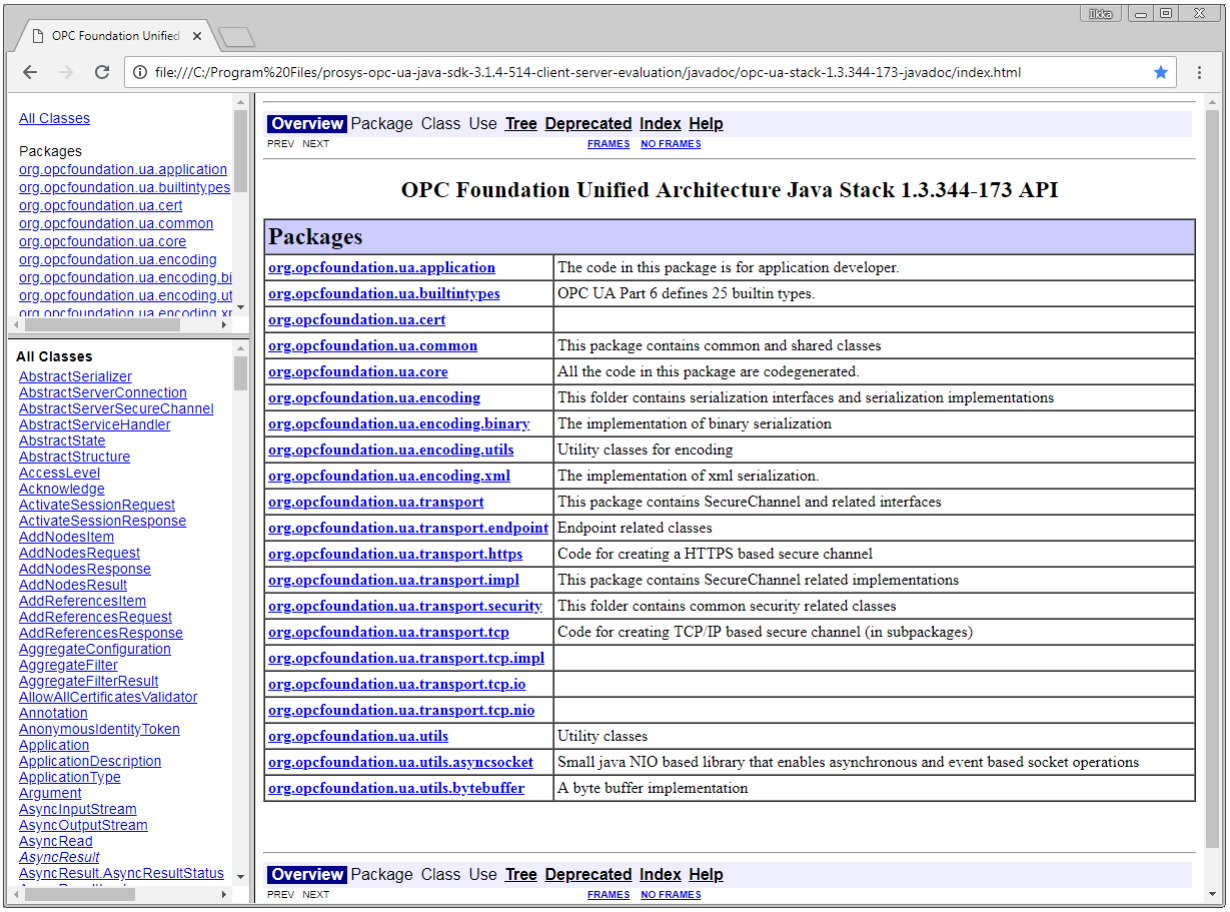

Figure 5. Javadoc documentation of Prosys OPC UA Java SDK.

2.4 UaExpert

UaExpert is a general purpose OPC UA client application which can be used to connect to any OPC UA server (see figure 6). It is available for download at Unified Automation website. Versions for Windows and Linux and documentation are provided. If necessary you can replace UaExpert with another generic OPC UA client, e.g. Prosys OPC UA Client.

In this exercise you can use UaExpert for connecting both the DemoServer and AppServer. You can study the contents of data at both servers and make sure that their form and content are as required.

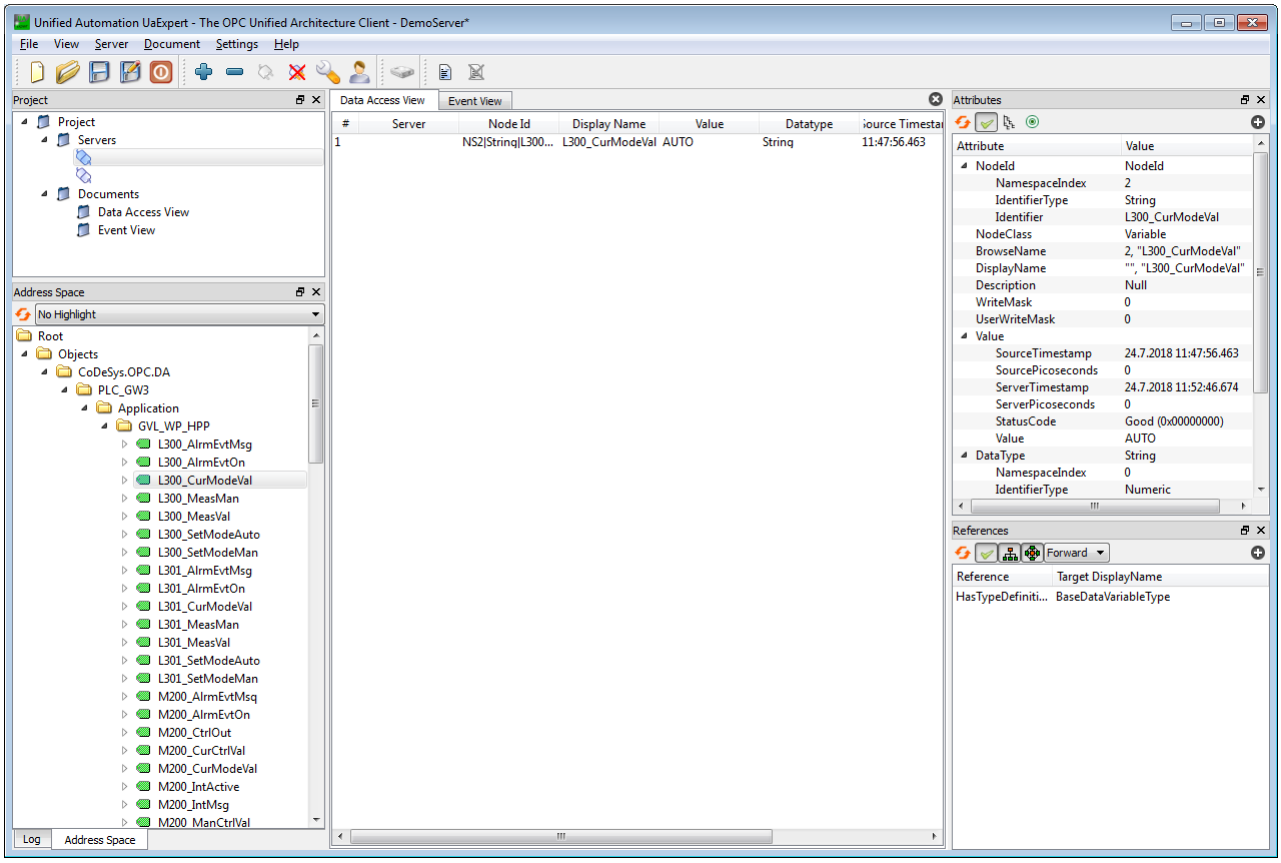

Figure 6. User interface of UaExpert.

2.5 Example programs

Two example programs are available as a starting point for the exercise: DemoServer and AppServer. Both are provided as source code and binaries. In Windows you can run them through their bat-files. However, you have to modify the content of the bat-files to match your installation. You can also create Eclipse projects for editing and running them. In this case you have to check that the project properties match your Eclipse installation (e.g. Java build path).

DemoServer is a relatively simple OPC UA server application developed with Prosys OPC UA Java SDK. It creates a simple address space with a structure consisting a flat list of variables. The variables have initial values but they do not change unless your write new values. There is no application logic between values of different variables. However, this is the input data to your application. You can access the address space of DemoServer e.g. with UaExpert at URL opc.tcp://localhost:52500/DemoServer.

AppServer provided as an example program is a relatively simple OPC UA server application that you can use as a starting point for developing your own AppServer application. You can access the address space of AppServer e.g. with UaExpert at URL opc.tcp://localhost:52520/AppServer. The provided AppServer does not fulfil all the requirements presented in chapter 1.3. The address space it creates is an identical copy of the original one. It implements reading, writing and subscription to variables but not subscription to alarms nor calling methods. Even so, the AppServer application

contains the basic structure of the requested application. The important Java classes are explained below.

- AppServer defines the application main program you can run. After initialization it creates other necessary objects (AppNodeManager and listeners). It also has a link to AppClient.
- AppNodeManager creates the address space. The example program creates a copy of the original address space read from DemoServer through AppClient. You have to modify this.
- AppIoManagerLister implements read and write services. Read and write service calls are redirected to the original address space.
- AppNodeManagerListener implements subscription to variables service. Subscriptions are redirected to the original address space.
- AppClient provides OPC UA connection to the original address space of DemoServer.
- AppMonitoredDataItemListener observes changes of variable values in the original address space. This is needed for implementing subscriptions.
- AppEventManagerListener does not exist but you will need one in your AppServer application. It will be needed for implementing subscriptions to alarms. It should be a subclass of EventManagerListener.
- AppMethodManagerListener does not exist either but should be developed too. It will be need for implementing method calls. It should be a subclass of CallableListener.

3 Useful links

The following links are likely to contain useful information about OPC UA and the necessary software tools. You can find more through Google searches. However, note that all information that you may find might not be relevant to those versions of OPC UA or the tools you have, or might not be relevant at all. Be critical about the information you find.

OPC UA

Mahnke, W., Leitner, S-H., Damm, M. OPC Unified Architecture, Springer, 2009. (available in Mycourses)

https://opcfoundation.org/developer-tools/specifications-unified-architecture

Java 8

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Eclipse

http://www.eclipse.org/downloads/eclipse-packages/

Prosys OPC UA Java SDK

https://www.prosysopc.com/products/opc-ua-java-sdk/ (mention Aalto University and ELEC-E8113 course in your request, ask for SDK version 4.9.0)

UaExpert

https://www.unified-automation.com/products/development-tools/uaexpert.html

4 Deliverables

You need to demonstrate the teacher that your application works and you have a justified understanding how it does its job. You are requested to provide the following deliverables:

1. Programs. You need to present the Java source code of your application that fulfils the given requirements.
2. Demonstration. You have to agree a demonstration event with the teacher. In the event you are requested to present the address space of your application and show that the application fulfils its functional requirements.
3. Document. You need to create a very short (preferably no more than 3 pages) document about the address space of your application. One or more figures describing the address space with the notation used in OPC UA specifications accompanied with short textual explanations when needed is quite enough. A draft version of this document should be delivered before the actual implementation of the application.

Appendix A: FAQ

Q1: We have serious trouble and cannot make any progress. What should we do?

A1: Check the course web page on mycouses.aalto.fi first. If that does not help then send email to ilkka.seilonen@aalto.fi**.**

Q2: How I add new Listener classes for Events and Methods?

A2: The following code could be useful.

```
appNodeManager.getEventManager().setListener(new
AppEventManagerListener(appClient.getClient(), this.server));

((MethodManagerUaNode)appNodeManager.getMethodManager()).addCallListener(new
AppMethodManagerListener(appClient.getClient()));
```

Q3: What is the minimum set of results needed for accepted teamwork?

A3: You have to be able to demonstrate an OPC UA server which has an address space with required data model. In addition, the server should fulfil requirements concerning read, write and subscriptions to variables at least.

Q4: Can I ask the teacher if my answer is “right” or “perfect”?

A4: The teacher should refuse to answer such questions. The quality of your answer will be evaluated after the teamwork. You should have a reason to believe that your answer is “good enough”.

The teacher is supposed to provide hints about how to make progress and overcome problematic situations.