OPC CONNECT

OPC Connectivity using Java

- What is OPC?
  - Why OPC?
  - Some examples of application
    - BOSCH
    - ABB
- Case Study: Java-based OPC Client implementation
  - DCOM configuration
  - Utgard
    - Eclipse configuration
    - Reading tags
    - Writing tags
- Summary and further steps

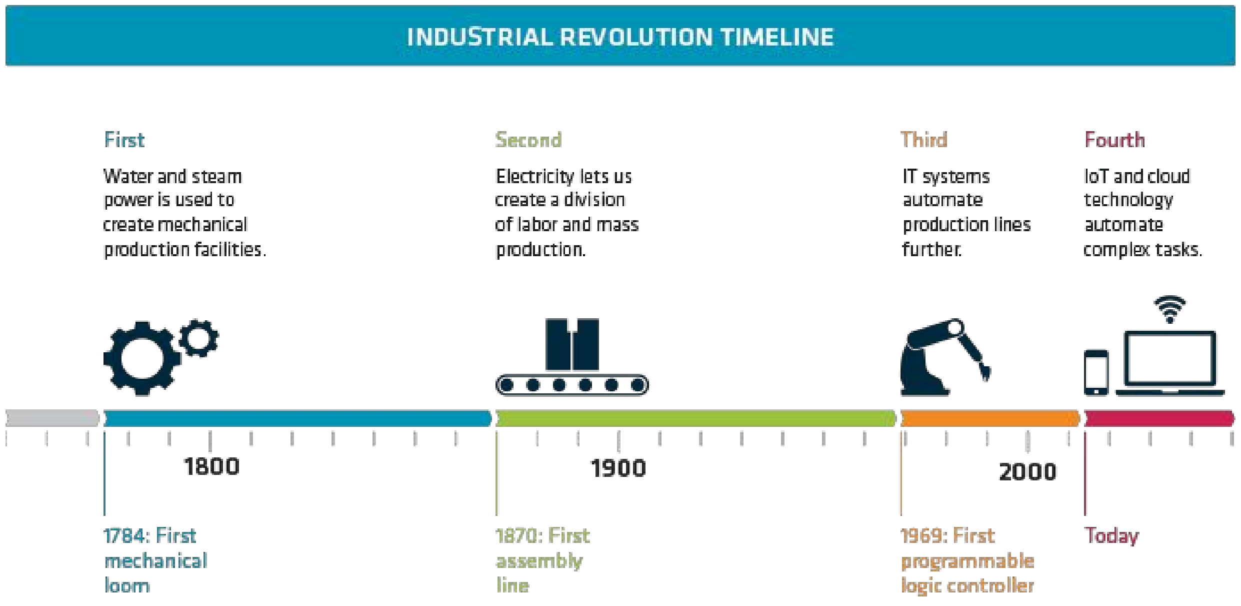
Nehmen Sie Kontakt zu uns auf:

Brockhaus Consulting
Gustav Stresemann Ring 1
D - 65189 Wiesbaden
Fon: +49-611-97774-332
Fax: +49-611-97774-432
Email: mbohnen@brockhaus-gruppe.de

In 2011 at the Hannover Fair the expression "Industrie 4.0" appeared by first time. It refers to the fourth industrial revolution, involves the employment of information systems and electronics in the manufacturing automation processes and comprises the technological concepts of cyber-physical systems, the Internet of Things and the Internet of Services. In turn, it allows the development of the so-called Smart Factory, whose six design principles are:

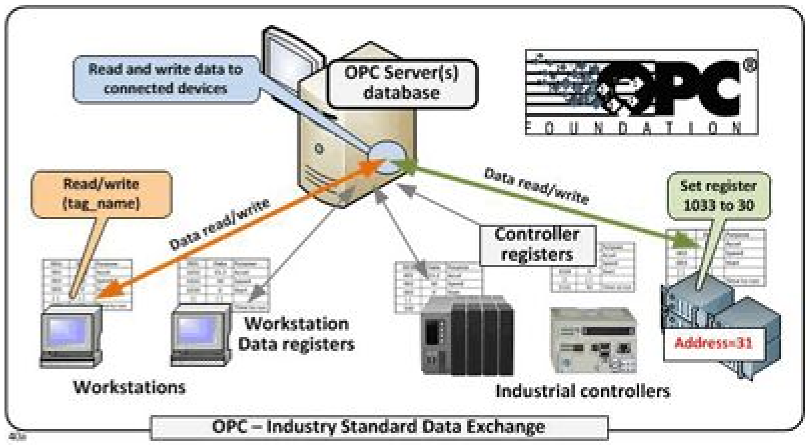
- *Interoperability: the ability of cyber-physical systems (i.e. workpiece carriers, assembly stations and products), humans and Smart Factories to connect and communicate with each other via the Internet of Things and the Internet of Services*
- *Virtualization: a virtual copy of the Smart Factory which is created by linking sensor data (from monitoring physical processes) with virtual plant models and simulation models*
- *Decentralization: the ability of cyber-physical systems within Smart Factories to make decisions on their own*
- *Real-Time Capability: the capability to collect and analyse data and provide the derived insights immediately*
- *Service Orientation: offering of services (of cyber-physical systems, humans or Smart Factories) via the Internet of Services*
- *Modularity: flexible adaptation of Smart Factories to changing requirements by replacing or expanding individual modules*

--- Source: Wikipedia


INDUSTRIAL REVOLUTION TIMELINE
First
Water and steam power is used to create mechanical production facilities.
Second
Electricity lets us create a division of labor and mass production.
Third
IT systems automate production lines further.
Fourth
IoT and cloud technology automate complex tasks.
1800
1900
2000
1784: First mechanical loom
1870: First assembly line
1969: First programmable logic controller
Today

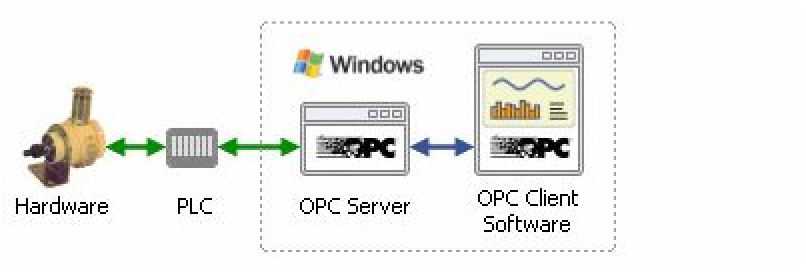
What is OPC?

- *"OPC is the interoperability standard for the secure and reliable exchange of data in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard."* - OPC Foundation 2015



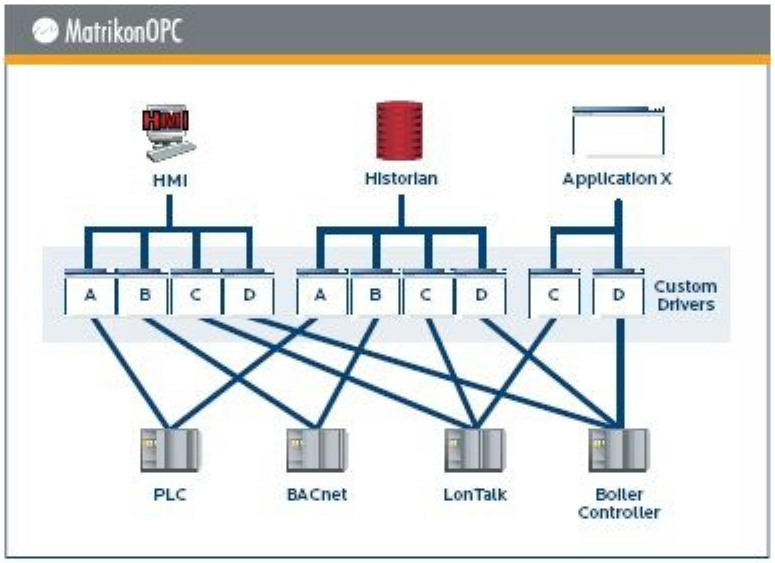
- *"Open Platform Communications (OPC) is a series of standards and specifications for industrial telecommunication. An industrial automation industry task force developed the original standard in 1996 under the name OLE for Process Control (Object Linking and Embedding for Process Control). OPC specifies the communication of real-time plant data between control devices from different manufacturers."* - Wikipedia 2015

- *"OPC is a widely accepted industrial communication standard that enables the exchange of data between multi-vendor devices and control applications without any proprietary restrictions. An OPC server can communicate data continuously among PLCs on the shop floor, RTUs in the field, HMI stations, and software applications on desktop PCs. Even when the hardware and software are from different vendors, OPC compliance makes continuous real-time communication possible. It is open connectivity in industrial automation and the enterprise systems that support the industry. Interoperability is assured through the creation and maintenance of non-proprietary open standards specifications. The first OPC standard specification resulted from the collaboration of a number of leading worldwide automation suppliers working in cooperation with Microsoft. Originally based on Microsoft's OLE COM and DCOM technologies, the specification defined a standard set of objects, interfaces, and methods for use in process control and manufacturing automation applications to facilitate interoperability. The COM/DCOM technologies provided the framework for software products to be developed. There are now hundreds of OPC Data Access servers and clients."* - Kepware Technologies 2015

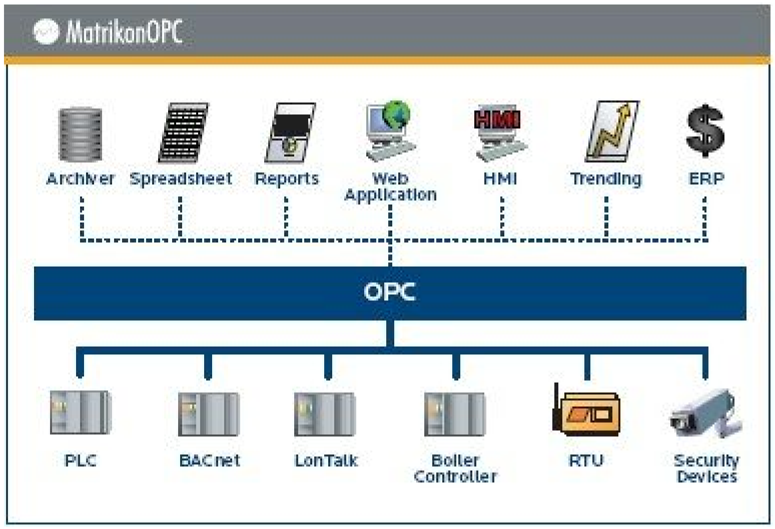


Why OPC?

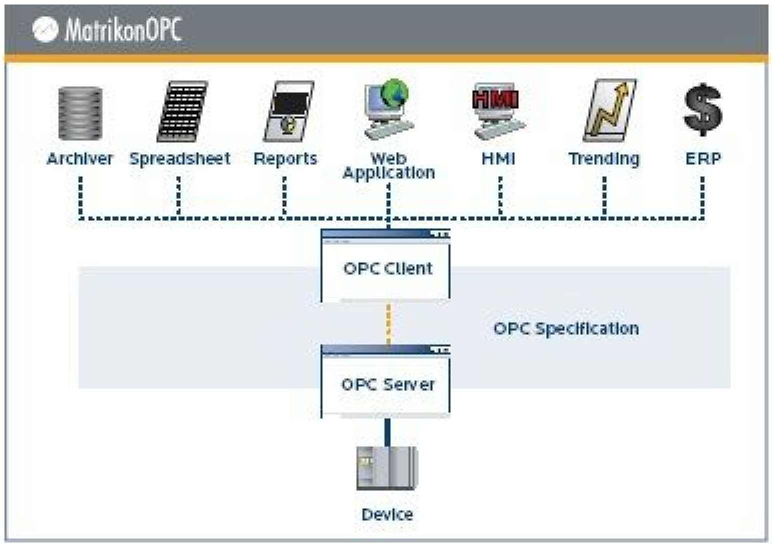
Initially in the process control industry, it arose the following problem (custom driver problem). It was necessary to share data between devices of different vendors. Sometimes this multi-vendor interconnectivity was not achieved due to proprietary protocols and hardware issues. When it was possible, it led to expensive custom solutions and difficult to maintain. Each application required a device or protocol specific driver to allow it to communicate with each respective device. Drivers were not re-usable between applications because each application used its own data format(s).



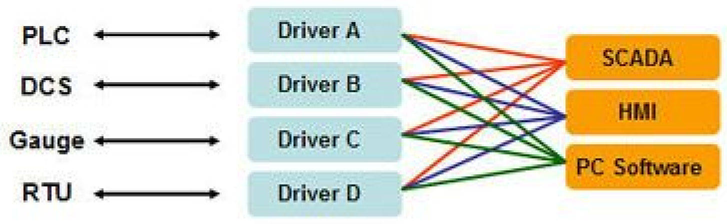
OPC offers a simple, standards based, solution for this problem. It introduces an abstraction layer between devices and applications to allow data to pass between them without them being aware of each other's internal data representations.

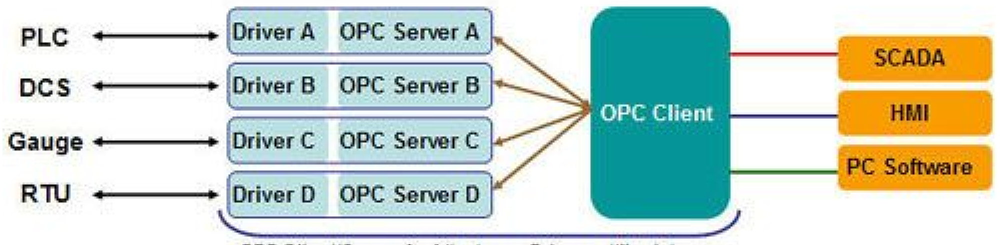


OPC Specifications define how OPC Clients and OPC Servers communicate but leave the OPC Client-Application and OPC Server-Device communications open since this depends on the Application and Devices being used.



This standardized data exchange between the OPC Clients and OPC Servers allows any OPC Client to communicate with any other OPC Server. Since the OPC Server takes care of translating Native device communications into an OPC format and the OPC Client does the same on the Application side; Applications and Devices can share data between each other without having to know each other's native protocols or data formats.



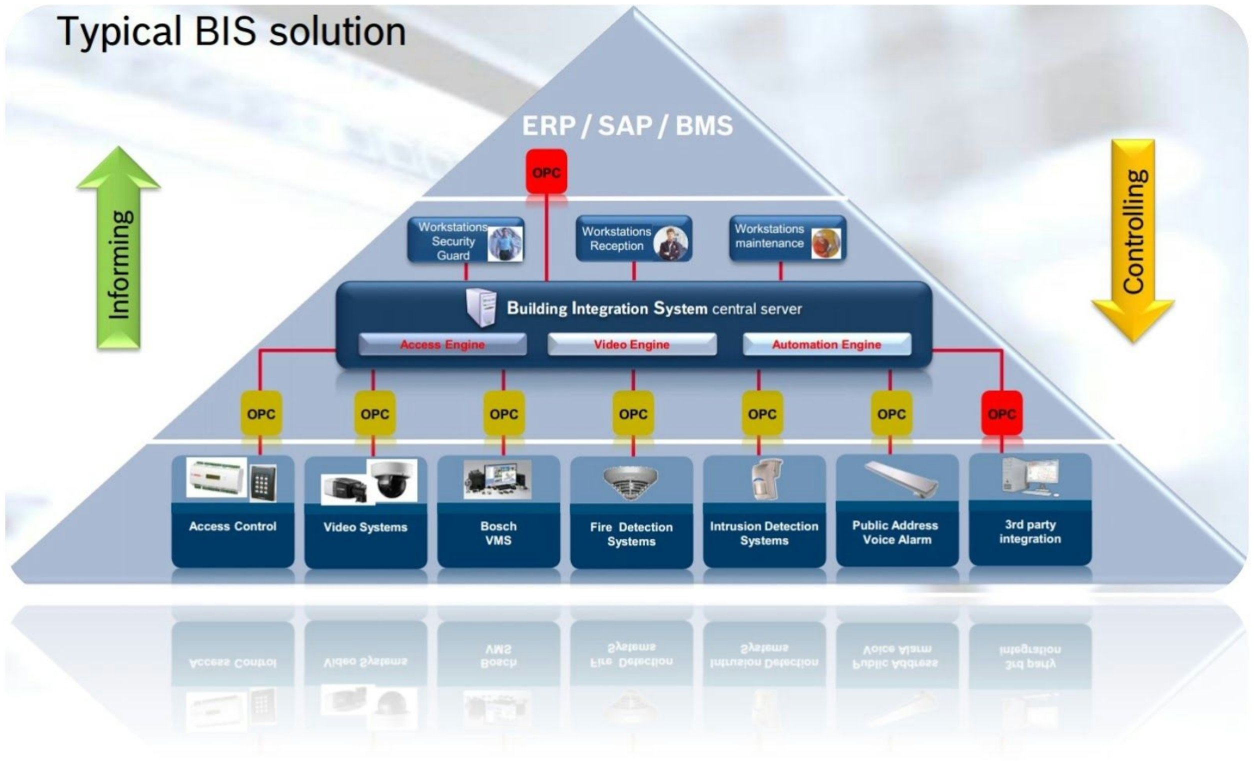


OPC Client/Server Architecture – Drivers still exist, but clients do not see them anymore

Some examples of applications

Such as it mentioned above, the OPC standard makes it possible the connection and communication between the different production devices, humans and Smart Factories in order to facilitate the interoperability. Below some current examples of application are commented briefly.

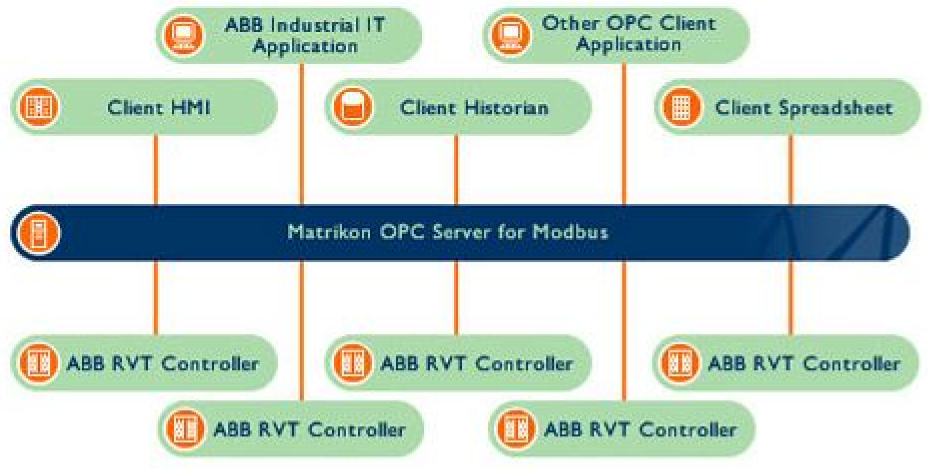
BOSCH



Bosch has developed the Building Integration System (BIS), providing an integrated and OPC-based solution in order to manage the security issue in buildings. More information about it can be found here.

ABB

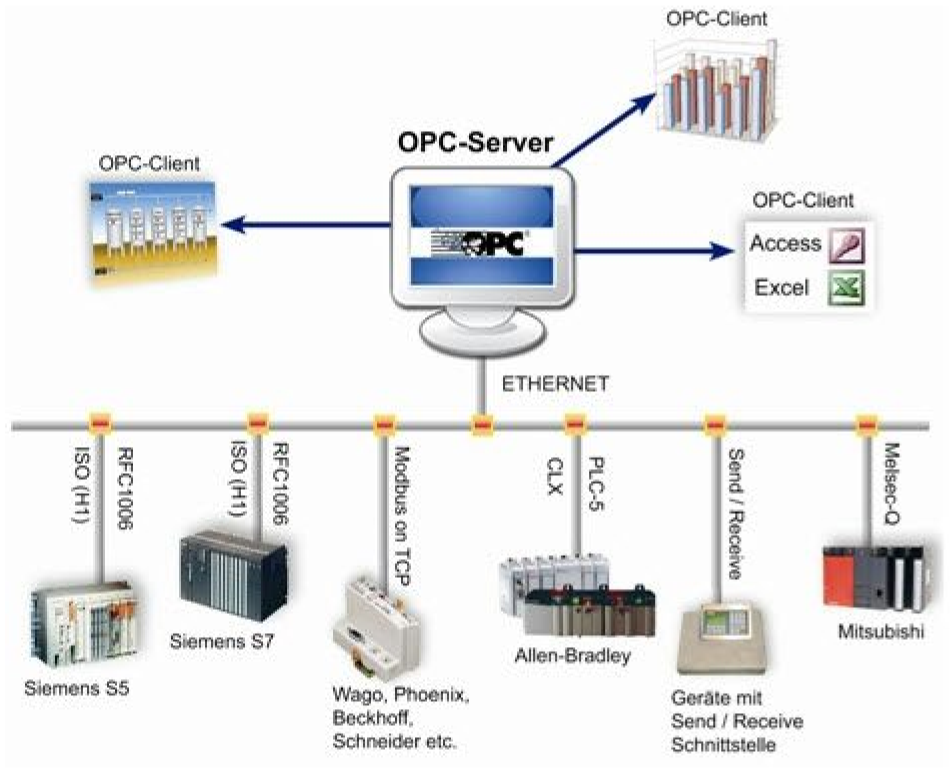
ABB (ASEA Brown Boveri), one of the world leading companies in robotics and the power and automation technology areas, also bet on the OPC standard usage for their solutions.



The schema shows five RVT controllers (Power Factor controllers) as Modbus slave devices connected to the Matrikon OPC Server for Modbus. Different OPC Clients can interact with the OPC Server.

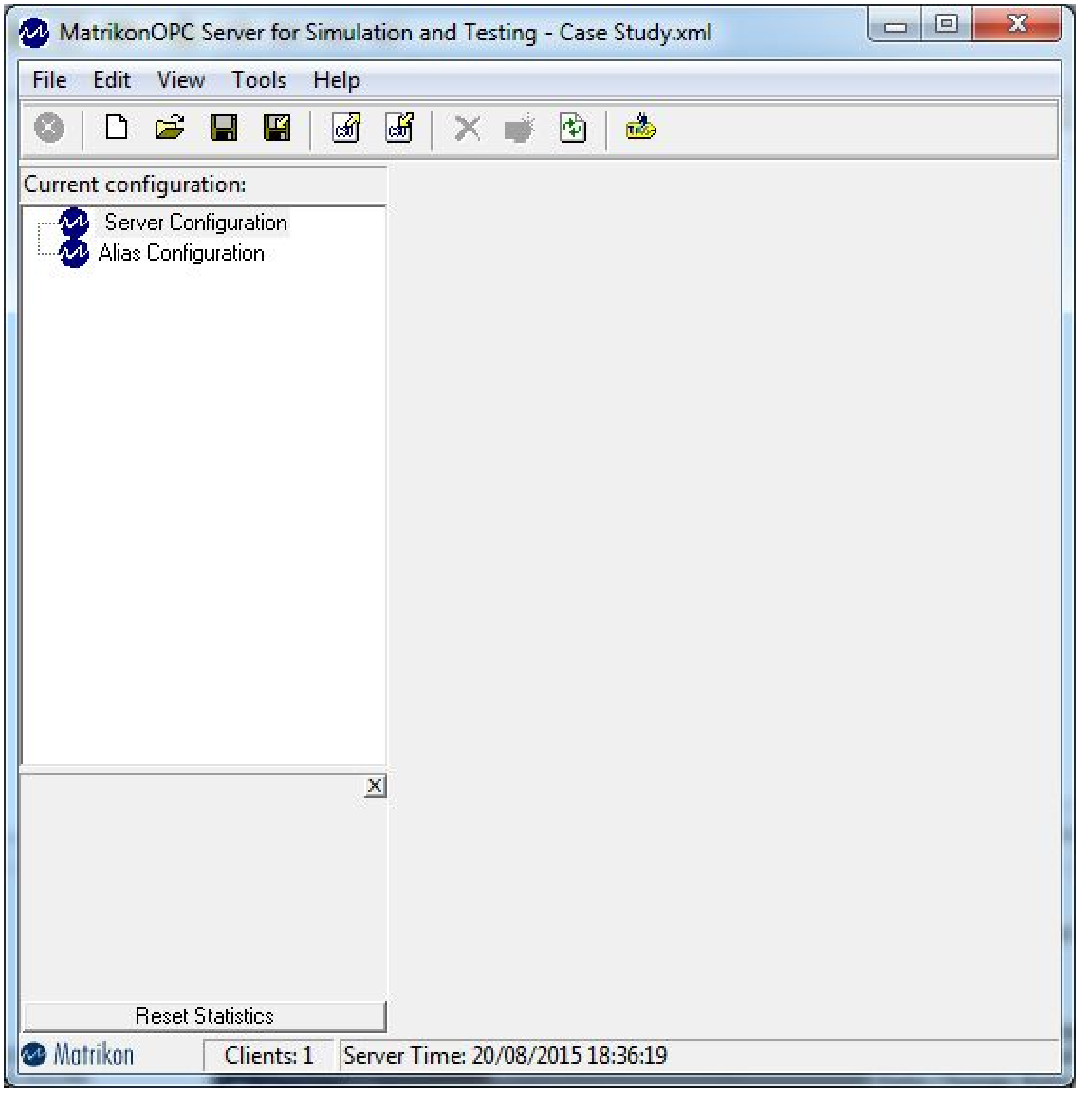
Case Study: Java-based OPC Client implementation

After the above concepts presentations, it is about time to show the connectivity between an OPC Server, MatrikonOPC Server for Simulation and Testing in this case, and an OPC Client (Java-based solution called Utgard). Once installed the server, the steps described below are necessary.

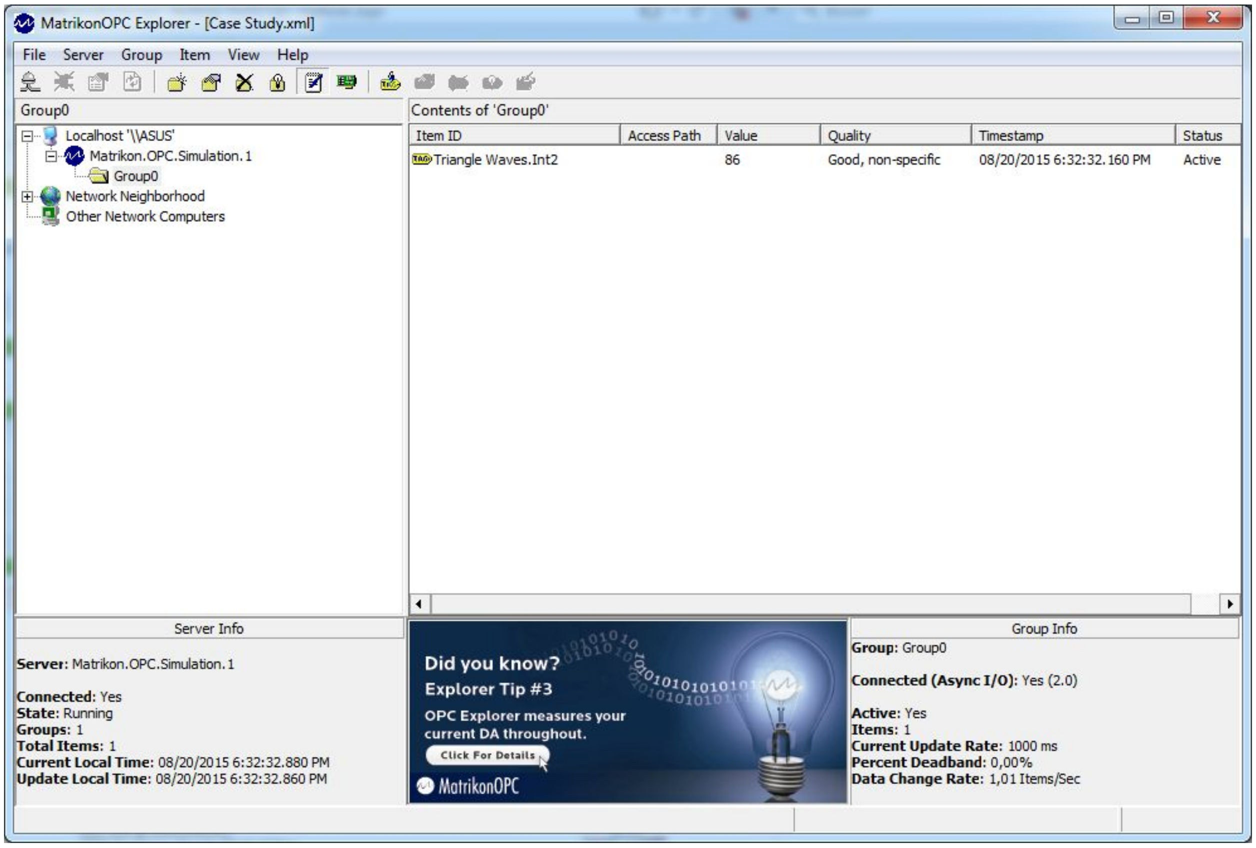


DCOM configuration

Distributed COM (DCOM) is a Microsoft technology that provides Windows applications with the ability to connect from one computer to another on a LAN, a WAN, or an Internet connection. For example, DCOM allows the OPC client application to communicate from one computer to the OPC server on another computer.
DCOM can be quite potent and enable many interesting applications. However it is important to understand the details to ensure OPC works properly using DCOM. The DCOM tutorial, corresponding to the operative system installed in your equipment, can be found in the website of MatrikonOPC. Although this process can be a bit long, it is important to follow all the instructions, especially deactivate the Windows firewall.


MatrikonOPC Server for Simulation and Testing - Case Study.xml
File Edit View Tools Help
Current configuration:
Server Configuration
Alias Configuration
Reset Statistics
Matrikon
Clients: 1
Server Time: 20/08/2015 18:36:19

Packed along with MatrikonOPC Server for Simulation and Testing, there is an application, called MatrikonOPC Explorer, which is a OPC Client with functionality for testing and troubleshooting OPC servers and OPC connections.



Item ID	Access Path	Value	Quality	Timestamp	Status
Triangle Waves.Int2		86	Good, non-specific	08/20/2015 6:32:32.160 PM	Active

Server Info

Server: Matrikon.OPC.Simulation.1

Connected: Yes
State: Running
Groups: 1
Total Items: 1
Current Local Time: 08/20/2015 6:32:32.880 PM
Update Local Time: 08/20/2015 6:32:32.860 PM

Group Info

Group: Group0

Connected (Async I/O): Yes (2.0)

Active: Yes
Items: 1
Current Update Rate: 1000 ms
Percent Deadband: 0,00%
Data Change Rate: 1,01 Items/Sec

Utgard

The most common OPC specification is OPC Data Access (OPC DA), which is used to read and write real-time data. Utgard is the OPC Client part implementation of the OPC DA 2.0 interface. Therefore, it does not work with any OPC Unified Architecture (OPC UA) setup. Below are listed the prerequisites on the development computer:

- The OPC Server should be installed in a Windows computer (note that Home Editions of Windows might not work due to restrictions on the Windows operating system)
- JDK (6 or 7)
- Eclipse 3.5+ or 4.x

In the following sections, the Eclipse basic configuration for Utgard as well as tags' writing and reading Java code examples will be shown.

Eclipse configuration

As first step, it is necessary to download a set of libraries:

- ❏ Utgard binaries
- ❏ jinterop binaries

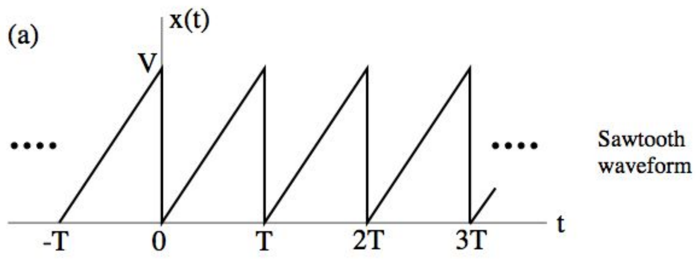
- ❑ External binaries

After starting Eclipse and creating a project, the following jar files, corresponding to the libraries mentioned above, should be put into a folder lib and added to the build path.

External binaries
• slf4j.api_1.6.4.jar • ch.qos.logback.classic_1.0.0.jar • ch.qos.logback.core_1.0.0.jar • org.openscada.external.jcifs_1.2.25.201303051448.jar
jinterop binaries
• org.openscada.jinterop.core_2.0.8.201303051454.jar • org.openscada.jinterop.deps_1.0.0.201303051454.jar
Utgard binaries
• org.openscada.opc.dcom_1.0.0.201303051455.jar • org.openscada.opc.lib_1.0.0.201303051455.jar

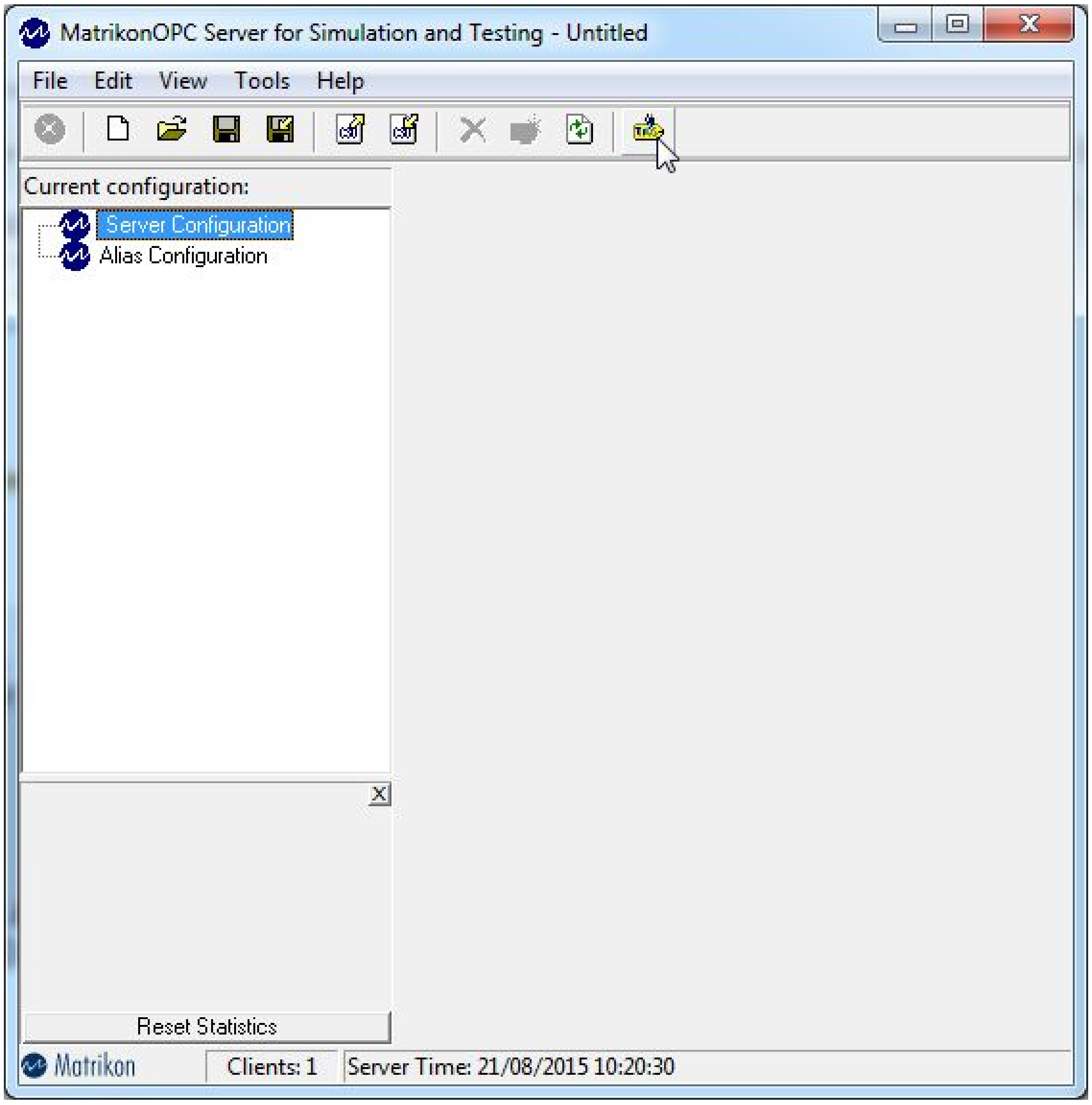
Reading tags

A Saw-toothed signal is created by the Matrikon OPC Server for Simulation and Testing.

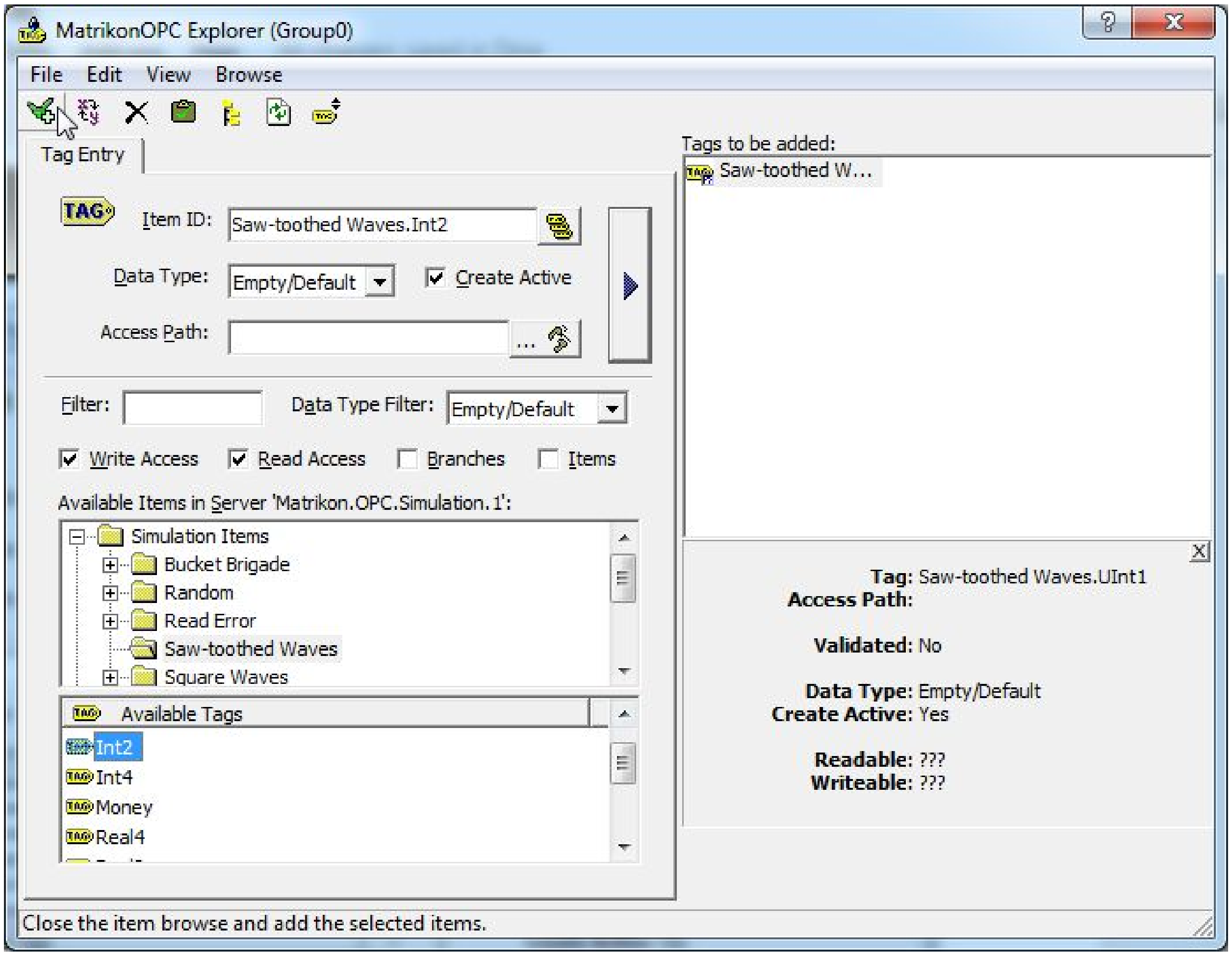


The procedure for it requires:

1. Open the server and click on the icon Tag of the icons bar.


MatrikonOPC Server for Simulation and Testing - Untitled
File Edit View Tools Help
Current configuration:
Server Configuration
Alias Configuration
Reset Statistics
Matrikon
Clients: 1
Server Time: 21/08/2015 10:20:30

2. A MatrikonOPC Explorer window opens. Select Simulation Items -> Saw-toothed Waves in the tree structure corresponding to "Available Items in Server 'Matrikon.OPC.Simulation.1'". Then, select Int2 in the drop-down menu corresponding to "Available Tags". In order to add this tag to the server, click on the first icon of the icons bar in this window.



Item ID: Saw-toothed Waves.Int2
Data Type: Empty/Default
Create Active: ✓
Access Path:
Filter:
Data Type Filter: Empty/Default
Write Access: ✓
Read Access: ✓
Branches: ☐
Items: ☐

Available Items in Server 'Matrikon.OPC.Simulation.1':

- Simulation Items
  - Bucket Brigade
  - Random
  - Read Error
  - Saw-toothed Waves
  - Square Waves

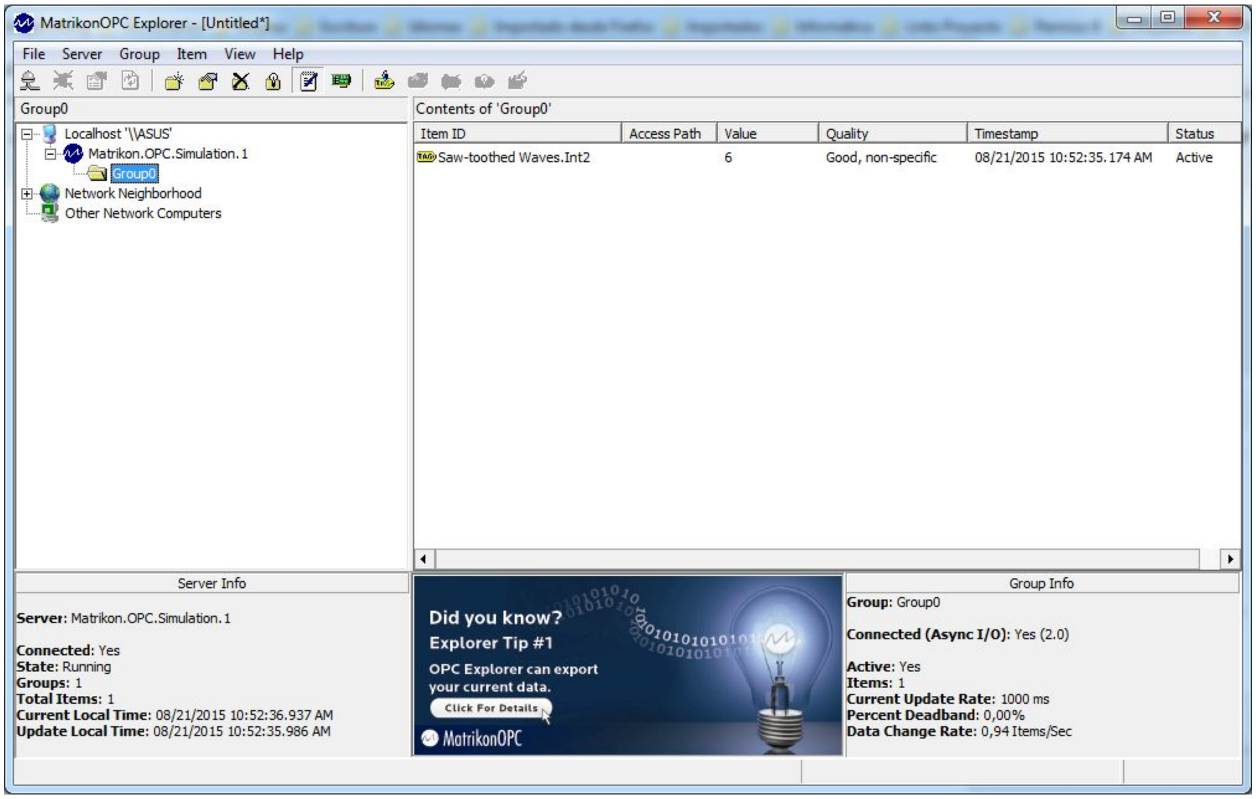
Available Tags: Int2, Int4, Money, Real4

Tags to be added: Saw-toothed W...

Tag: Saw-toothed Waves.UInt1
Access Path:
Validated: No
Data Type: Empty/Default
Create Active: Yes
Readable: ???
Writeable: ???

Close the item browse and add the selected items.

Once the tag has been added, the MatrikonOPC Explorer is opened and shows the evolution of value tag.


MatrikonOPC Explorer - [Untitled*]
File Server Group Item View Help

Group0

Localhost '\\ASUS'
Matrikon.OPC.Simulation.1
Group0
Network Neighborhood
Other Network Computers

Contents of 'Group0'

Item ID	Access Path	Value	Quality	Timestamp	Status
Saw-toothed Waves.Int2		6	Good, non-specific	08/21/2015 10:52:35.174 AM	Active

Server Info

Server: Matrikon.OPC.Simulation.1

Connected: Yes
State: Running
Groups: 1
Total Items: 1
Current Local Time: 08/21/2015 10:52:36.937 AM
Update Local Time: 08/21/2015 10:52:35.986 AM

Did you know?
Explorer Tip #1
OPC Explorer can export your current data.
Click For Details
MatrikonOPC

Group Info

Group: Group0

Connected (Async I/O): Yes (2.0)

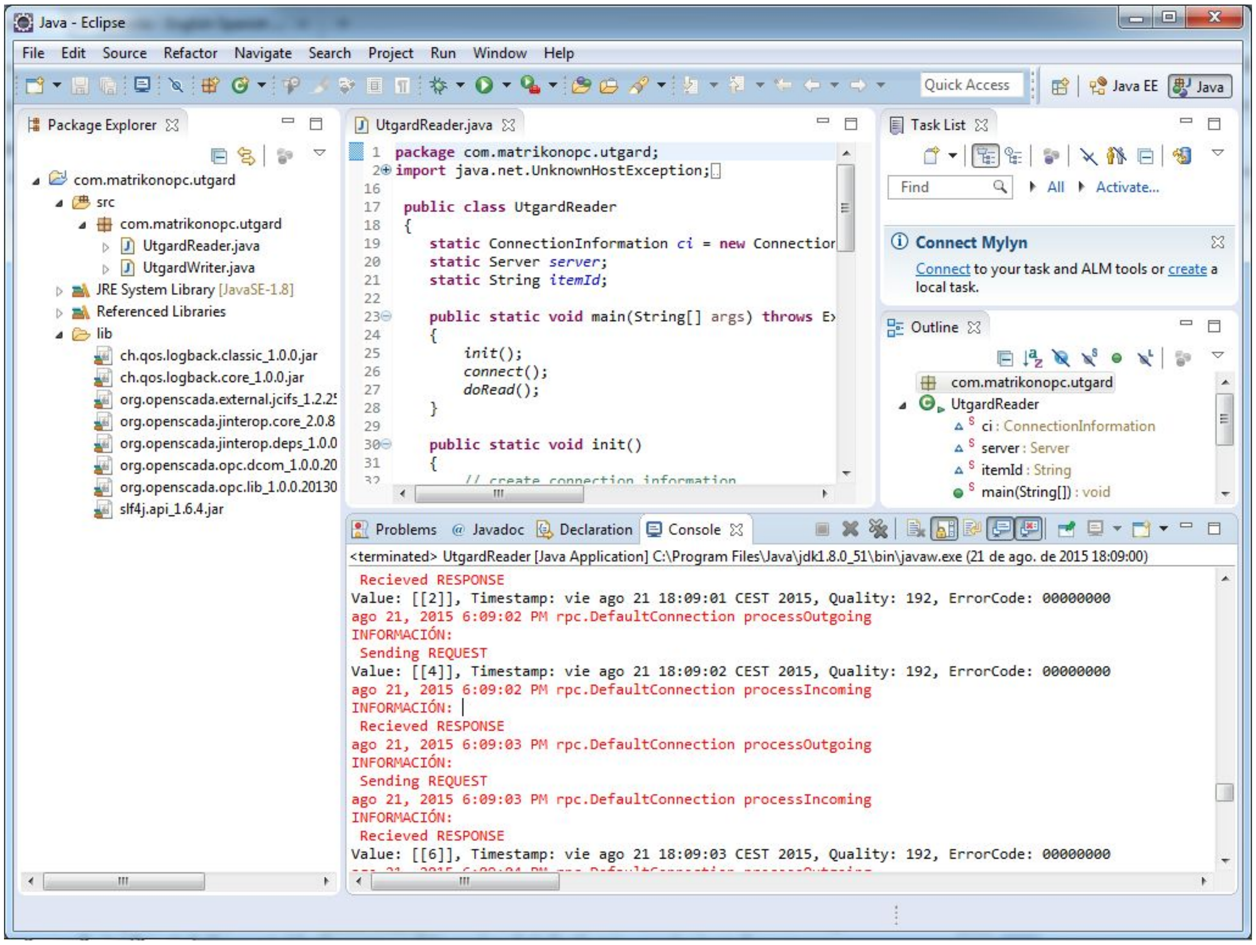
Active: Yes
Items: 1
Current Update Rate: 1000 ms
Percent Deadband: 0,00%
Data Change Rate: 0,94 Items/Sec

The following code summary will connect to the server and show its value each second (1000 ms). A detailed copy of it can be found at GitHub.

```
package com.matrikonopc.utgard;
import ...

 public class UtgardReader
 {
        static ConnectionInformation ci = new ConnectionInformation();
         static Server server;
         static String itemId;

        public static void main(String[] args) throws Exception
        {
         init();
         connect();
         doRead();
        }
  }
        public static void init() {...}

        public static void connect() throws IllegalArgumentException, UnknownHostException,
AlreadyConnectedException {...}

        public static void doRead() throws IllegalArgumentException, UnknownHostException,
NotConnectedException, JIException, DuplicateGroupException, AddFailedException,
InterruptedException {...}
```

The output looks like

```
...snip...

ago 21, 2015 6:09:01 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:09:01 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
Value: [[2]], Timestamp: vie ago 21 18:09:01 CEST 2015, Quality: 192, ErrorCode: 00000000
ago 21, 2015 6:09:02 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
Value: [[4]], Timestamp: vie ago 21 18:09:02 CEST 2015, Quality: 192, ErrorCode: 00000000
ago 21, 2015 6:09:02 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
ago 21, 2015 6:09:03 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:09:03 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
Value: [[6]], Timestamp: vie ago 21 18:09:03 CEST 2015, Quality: 192, ErrorCode: 00000000
ago 21, 2015 6:09:04 PM rpc.DefaultConnection processOutgoing

...snip...
```


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
com.matrikonopc.utgard
src
com.matrikonopc.utgard
UtgardReader.java
UtgardWriter.java
JRE System Library [JavaSE-1.8]
Referenced Libraries
lib
ch.qos.logback.classic_1.0.0.jar
ch.qos.logback.core_1.0.0.jar
org.openscada.external.jcifs_1.2.2!
org.openscada.jinterop.core_2.0.8
org.openscada.jinterop.deps_1.0.0
org.openscada.opc.dcom_1.0.0.20
org.openscada.opc.lib_1.0.0.20130
slf4j.api_1.6.4.jar
UtgardReader.java
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
com.matrikonopc.utgard
UtgardReader
ci : ConnectionInformation
server : Server
itemId : String
main(String[]) : void
Problems
Javadoc
Declaration
Console

```java
package com.matrikonopc.utgard;
import java.net.UnknownHostException;

public class UtgardReader
{
    static ConnectionInformation ci = new Connectior
    static Server server;
    static String itemId;

    public static void main(String[] args) throws E>
    {
        init();
        connect();
        doRead();
    }

    public static void init()
    {
        // create connection information
```

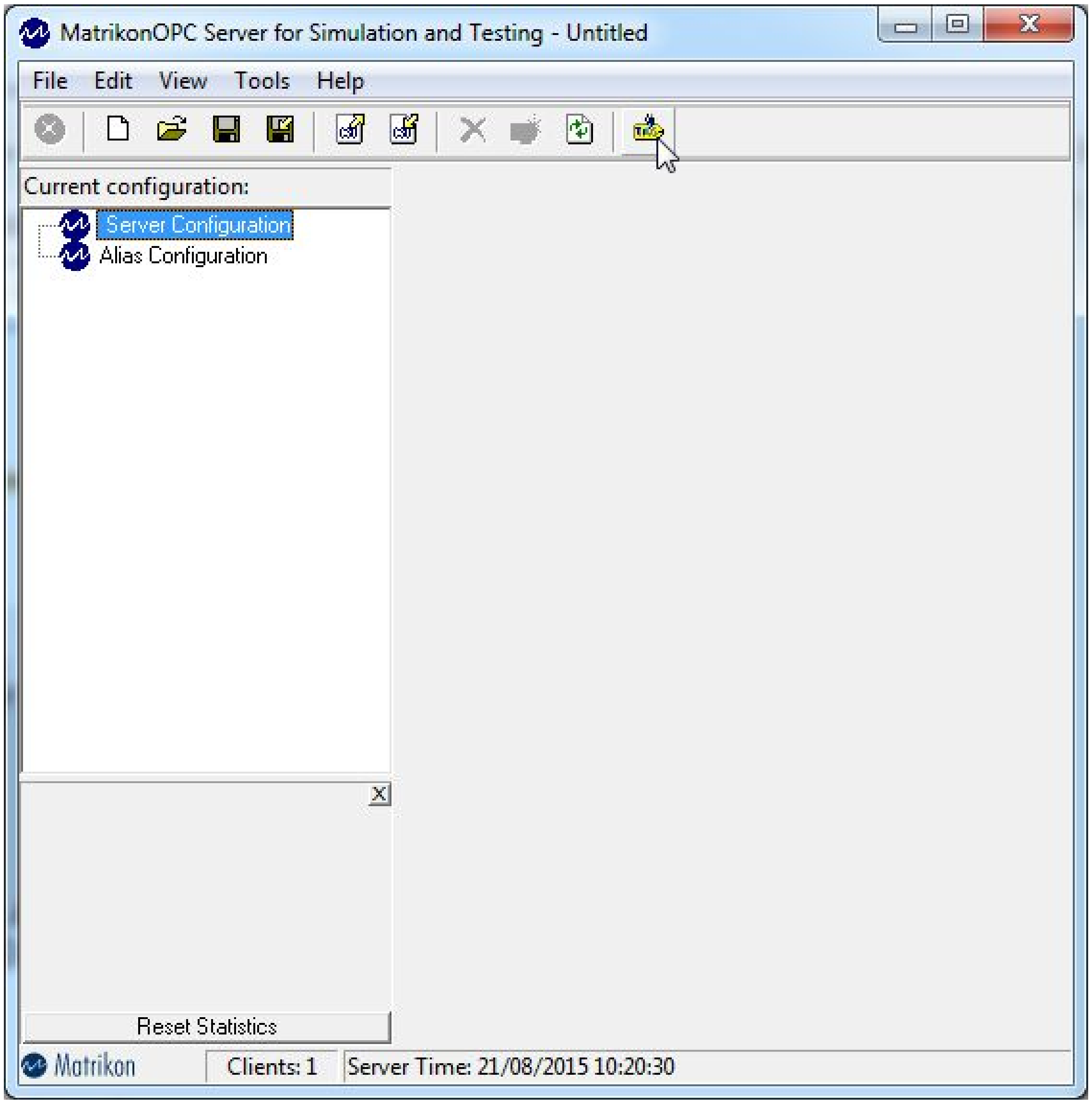
```
<terminated> UtgardReader [Java Application] C:\Program Files\Java\jdk1.8.0_51\bin\javaw.exe (21 de ago. de 2015 18:09:00)
 Recieved RESPONSE
Value: [[2]], Timestamp: vie ago 21 18:09:01 CEST 2015, Quality: 192, ErrorCode: 00000000
ago 21, 2015 6:09:02 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
Value: [[4]], Timestamp: vie ago 21 18:09:02 CEST 2015, Quality: 192, ErrorCode: 00000000
ago 21, 2015 6:09:02 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
ago 21, 2015 6:09:03 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:09:03 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
Value: [[6]], Timestamp: vie ago 21 18:09:03 CEST 2015, Quality: 192, ErrorCode: 00000000
```

Writing tags

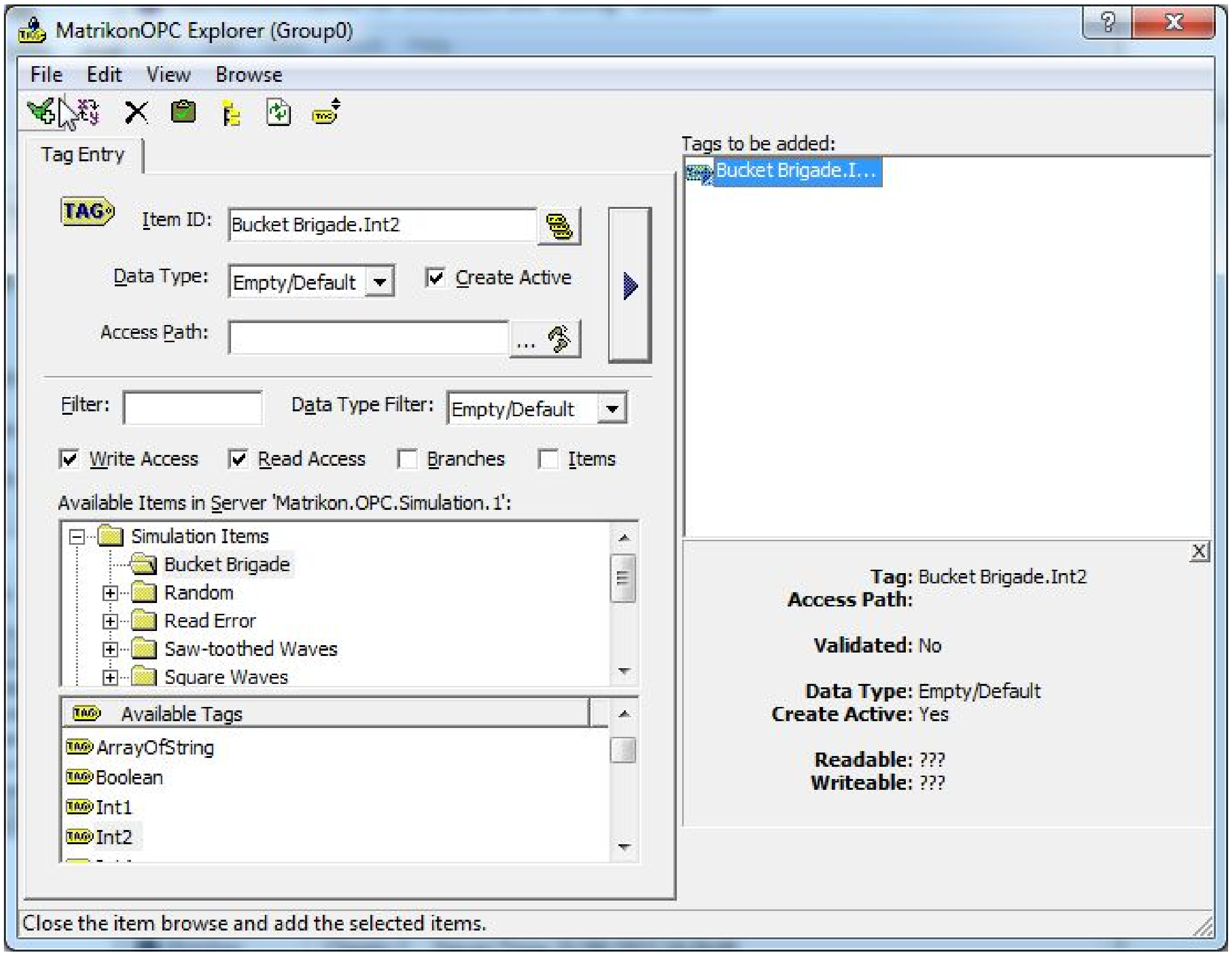
The first step is creating a writing access tag. According to the MatrikonOPC Server for Simulation and Testing user manual, the items Bucket Brigade are the only ones that can be written. Therefore, a tag Bucket Brigade.Int2 is created for the purpose of this section. Initially, the integer value tag is 0.

The required procedure comprises the following stages:

1. Open the server and click on the icon Tag of the icons bar.



2. A MatrikonOPC Explorer window opens. Select Simulation Items -> Bucket Brigade in the tree structure corresponding to "Available Items in Server 'Matrikon.OPC.Simulation.1'". Then, select Int2 in the drop-down menu corresponding to "Available Tags". In order to add this tag to the server, click on the first icon of the icons bar in this window.



Item ID: Bucket Brigade.Int2
Data Type: Empty/Default
Create Active: ✓
Data Type Filter: Empty/Default
Write Access: ✓
Read Access: ✓

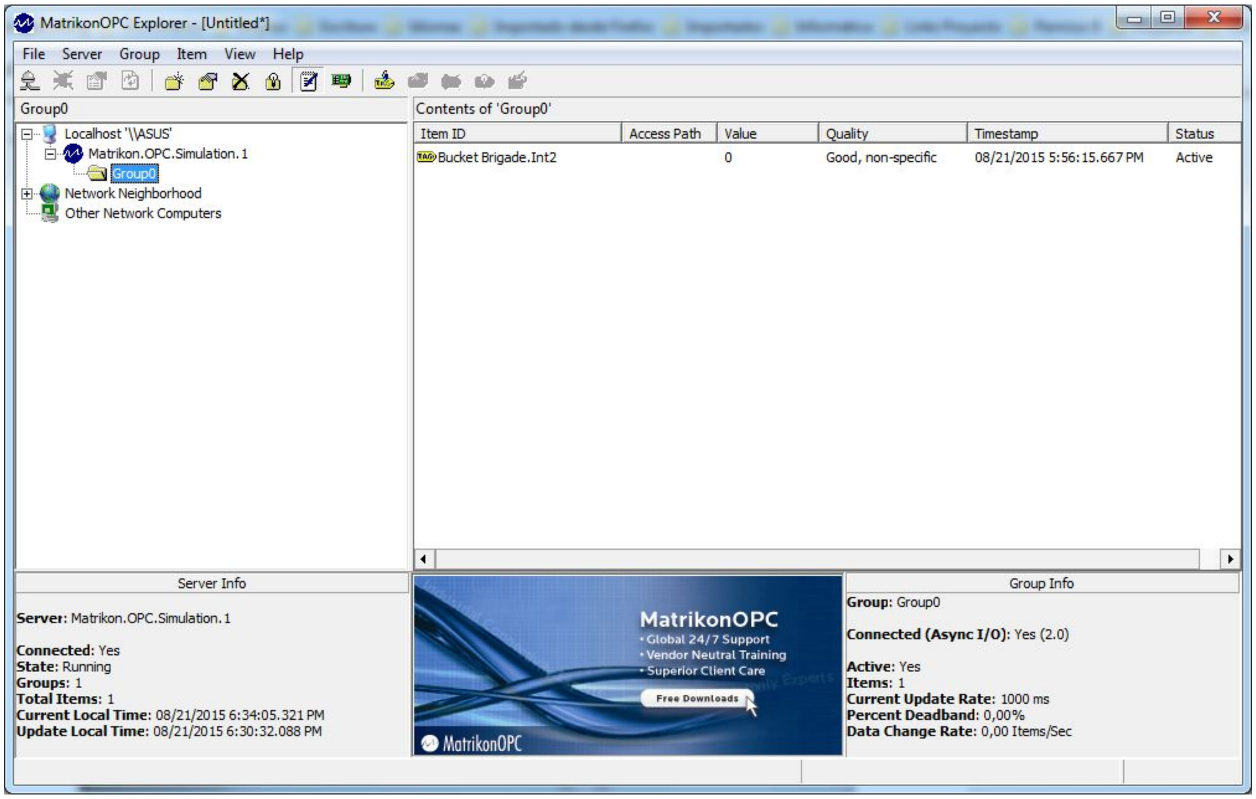
Tag: Bucket Brigade.Int2
Access Path:

Validated: No

Data Type: Empty/Default
Create Active: Yes

Readable: ???
Writeable: ???

Once the tag has been added, the MatrikonOPC Explorer is opened and shows the initial state of the tag.


MatrikonOPC Explorer - [Untitled*]
File Server Group Item View Help
MatrikonOPC
• Global 24/7 Support
• Vendor Neutral Training
• Superior Client Care
Free Downloads
MatrikonOPC

Group0

Localhost '\\ASUS'
Matrikon.OPC.Simulation.1
Group0
Network Neighborhood
Other Network Computers

Contents of 'Group0'

Item ID	Access Path	Value	Quality	Timestamp	Status
Bucket Brigade.Int2		0	Good, non-specific	08/21/2015 5:56:15.667 PM	Active

Server Info

Server: Matrikon.OPC.Simulation.1

Connected: Yes
State: Running
Groups: 1
Total Items: 1
Current Local Time: 08/21/2015 6:34:05.321 PM
Update Local Time: 08/21/2015 6:30:32.088 PM

Group Info

Group: Group0

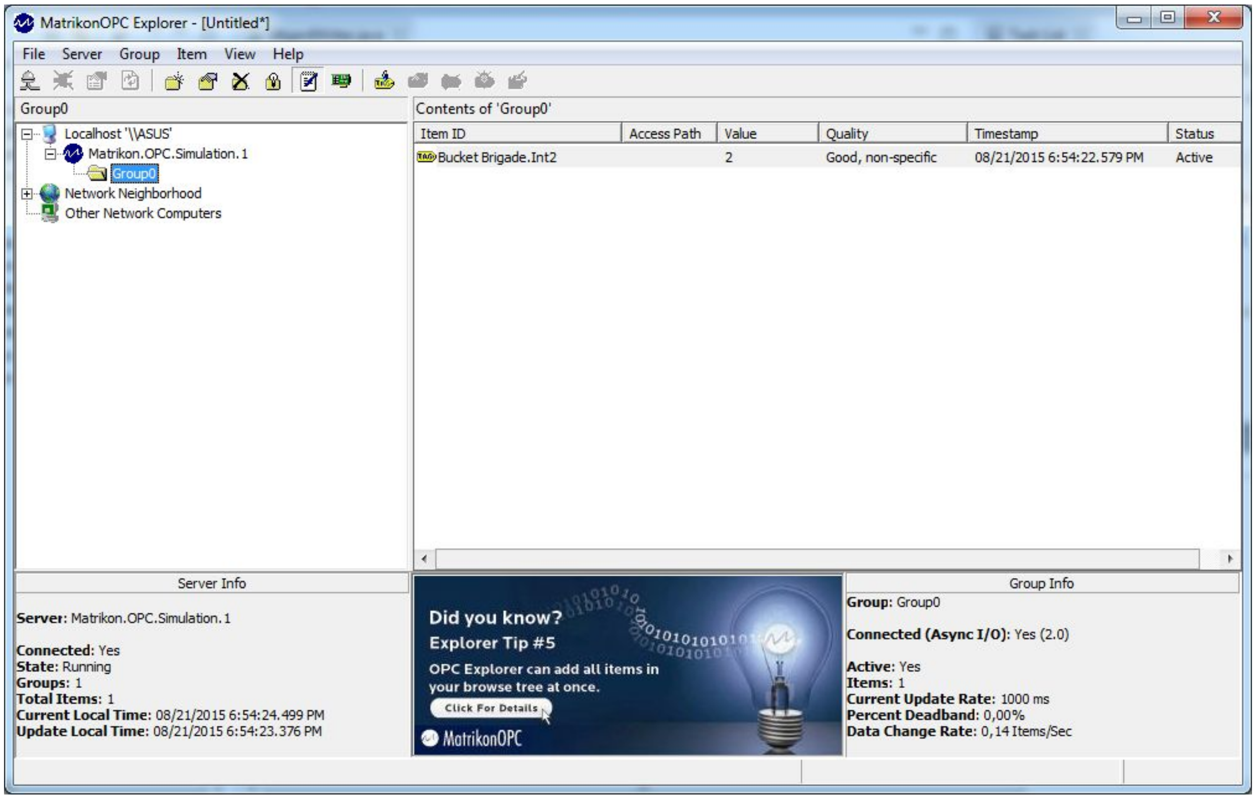
Connected (Async I/O): Yes (2.0)

Active: Yes
Items: 1
Current Update Rate: 1000 ms
Percent Deadband: 0,00%
Data Change Rate: 0,00 Items/Sec

The following code schema will write an increasing integer value into the tag each 3 seconds. A complete code description can be download from GitHub.

```
package com.matrikonopc.utgard;
import java.net.UnknownHostException;
import ...

public class UtgardWriter
{
    static ConnectionInformation ci = new ConnectionInformation();
         static Server server;
         static String itemId;

        public static void main(String[] args) throws Exception
        {
         init();
         connect();
         doWrite();
        }

        public static void init() {...}

        public static void connect() throws IllegalArgumentException, UnknownHostException,
AlreadyConnectedException       {...}

        public static void doWrite() throws InterruptedException, IllegalArgumentException,
UnknownHostException, NotConnectedException, JIException, DuplicateGroupException,
AddFailedException {...}
```
The corresponding output is shown below.

```
...snip...
 Recieved RESPONSE
<<< Value: [[0]], Timestamp: vie ago 21 18:54:05 CEST 2015, Quality: 192, ErrorCode: 00000000 /
value = 0
ago 21, 2015 6:54:19 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:54:19 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
<<< Value: [[1]], Timestamp: vie ago 21 18:54:19 CEST 2015, Quality: 192, ErrorCode: 00000000 /
value = 1
ago 21, 2015 6:54:20 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:54:20 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
<<< Value: [[1]], Timestamp: vie ago 21 18:54:19 CEST 2015, Quality: 192, ErrorCode: 00000000 /
value = 1
ago 21, 2015 6:54:20 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:54:20 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
<<< Value: [[1]], Timestamp: vie ago 21 18:54:19 CEST 2015, Quality: 192, ErrorCode: 00000000 /
value = 1
ago 21, 2015 6:54:21 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:54:21 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
<<< Value: [[1]], Timestamp: vie ago 21 18:54:19 CEST 2015, Quality: 192, ErrorCode: 00000000 /
value = 1
>>> writing value 2
...snip...
```


MatrikonOPC Explorer - [Untitled*]
File Server Group Item View Help
Group0
Localhost '\\ASUS'
Matrikon.OPC.Simulation.1
Group0
Network Neighborhood
Other Network Computers
Did you know?
Explorer Tip #5
OPC Explorer can add all items in your browse tree at once.
Click For Details
MatrikonOPC

Contents of 'Group0'

Item ID	Access Path	Value	Quality	Timestamp	Status
Bucket Brigade.Int2		2	Good, non-specific	08/21/2015 6:54:22.579 PM	Active

Server Info

Server: Matrikon.OPC.Simulation.1

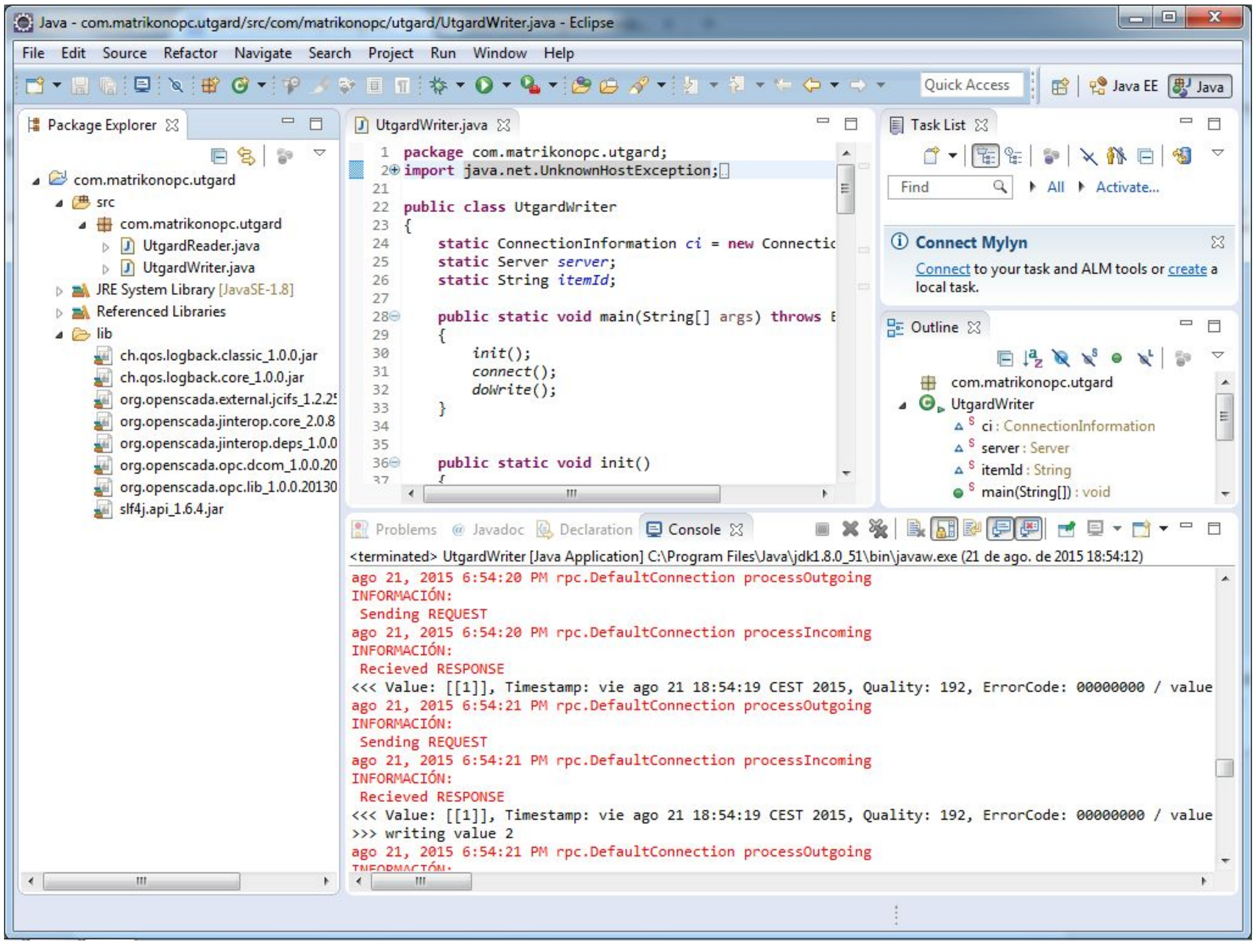
Connected: Yes
State: Running
Groups: 1
Total Items: 1
Current Local Time: 08/21/2015 6:54:24.499 PM
Update Local Time: 08/21/2015 6:54:23.376 PM

Group Info

Group: Group0

Connected (Async I/O): Yes (2.0)

Active: Yes
Items: 1
Current Update Rate: 1000 ms
Percent Deadband: 0,00%
Data Change Rate: 0,14 Items/Sec


Java - com.matrikonopc.utgard/src/com/matrikonopc/utgard/UtgardWriter.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
com.matrikonopc.utgard
src
com.matrikonopc.utgard
UtgardReader.java
UtgardWriter.java
JRE System Library [JavaSE-1.8]
Referenced Libraries
lib
ch.qos.logback.classic_1.0.0.jar
ch.qos.logback.core_1.0.0.jar
org.openscada.external.jcifs_1.2.2!
org.openscada.jinterop.core_2.0.8
org.openscada.jinterop.deps_1.0.0
org.openscada.opc.dcom_1.0.0.20
org.openscada.opc.lib_1.0.0.20130
slf4j.api_1.6.4.jar
UtgardWriter.java
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
com.matrikonopc.utgard
UtgardWriter
ci : ConnectionInformation
server : Server
itemId : String
main(String[]) : void
Problems
Javadoc
Declaration
Console

```
package com.matrikonopc.utgard;
import java.net.UnknownHostException;

public class UtgardWriter
{
    static ConnectionInformation ci = new Connectic
    static Server server;
    static String itemId;

    public static void main(String[] args) throws E
    {
        init();
        connect();
        doWrite();
    }


    public static void init()
    {
```

```
<terminated> UtgardWriter [Java Application] C:\Program Files\Java\jdk1.8.0_51\bin\javaw.exe (21 de ago. de 2015 18:54:12)
ago 21, 2015 6:54:20 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:54:20 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
<<< Value: [[1]], Timestamp: vie ago 21 18:54:19 CEST 2015, Quality: 192, ErrorCode: 00000000 / value
ago 21, 2015 6:54:21 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
 Sending REQUEST
ago 21, 2015 6:54:21 PM rpc.DefaultConnection processIncoming
INFORMACIÓN:
 Recieved RESPONSE
<<< Value: [[1]], Timestamp: vie ago 21 18:54:19 CEST 2015, Quality: 192, ErrorCode: 00000000 / value
>>> writing value 2
ago 21, 2015 6:54:21 PM rpc.DefaultConnection processOutgoing
INFORMACIÓN:
```

Summary and further steps

In this paper a brief introduction and presentation of the concepts related to OPC has been made. OPC solution allows to resolve the custom driver problem, when devices of different vendors coexist in the same working environment, and facilitate the interoperability between them.
Afterwards, two current examples of OPC technology usage have been presented plainly.
Finally, a case study in which the OPC connectivity between a Client and an Server have been carried out. For the OPC Server, the chosen option is MatrikonOPC Server for Simulation and Testing. For the client part, it is used a Java-based implementation called Utgard.

Once this practical example has been succeeded, the later steps should be addressed to the deployment of real devices (PLCs for examples) connected to OPC Servers and the monitoring of real variables through OPC Clients.