SCADA & OPC-UA Report

Seyedsina Miri
256347

Factory Information Systems

March 12, 2016

Establish connection between Ignition and SQL DB:

- **setup details**

First, the ignition software was updated to its latest version which is 7.8.2 in order to have its optimum functionality. After installation completed, on the browser at http://localhost:8088 under the configuration tab, a new connection to database can be established through connections (under Databases) in left panel. The same can also be done in the home tab by simply clicking on "connect to a database".

Before the connection is created, there should be a database on the machine that the ignition has its drive. For the case of this assignment, MySQL database is chosen as its workbench had already been installed on my machine. Server status of the database is shown in the screenshot below:

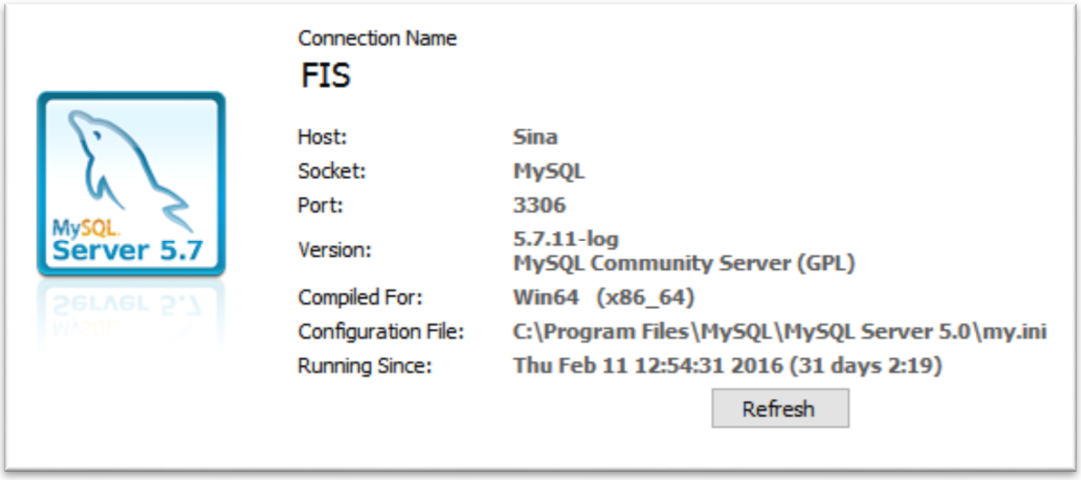

Figure 1: status of the MySQL server used for connecting to ignition

- **configuration details with decisions**

Now the connection between ignition and the database can be created. After clicking on "connect to a database" in home tab of ignition, the following page appears:

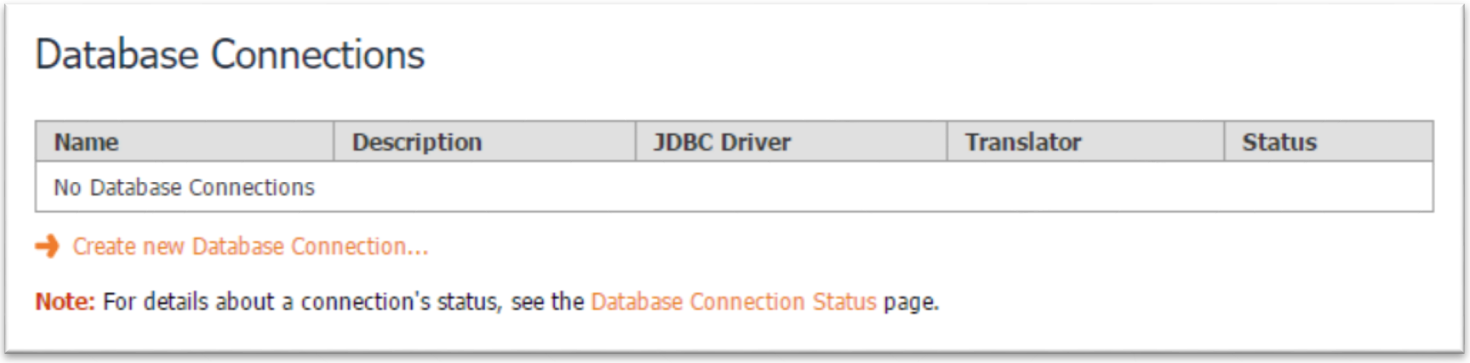

Figure 2: a connection to database is created in this page

In order to create connection, "Create new Database Connection…" is clicked which leads to the following page:

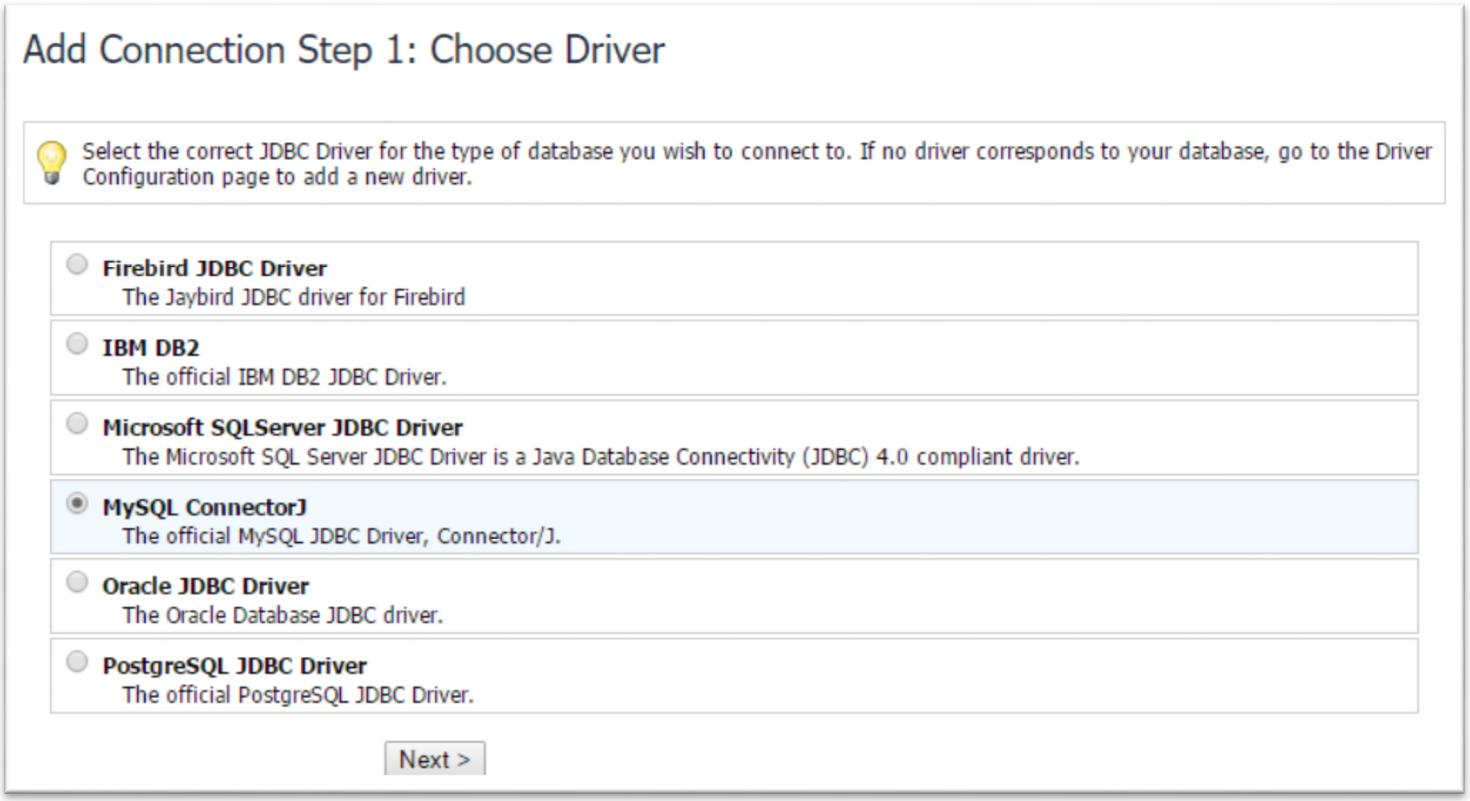

Figure 3: database drive is chosen on this page

As we have chosen MySQL as our database, the appropriate drive for it is selected in this page. Then, "Next" button is pushed and in the following page, name for connection, connection URL, username and password are defined.

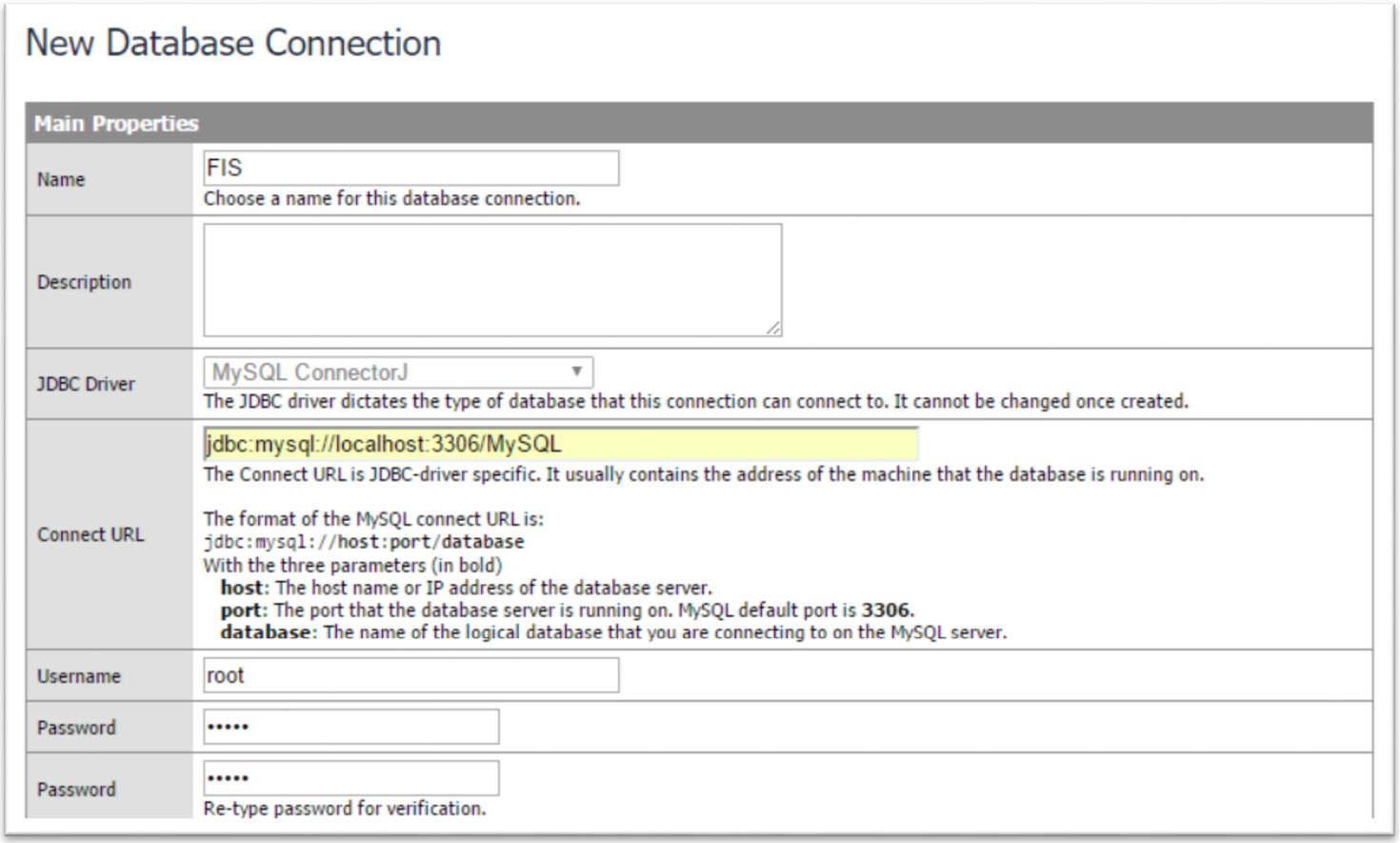

Figure 4: necessary fields for creating the connection are filled

After filling the fields with appropriate values, the connection is created. The following screenshot shows that the database is successfully created and is valid.

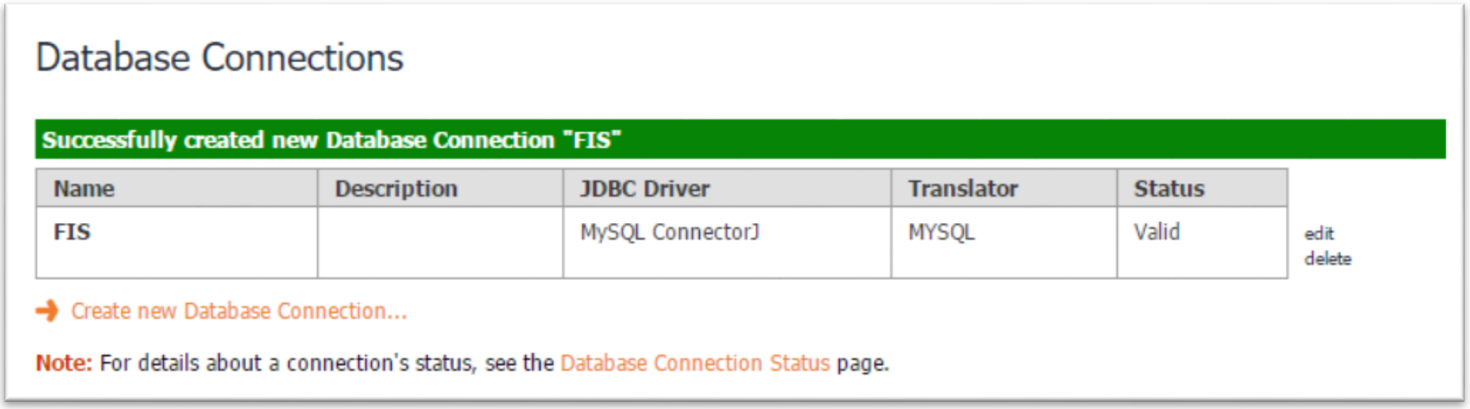

Figure 5: a valid database is created

Establish connection between Ignition and OPC-UA Server:

- **setup details**

In order to create an OPC-UA server, node.js has to be installed. Then, using command prompt, in the directory where we want to have our server, should enter “npm init” in order to the package.jason, then “npm install node-opcua --save” to add the node-opcua module. This node module, allows us to create an OPC-UA server.

Then, a script with some name e.g. server.js is created in which the code for OPC-UA server is written.

- **programming details with decisions**

In the script created for OPC-UA server, the programming is done. In order to do so, with help from tutorial in http://node-opcua.github.io the programming for creation of server is done as follows:

First the node-opcua SDK is included in the server:

```
1  // declaratin
2  var opcua = require("node-opcua"); //node-opcua sdk is included
```

Then, for a simple server, the TCP port for listening socket of the server is decided to be 4334 and resourcePath that is a path added to endpoint resource name is set to be UA/OPC-UAServer. OPC-UA server instance is created as follows:

```
4  // server instantiation
5  var server = new opcua.OPCUAServer({
6      port: 4334, // TCP port is 4334
7      resourcePath: "UA/OPC-UAServer", // used for creating endpoint resource name
8  });
```

Next step is to initialize the server. To do so, initialize has an argument that is comprised of other components e.g. server.start( ) to start the server.

```
17 // server initialisation
18 server.initialize(post_initialize);
19
20 // server initialisation argument
21 function post_initialize() {
22     var endpointUrl = server.endpoints[0].endpointDescriptions()[0].endpointUrl;
23     console.log('endpoint Url is: ', endpointUrl);
24     server.start();
25     constructAddressSpace(server);
26     console.log("server initialized");
27 }
```

In the “post_initialize()” there is another functions that is not specified yet. The “constructAddressSpace()” function is later used for defining device and variables. This function is explained in the task for creating variables.

The variable “endpointUrl” is used for printing the endpoint URL in console:

- **configuration details with decisions**

In order to establish connection between OPC-UA server and ignition, first the server has to be executed by typing “node server.js” in console:

```
C:\Users\Sina\Documents\Master's\Factory Information Systems\Assignment 3\LECTURE8_OPCUA_SERVER>node server.js
 Server with max connections  10
endpoint Url is:  opc.tcp://SINA:4334/UA/OPC-UAServer
server initialized
```

Now the server is running properly, so it can be connected to ignition. To do so, on the browser at http://localhost:8088 under the configuration tab, under “OPC Connections” through “Servers” in the following page, connection can be created:

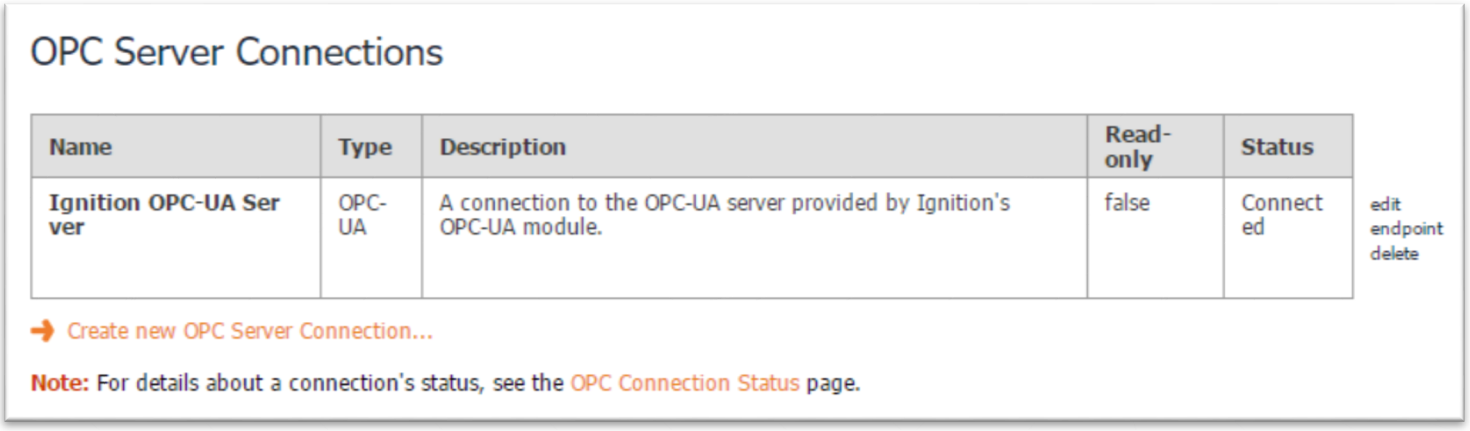

Figure 6: new OPC Server connection can be created on this page

Now by clicking on “Create new OPC Server Connection…” a new connection can be created as follows:

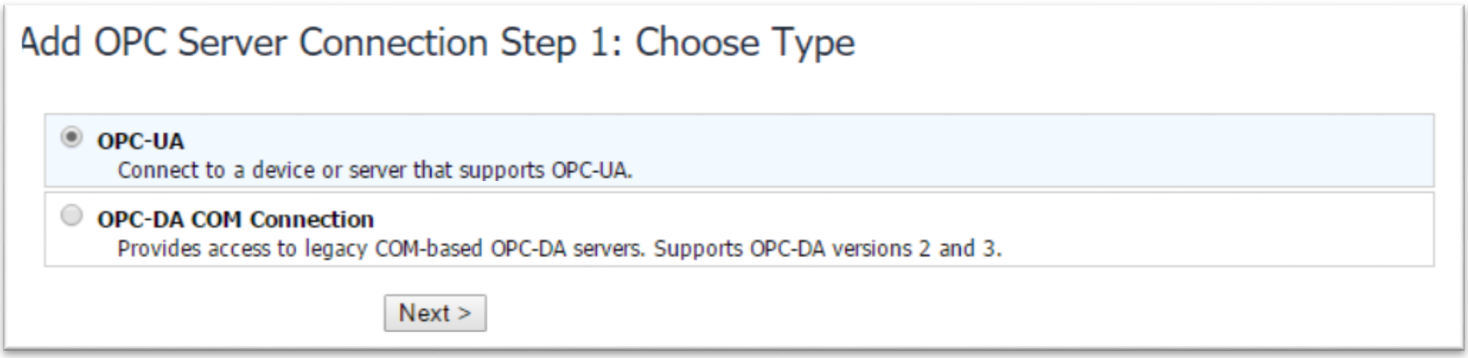

Figure 7: first step in creating server connection

After pushing “next” button in the page, the next page for discovering endpoint URL is shown as follows:

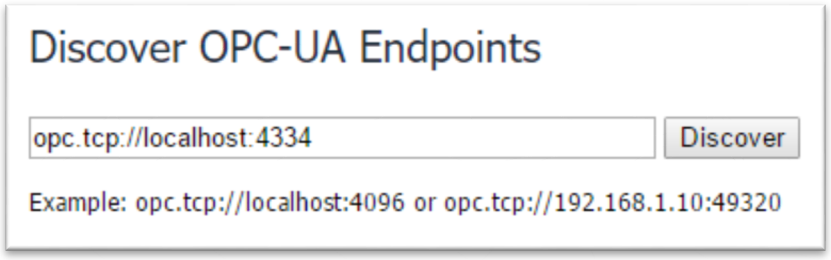

Figure 8: by clicking "discover", endpoint URL is discovered

Then OPC-UA endpoint URL is discovered, then the connection without security is chosen in this case:

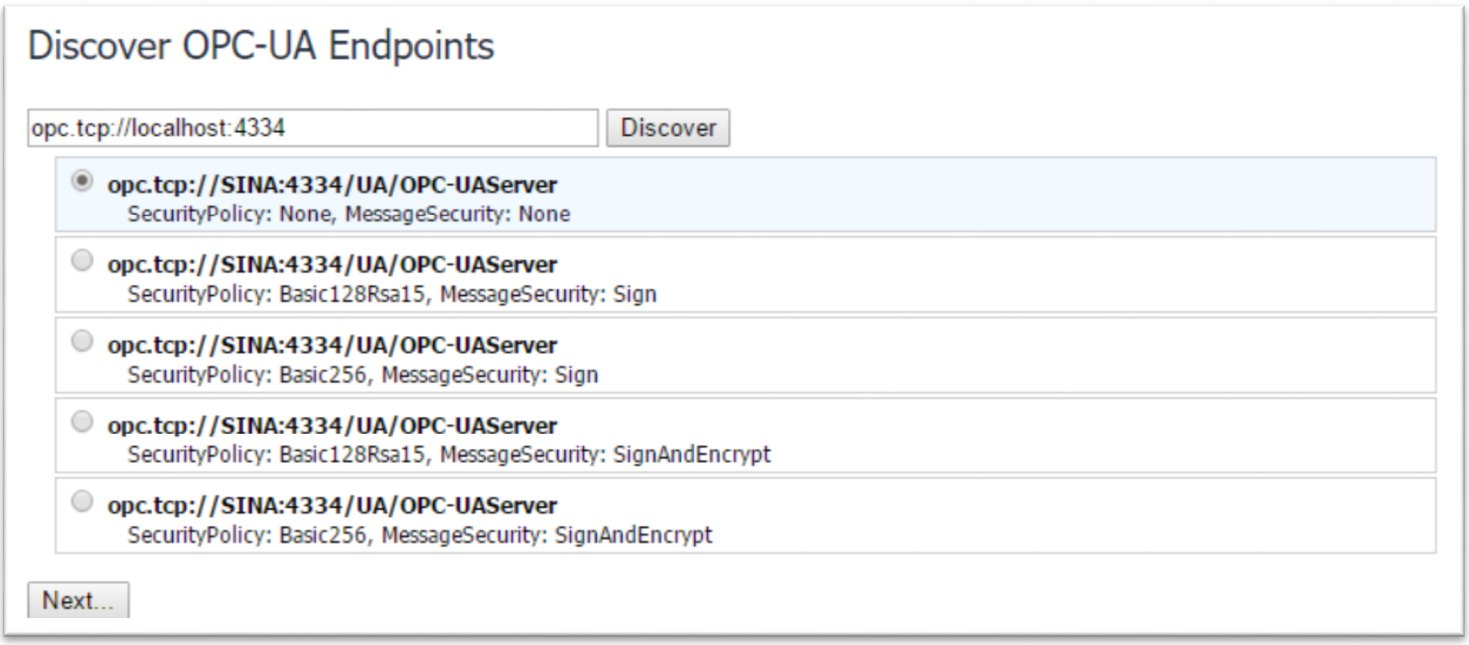

Figure 9: security policy and message security is chosen for the discovered endpoints

Then in the next page, other settings are entered as follows:

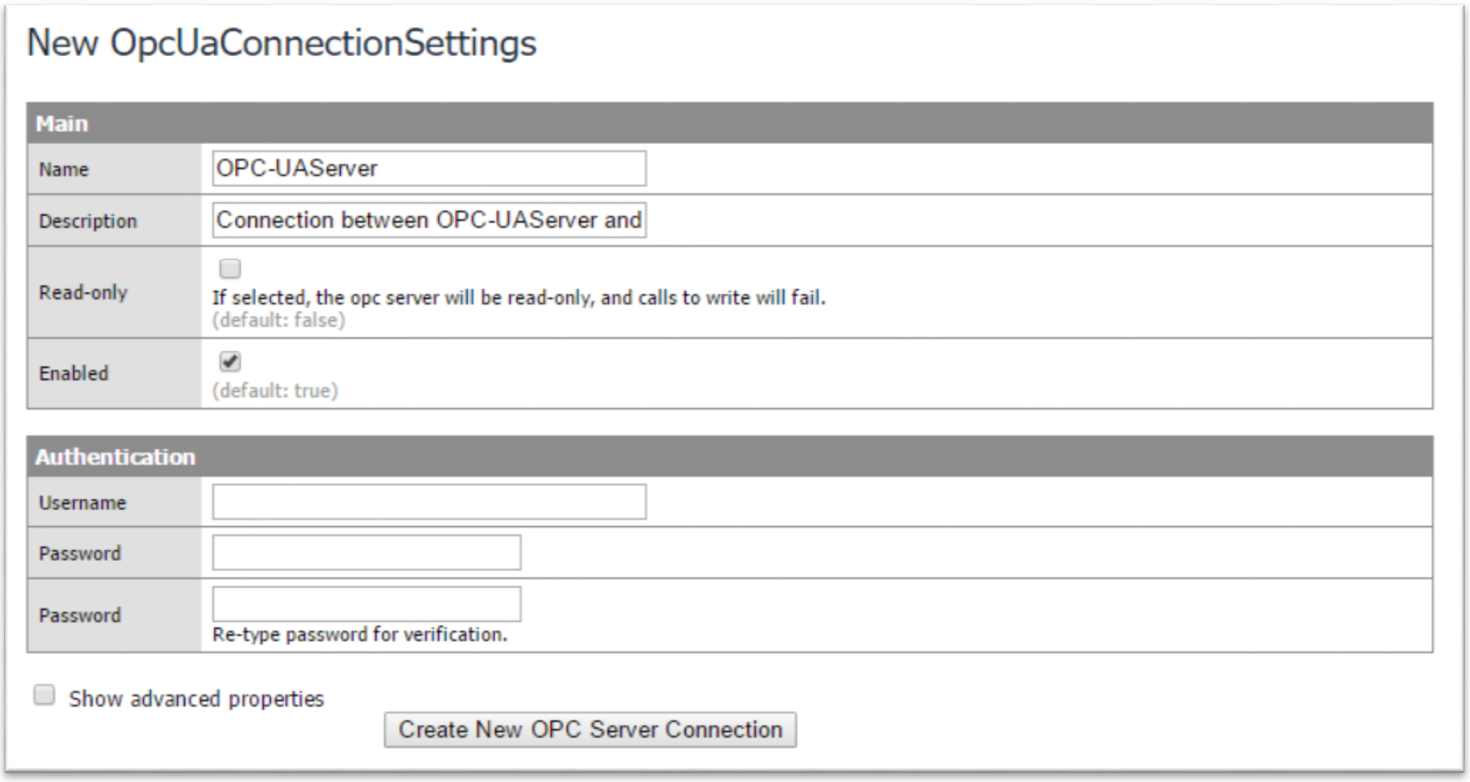

Figure 10: other settings for creating connection between OPC server and ignition

No authentication is specified in this case. Then, the connection is successfully created:

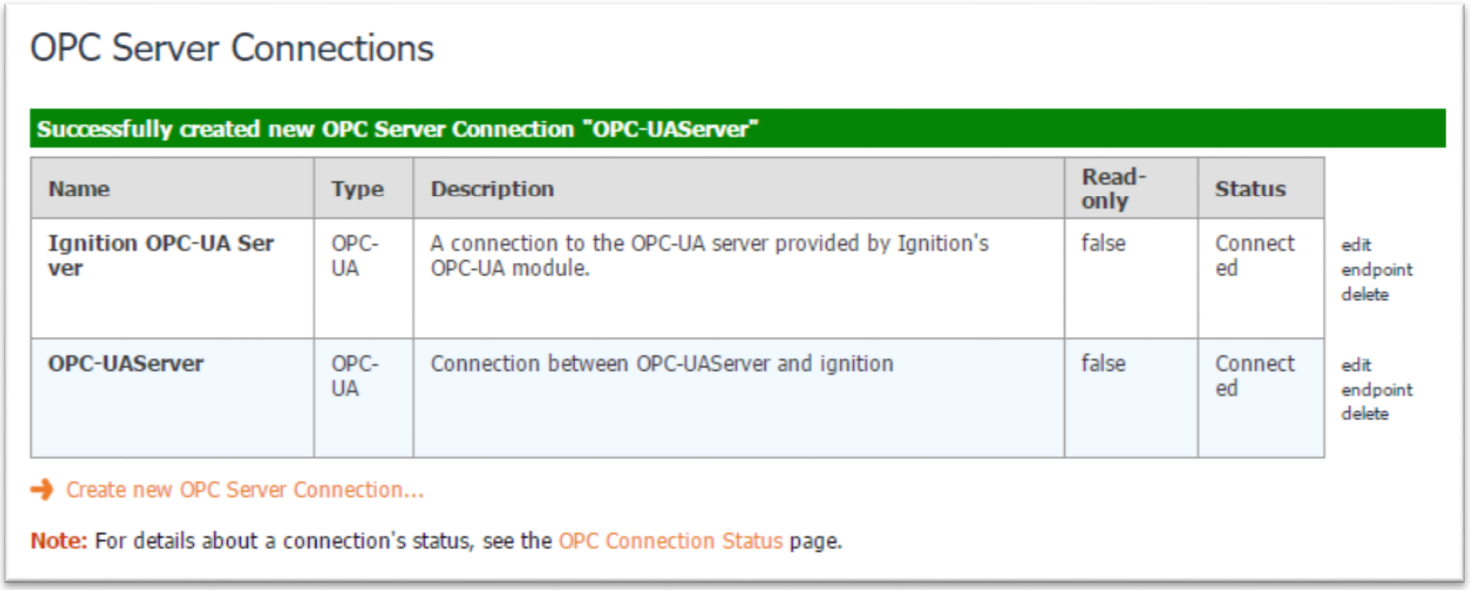

Figure 11: connection is successfully created

Create OPC-UA Server with read-write variable:

- **programming details with decisions**

Next step is to extend our default namespace with a device and several variables. This task is done in a function named "constructAddressSpace(server)" and is used when the server is being initialized.

First the device is created as follows:

```
29 // extending default namespace with a device and variables
30 function constructAddressSpace(server) {
31
32     var addressSpace = server.engine.addressSpace;
33
34     // creating a new folder in root folder for device
35     var device = addressSpace.addFolder("ObjectsFolder", {browseName: "Device"});
36
```

As it can be seen in the figure above, a new folder is created in the default namespace which its name is "Device" and serves as a device containing variables.

Then, two different variables that one produces random numbers indicating "temperature" and the other creating periodic numbers as "pressure" are assumed in the device.

In the figure below, codes are shown for creating a variable that produces some arbitrary random numbers above 50. This variable is denoted as "temperature" and is a component of previously created device.

```
38     // adding variable in the created device
39     var variable1 = 50.0;
40
41     server.nodeVariable1 = addressSpace.addVariable({
42         componentOf: device,
43         browseName: "temperature",
44         dataType: "Double",
45         value: {
46             get: function () {
47                 var t = Math.random() * 3.0;
48                 var value = variable1 + Math.exp(t);
49                 return new opcua.Variant({dataType: opcua.DataType.Double, value: value});
50             }
51         }
52     });
```

Then, another variable indicating "pressure" that creates periodic numbers between 100 and 120 is created and shown in the figure below:

```
    var variable2 = 110.0;

    server.nodeVariable2 = addressSpace.addVariable({
        componentOf: device,
        browseName: "Pressure",
        dataType: "Double",
        value: {
            get: function () {
                var p = new Date() / 10000.0;
                var value = variable2 + 10.0 * Math.sin(p);
                return new opcua.Variant({dataType: opcua.DataType.Double, value: value});
            }
        }
    });
}
```

- **modifying and testing variables**

Variables can be tested and modified until they can be used for the purpose of this assignment. It is important to note that the variables previously created (temperature and pressure) are modified to produce some reasonable values.

In order to test and modify the variables, first the OPC-UA server containing a device and variables has to be executed. Then, on the browser at http://localhost:8088 after the connection between OPC-UA server and ignition is established, in configuration tab, "Quick Client" can be used as a tool for reading and testing the variables.

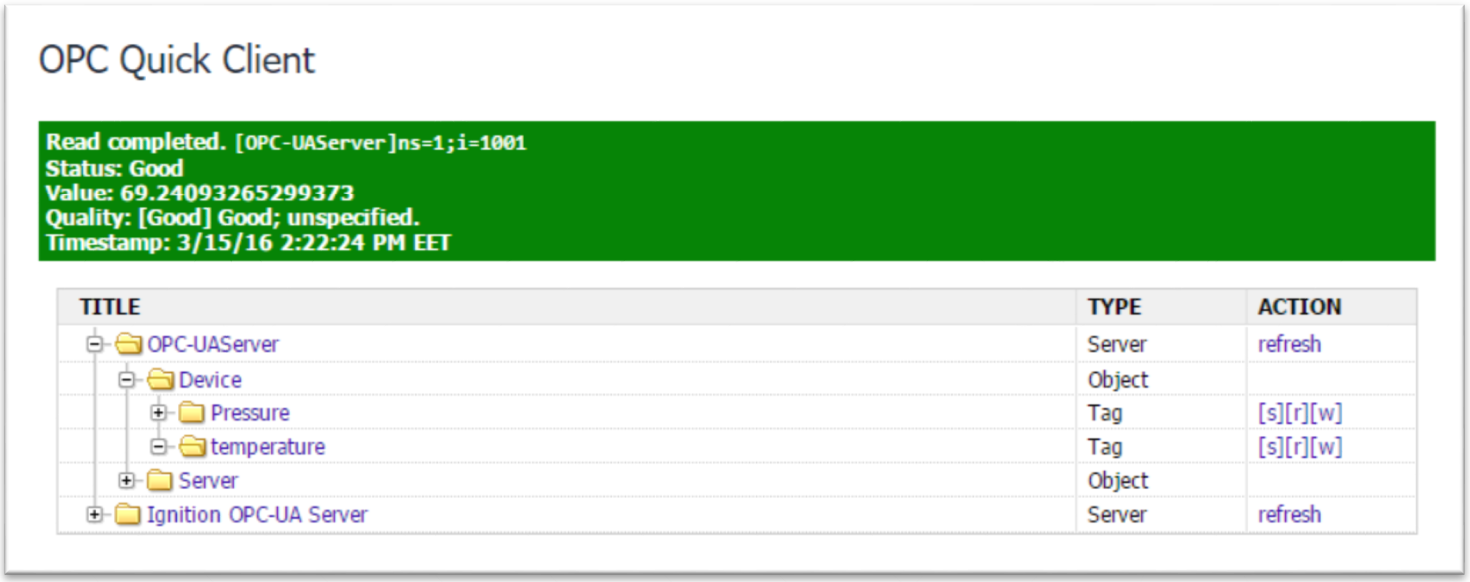

Figure 12: reading the variable "temperature" in Quick Client for testing

It is also possible to write variables as it is shown in the figure below:

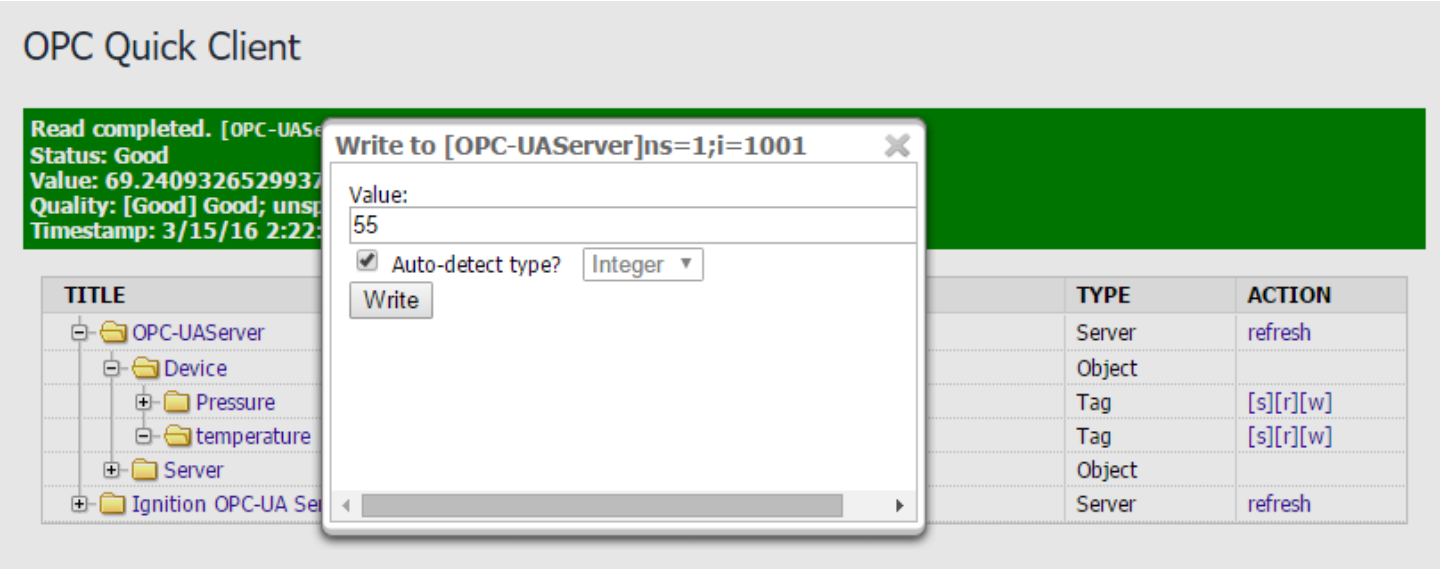

Figure 13: it is possible to write some values for variable in Quick Client

After testing the variables, if the values are not suitable, modifications can be done in the OPC-UA server. Otherwise, it is possible to proceed to next step and create a UI connected to variables.

Create UI connected to the variable (tag):

- **configuration details with decisions**

For creating the UI, Ignition designer has to run. Then, a new project with its name being "FIS_assignment3" and its title "SCADA&OPC-UA" is created. In the opened window, on the top left panel (Project Browser), the "User management window" under "Main windows" is selected for placing the UI elements.

As we save and publish the window, the project is created in the home tab of http://localhost:8088. Then, the project can be launched from there.

Figure 14: SCADA &OPC-UA is the project that is created and it can be launched

- **screenshot of UI element configurations explained**

UI elements are brought to the user management window by dragging and dropping from the right panel. For the variables created previously, two UI elements are chosen; one that serves as a Thermometer for visualizing temperature and the other Barometer used for pressure. A screenshot of the UI elements is shown below:

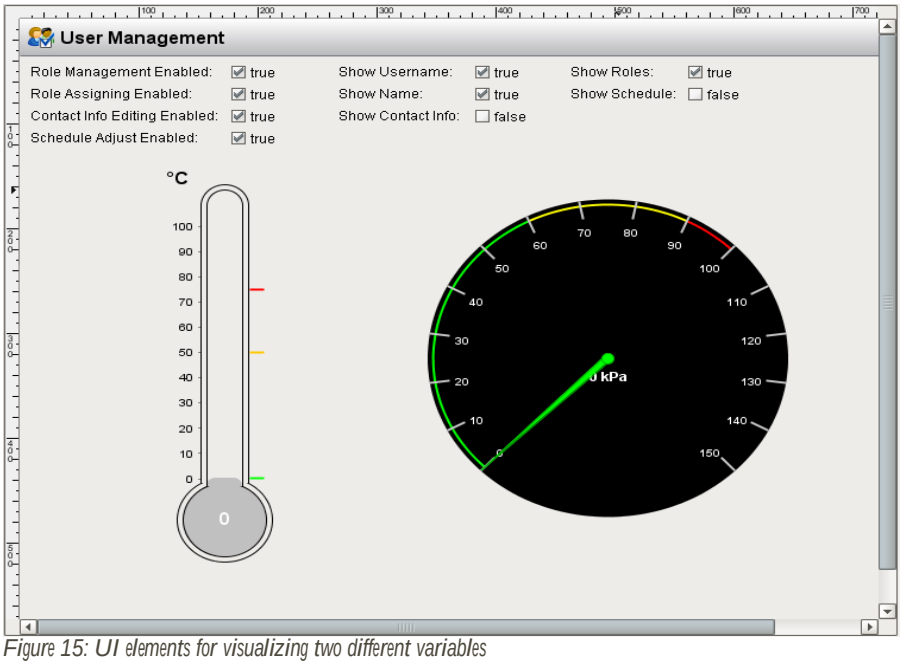

Figure 15: UI elements for visualizing two different variables

The UI element on the left is a thermometer that ranges from 0 to 100 and is suitable for visualizing the “temperature” values. The other element on the right is modified to serve as a Barometer, and its upper bound is extended to 150. The configuration for this element is done as follows:

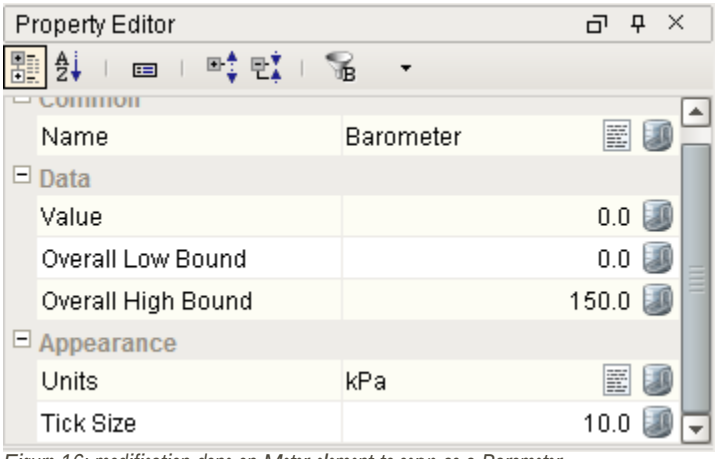

Figure 16: modification done on Meter element to serve as a Barometer

Now, this element is also suitable for visualizing pressure which ranges between 100 and 120.

Next step is create tags for two available variables and connect them to the UI elements. In order to create a new tag, by right clicking on tags (tag browser in the left panel) a new tag and then OPC tag is selected that leads to the following window in which appropriate values for Name, OPC Server and OPC item path have to be selected:

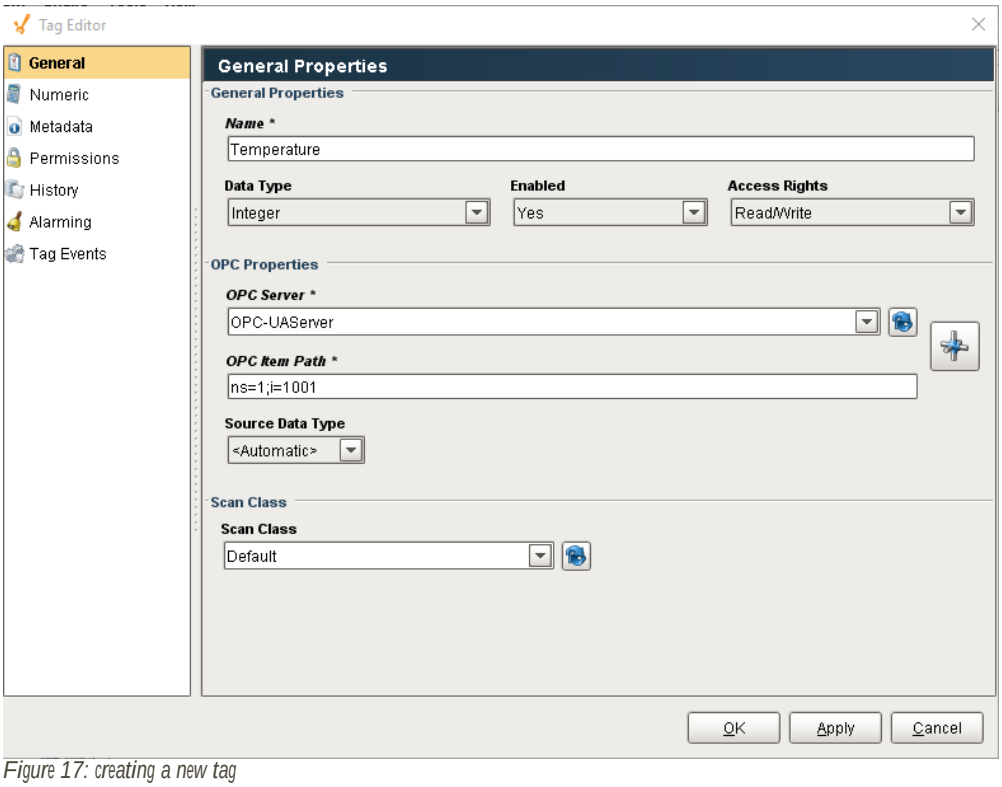

Figure 17: creating a new tag

Then, by applying and pressing OK, a new tag indicating temperature variable, is created. The same is also done for the second variable which is pressure. Then, two tags are available in the tag browser:

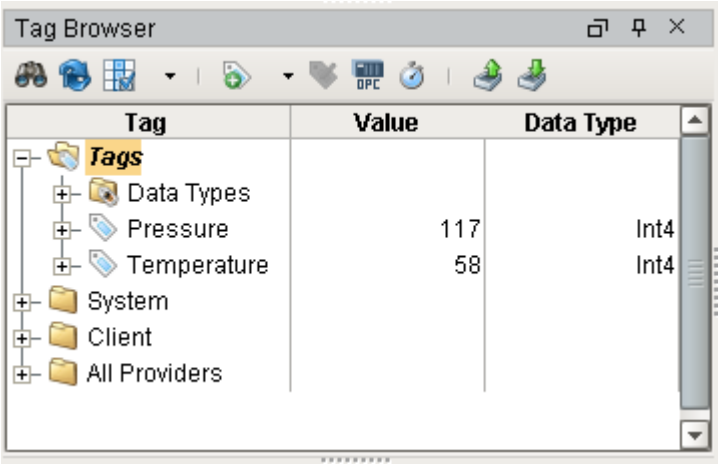

Figure 18: two tags created based on variables

- **screenshot of UI running**

  Finally the tags created based on two variables, can be connected to UI elements by simply dragging and dropping the tags on the corresponding UI elements. Then, UI starts running and showing the values created in OPC-UA server. The project is saved and published and can be lunched form browser (http://localhost:8088).

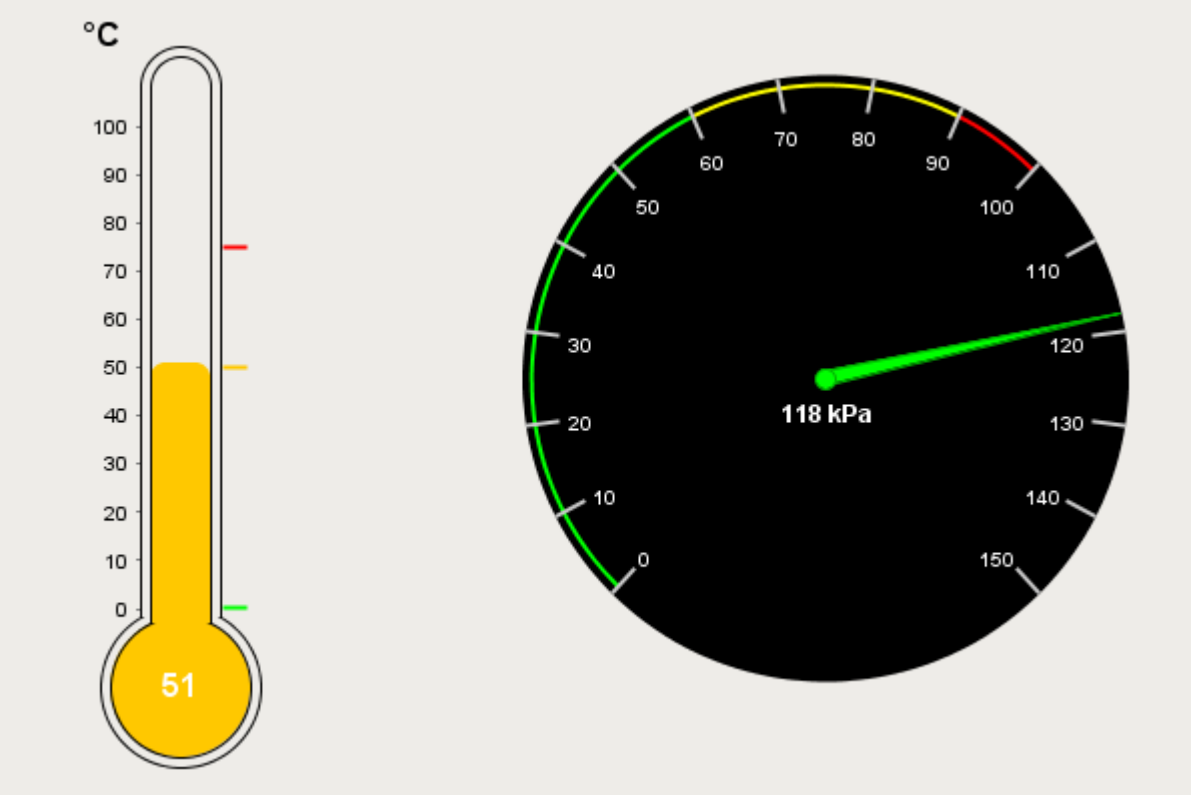

Figure 19: screenshot of UI running

Plot variable history:

- **configuration details with decisions**

In order to plot the variable history, in the ignition designer, from the right panel, an easy chart is dragged and dropped to the user management window. Then, by double clicking on each variable e.g. temperature, in the history tab, recording the data history is commanded as follows:

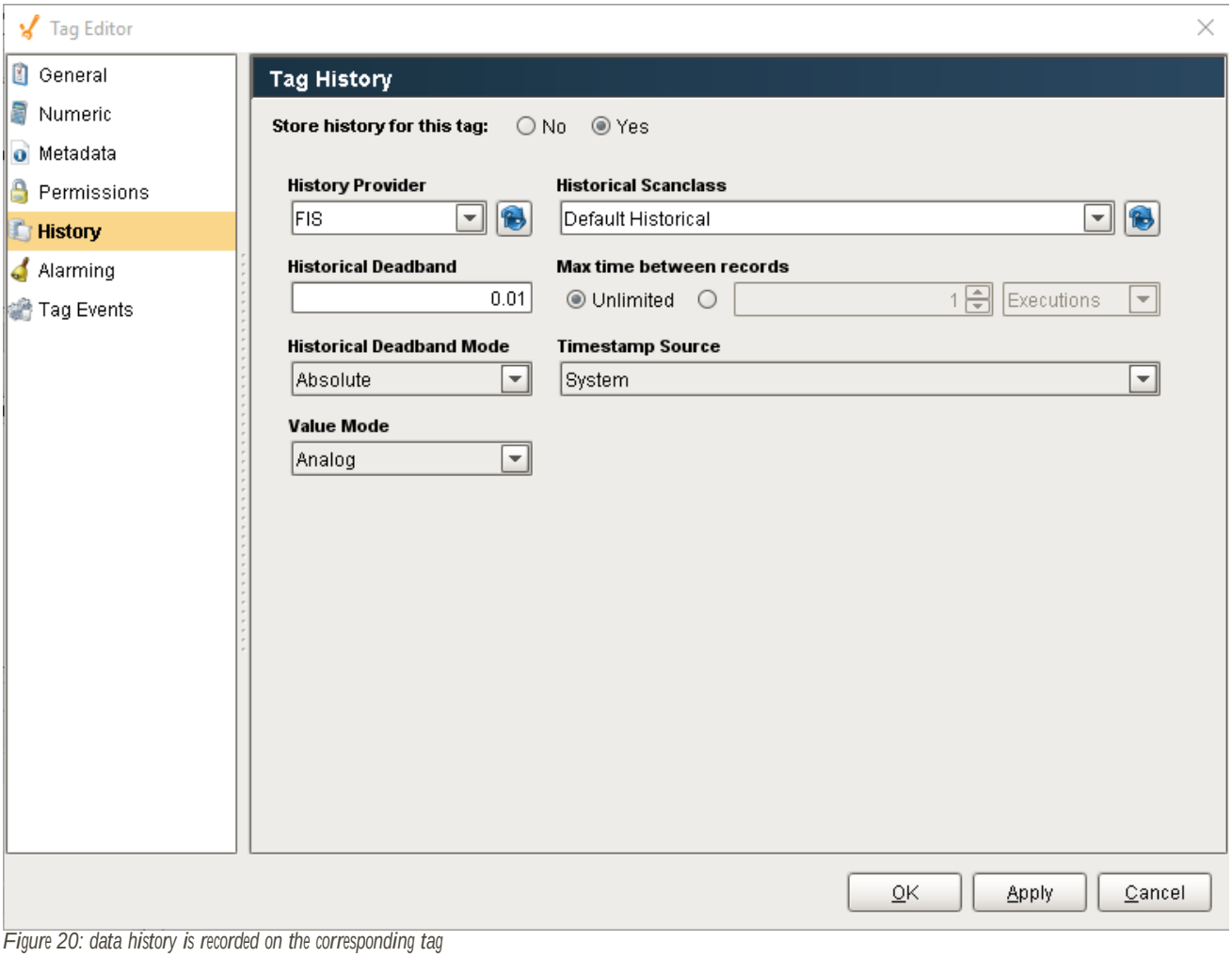

Figure 20: data history is recorded on the corresponding tag

The same is also done for the other variable (pressure). Then, the data history is being recorded and entered into the database.

- **screenshot of UI element configurations explained**

The UI element is easy chart element and is created by dragging and dropping it from the right panel.

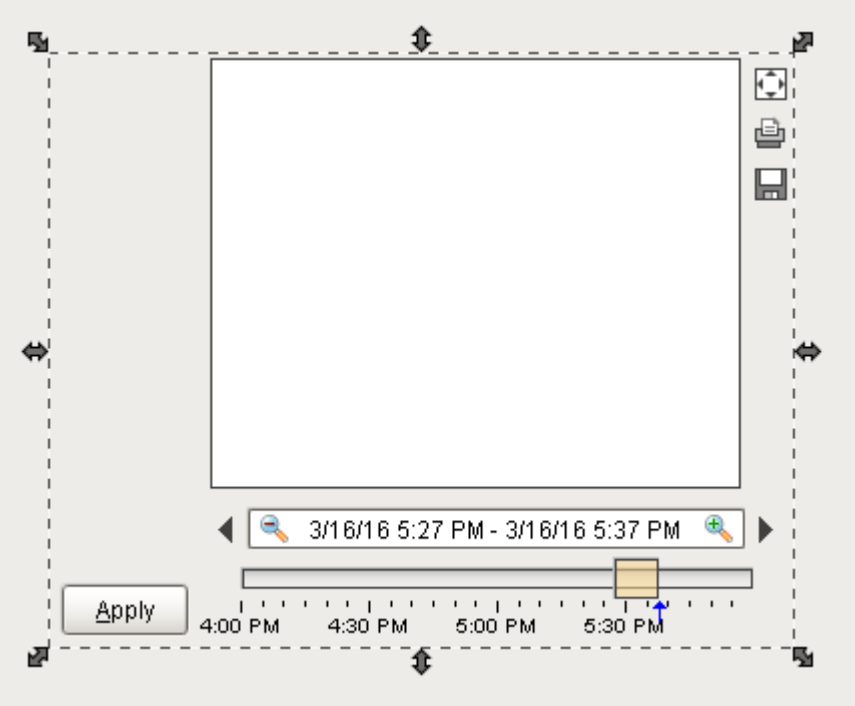

Figure 21: The UI element that is going to be used for visualizing data history

- **screenshot of UI running**

To visualize the data, the tags for corresponding variables are dragged and dropped to the easy chart visualization. Then, by launching the project, the UI used for recording the data starts running:

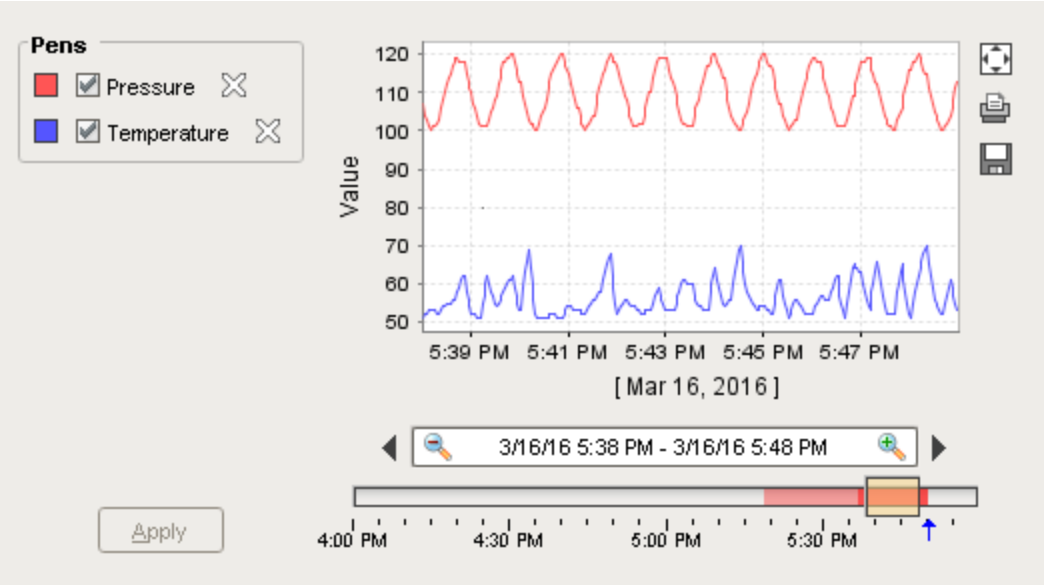

Figure 22: historical data recorded for two variables

In the chart shown above, the red graph indicates “pressure” and the blue one “temperature”.

Analyze interactions with DB:

- **configuration details with decisions**

When previously the history for tags was activated, the data started to be stored in the database. So no further configuration is required.

- **overall observations and screenshots if required**

Interactions between ignition and the database can be observed through http://localhost:8088. In status tab, under "Database Connections" and "Store & Forward" the total number of queries, active connections and active queries can be observed.

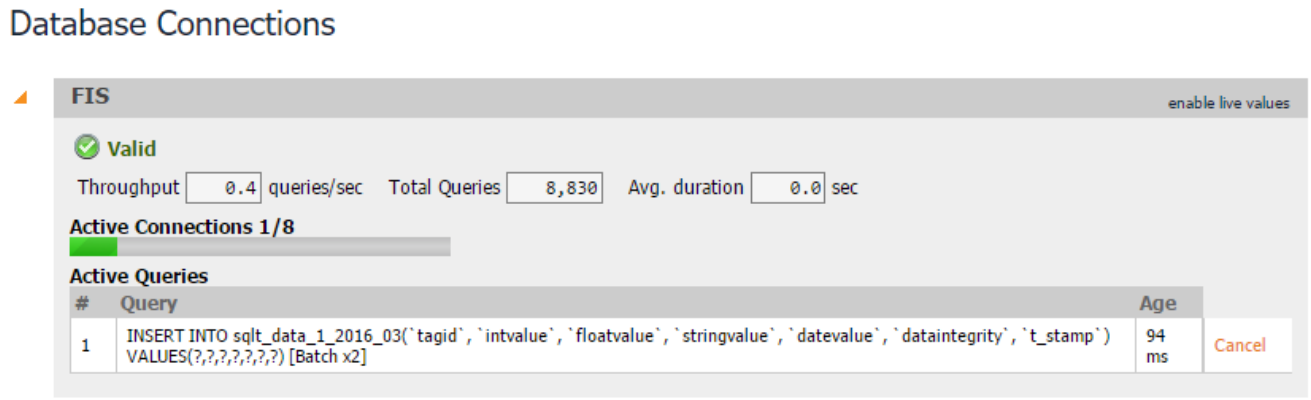

Figure 23: Database Connections

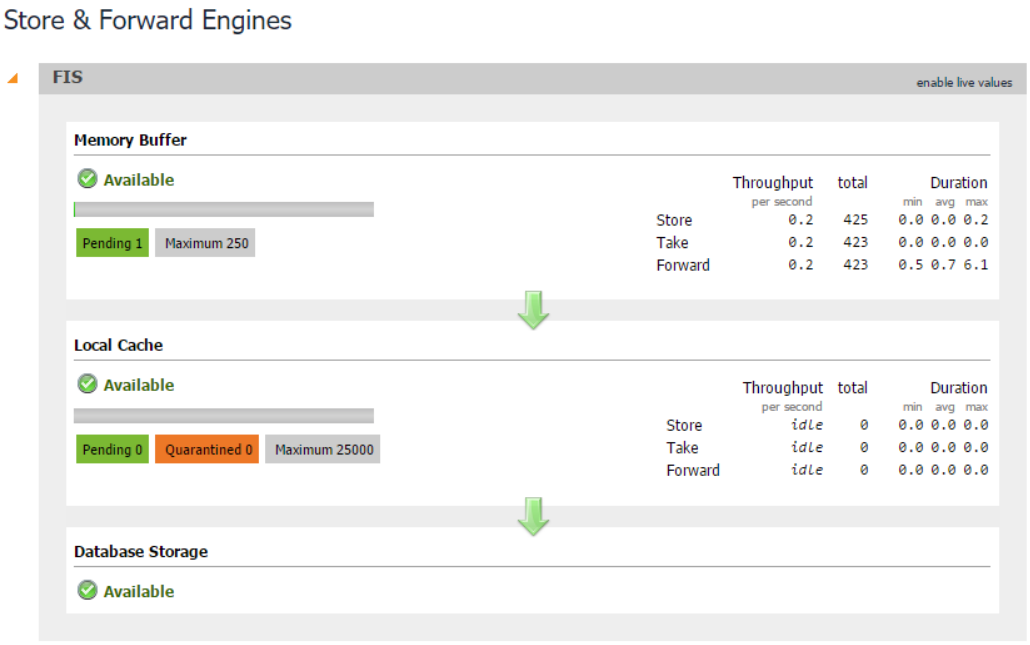

Figure 24: Store & Forward