# Datonis Gateway 3.5.0

# PURPOSE

This document describes the procedure to employ the Datonis IOT Gateway in your environment.

# SYSTEM REQUIREMENTS

## Operating System

The Datonis Gateway is certified to run on:

- Windows 7 or above
- Linux 3.x kernel or above

## Supported Protocols

- http (80)
- https (443)
- mqtt (1883)
- mqtts (8883)

## Internet connectivity

Internet connectivity is required so that the gateway can do outbound connections to either of the ports (80 - http, 443 - https, 1883 - mqtt, 8883 - mqtts)
For http/https, you can also configure a http proxy if there is no direct internet connectivity available.

## JAVA

Gateway needs **JAVA 1.7** or higher to work. Please download from Oracle site.

# SUPPORTED ENVIRONMENTS

The Datonis Gateway supports following environments (via configurable adapters) to collect data and push it to the Altizon's Datonis Platform

### OPC DA

This adapter can be used to point to a OPC DA server either running alongside the gateway on the same machine or on a different machine on the LAN.
Detailed instructions are provided in the OPC DA CONFIGURATION section.
Certified OPC DA servers include: Matrikon, Kepware, Proface, Acromag

### OPC UA

This adapter can be used to point to a OPC UA server either running alongside the gateway on the same machine or on a different machine on the LAN.
Detailed instructions are provided in the OPC UA CONFIGURATION section
Certified OPC UA servers include: Kepware OPC UA server, Beckhoff Twincat

### MTConnect Agent

Data can be queried from MTConnect agents that expose a XML based interface. Detailed instructions are provided in MTCONNECT AGENT CONFIGURATION section

### Modbus TCP

This adapter can be used to read data from a number of Modbus devices using TCP.
Detailed instructions are provided in the MODBUS TCP CONFIGURATION section.

# PACKAGE STRUCTURE

datonis-gateway-3.5.0/

```
|
|--- bin
|--- conf
|--- lib
|--- commons-daemon-1.0.15-bin-windows
```

- bin directory contains all the scripts required to install/run the datonis gateway.
- conf directory contains:
    - A bunch of configuration templates for configuring the datonis gateway e.g. opc da, http poller, mtconnect etc. that would help you get started
    - log4j.properties where logging of the gateway can be controlled
- lib directory contains all dependent java libraries required for the gateway to work
- commons-daemon-1.0.15-bin-windows contains a program to install/run/uninstall the gateway as a service on Windows

# INSTALLATION

## Windows

On windows, there are a couple of files in the bin directory

- install-windows-service.bat
  This file installs a Windows service that lets you start/stop the Datonis IOT Gateway via the Windows Service Control Manager.
  In the root folder of the package, just run **bin\install-windows-service.bat**
- uninstall-windows-service.bat
  This file uninstalls the Windows service for the Datonis IOT gateway. Invoke this script similar to the one above in case you need to remove the Datonis gateway from your machine

## Linux

There is a script **start-datonis-gateway.sh** in the bin directory. Invoking this from the shell launches the agent

**NOTE: Make sure to do the configurations below before starting the Datonis Gateway**

# GATEWAY CONFIGURATION

The Datonis Gateway needs a file called **datonis-gateway.properties** to be present in the conf directory. When you unzip the package, you will find a few templates to help you get started for a particular environment. It is a **JSON** file so make sure that the JSON is parseable at all times. You can do that by pasting the contents of this file at: http://jsonlint.com/ and clicking validate.

This part of the configuration defines how the gateway connects to the platform and identifies itself. Configuration options required by the Gateway are as described below. Commonly configured ones are those that have a highlighted background. Others you may not need to explicitly specify unless otherwise necessary:

| Parameter | Description | Data type | Default value | Mandatory |
|---|---|---|---|---|
| access_key | Access key associated with a key pair that identifies the customer account on the Datonis Platform.<br>This can be downloaded from the platform UI at: http://www.datonis.io | String | None | Yes |
| secret_key | Secret key associated with the key pair.<br>Note that this key should not be compromised. It is supposed to be private. | String | None | Yes |
| bulk_transmit | Whether to transmit events in a batch manner or immediately as they get generated. Transmitting in batch manner saves bandwidth. | Boolean | true | Yes |
| heartbeat_interval | How often to send heartbeats for things to the platform. This is to make the platform know that the things are alive (even if they are not actively transmitting data).<br>Unit is milliseconds | Integer | 120000 | No |
| request_timeout | Timeout while connecting/receiving response from the platform.<br>Unit is milliseconds | Integer | 10000 | No |

| | | | | |
|---|---|---|---|---|
| thread_pool_size | Number of worker threads | Integer | 5 | No |
| bulk_transmit_interval | If bulk_transmit is true, how long to batch requests before finally sending the batch.<br>Unit is milliseconds | Integer | 60000 | No |
| bulk_max_elements | How many events to batch before finally sending the batch.<br>Unit is milliseconds | Integer | 25 | No |
| protocol | Which protocol to use to send data:<br>HTTP/HTTPS/MQTT/MQTTS | String | HTTPS | No |
| proxy_host | In case of HTTP/HTTPS, http proxy server name in case of indirect Internet connectivity | String | None | None |
| proxy_port | proxy server port | Integer | None | None |
| proxy_username | proxy server username | String | None | None |
| proxy_password | proxy server password | String | None | None |
| proxy_domain | Proxy server domain (in case of NTLM) | String | None | None |

# ADAPTER THING CONFIGURATION

This part of the configuration defines how things on the Datonis platform are mapped to the tags in your environment. All the environments (OPC DA/UA, MTconnect, Modbus etc.) mentioned above are supported via some sort of an adapter. All types of adapters have a thing configuration where you specify the mapping between - the Datonis Things - and - the tags/metric values collected and available in your environment.
Each adapter in the **datonis-gateway.properties** file has a **thing_configuration** section which is an array. Each element describes a Thing on the datonis platform and its attributes.
Each attribute maps to a tag in your environment

## Thing Configuration

| Parameter | Description | Data type | Default value | Mandatory |
|---|---|---|---|---|
| thing_key | Key of the thing obtained from the Datonis Platform UI: http://www.datonis.io | String | None | Yes |
| thing_name | Name of the thing associated with above thing key | String | None | Yes |
| refresh_interval | How frequently you want the tags configured this thing's attributes to be probed/queried.<br>Value to be specified in milliseconds | Integer | None | Yes |
| thing_attributes | List of attributes for this thing having their own configuration.<br>Please refer table below for thing attribute configuration | List | None | Yes |

## Thing Attribute configuration

| Parameter | Description | Data type | Default value | Mandatory |
|---|---|---|---|---|
| name | Name that identifies the attribute for a thing | String | None | Yes |
| tag_id | Id of the tag in a target system like an OPC server. You can even specify multiple comma separated tags<br><br>In case of MTconnect, it is the Xpath of the agent that provides value for a certain Macro<br><br>In case of Modbus, you need to specify memory locations and bit mask addresses | String | None | Yes |
| diff | If the tag represents an ever increasing number, and you want to send the diff since last reading, then set this to true.<br><br>So in case there is a "counter" tag that increases like this:<br>31234, 31235, 31238, 31240<br>then, the values transmitted for this attribute would be 0, 1, 3, 2,.. | Boolean | false | No |

| | | | | |
|---|---|---|---|---|
| expression | In case of an expression, you can specify any simple mathematical formula.<br><br>There are two context arrays available for the expression:<br>latestValue and previousValue.<br><br>latestValue[i] corresponds to the latest value of tag number ‘i’ in the comma separated list of tags (see tag_id field).<br><br>previousValue[i] corresponds to the previous value of tag number ‘i’ in the comma separated list of tags (see tag_id field) | | | |

# OPC DA CONFIGURATION

There are two parts to this configuration.

- Configuring the OPC server DCOM settings so that the Datonis Gateway can query data from it
- Configuring the OPC server details in the **datonis-gateway.properties** file

## DCOM configuration on the OPC Server

The Datonis Gateway acts as a OPC client while fetching data from an OPC server. The default DCOM settings for a Windows host do not permit this communication. Thus it becomes necessary to make appropriate configuration changes so that this communication goes through.

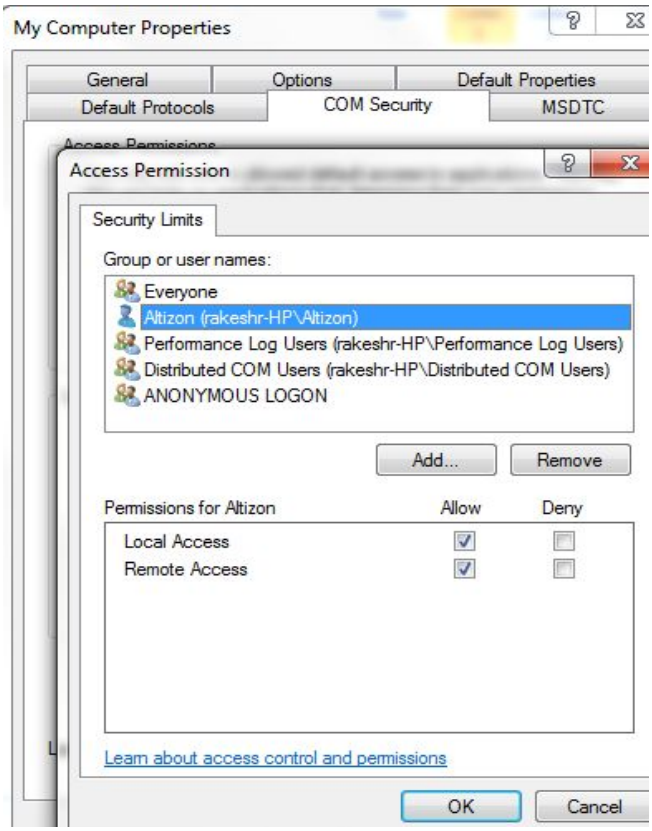
1. Create a local Windows user called “altizon” on the OPC server machine.
    a. Right click **Computer** and click **Manage**
    b. Under **Computer Management** -> **Users and Groups** -> **Users**, right click and select **New User…**
    c. Set an appropriate password
    d. Uncheck all options and then only check the **Password never expires** option
    e. Also, add it to the **Administrators** group

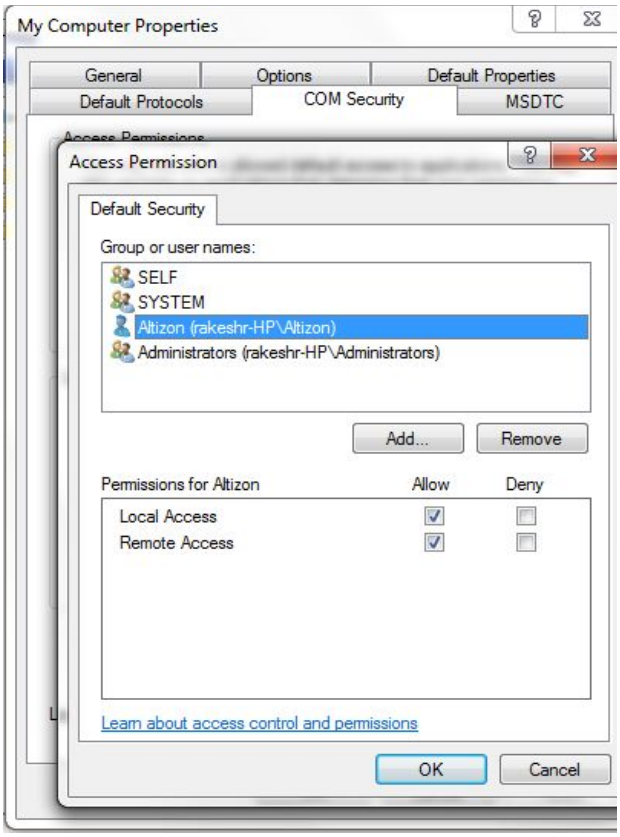
2. In the start menu, in search/run, enter **dcomcnfg**

3. Under **Console Root** -> **Component Services** -> **Computers** -> **My Computer**, right click and select properties

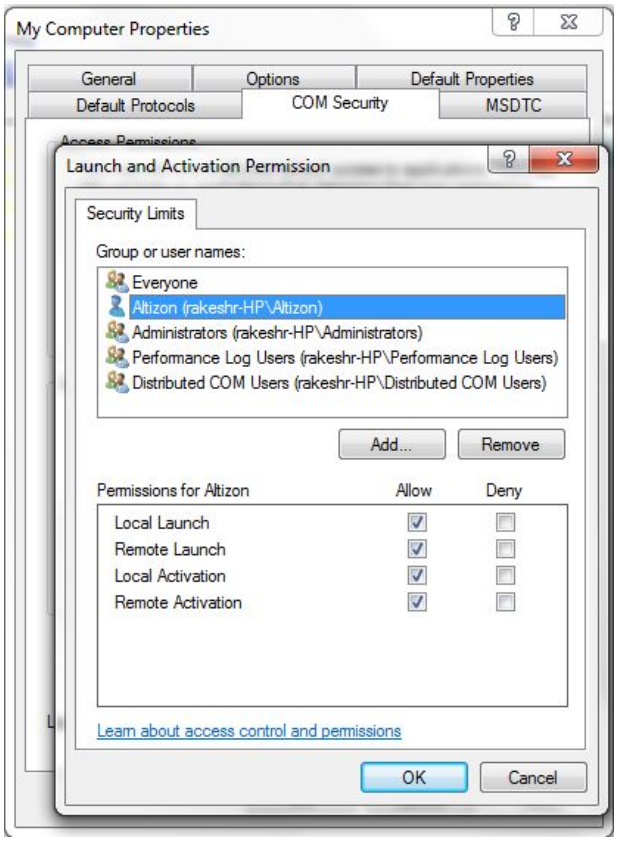
4. In the **COM Security** tab, under **Access Permissions**

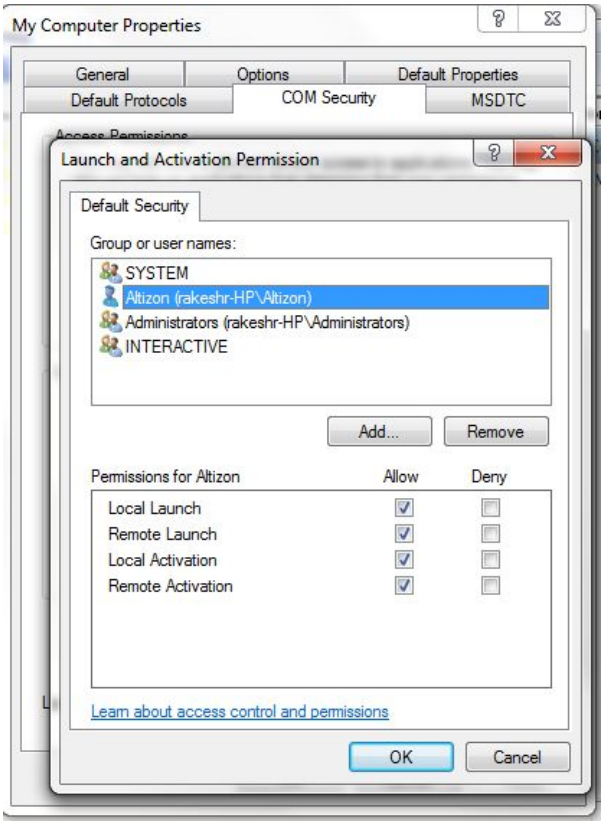
a. click **Edit Limits…,** In the pop up window that opens, click **Add…** and select the user we added in step 1.
b. Select both checkboxes **Local Access** and **Remote Access** under **Allow**
c. click **Edit Defaults…** In the pop up window that opens, click **Add…** and select the user we added in step 1.
d. Select both checkboxes **Local Access** and **Remote Access** under **Allow**



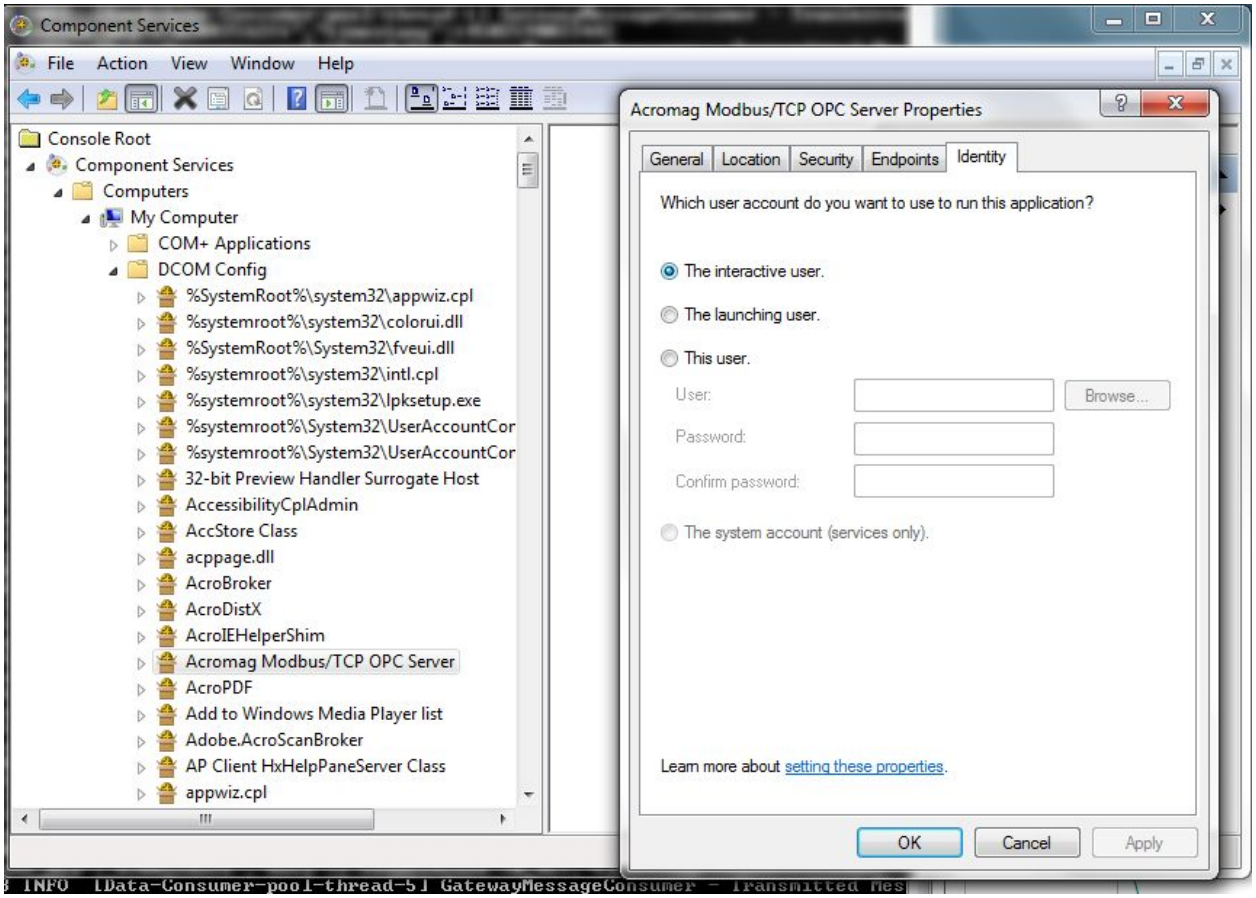



5. In the **COM Security** tab, under **Launch and Activation Permissions**
    a. click **Edit Limits…** In the pop up window that opens, click **Add…** and select the user we added in step 1.
    b. Select all checkboxes under **Allow**
    c. click **Edit Defaults…** In the pop up window that opens, click **Add…** and select the user we added in step 1.
    d. Select all checkboxes under **Allow**




6. Under **Console Root** -> **Component Services** -> **Computers** -> **My Computer** -> **DCOM Config**
   look for the entry corresponding to your OPC server. Right click and select **Properties**
7. In the Identity tab, select the user to **The Interactive user**.




8. **Reboot** the machine. Reboot is required for these changes to take effect

## Datonis Gateway configuration for OPC DA

As mentioned in the ADAPTER THING CONFIGURATION section, each adapter in the **datonis-gateway.properties** file has its own section. For OPC-DA, we need to configure the **opc_da** adapter. Please refer the datonis-gateway.properties.opc-da-template for a sample. Under this adapter, there is a **opc_server** section where we need to configure the OPC server details

| **Parameter** | **Description** |
|---|---|
| opc_server_ip | IP Address of the OPC server |
| opc_server_cls_id | DCOM class id for the OPC server. Details of how to get it are mentioned below |
| opc_server_username | Specify the user we added in the DCOM configuration section above |
| opc_server_password | Specify the corresponding password |
| ops_server_domain | In case the user is a domain user, specify the Windows domain with this attribute |

**How to find the OPC server's DCOM class id?**

1. In the start menu, type **regedit**.
2. expand **HKEY_CLASSES_ROOT** and look for the name that matches your OPC server.
3. Once found, expand it and you will find a key named **CLSID**
4. Copy the corresponding value. This is your class id. Note that you need to ignore the curly braces around the ID.

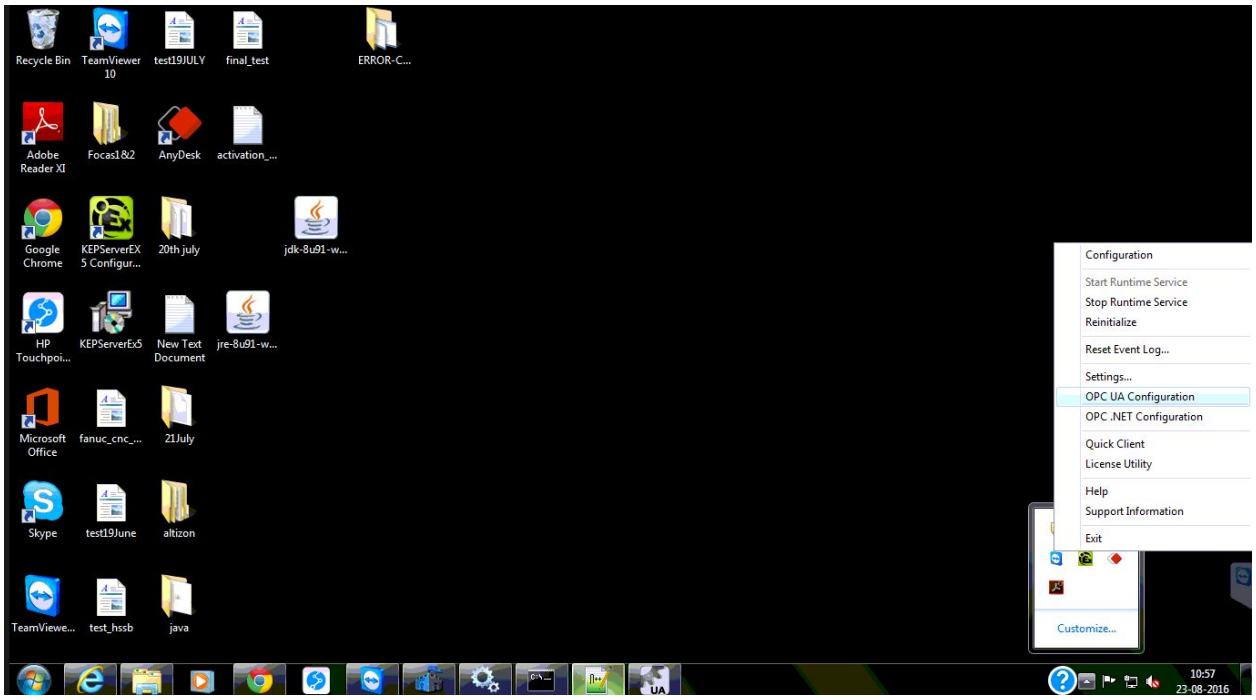
# OPC UA CONFIGURATION

## Datonis Gateway configuration for OPC UA

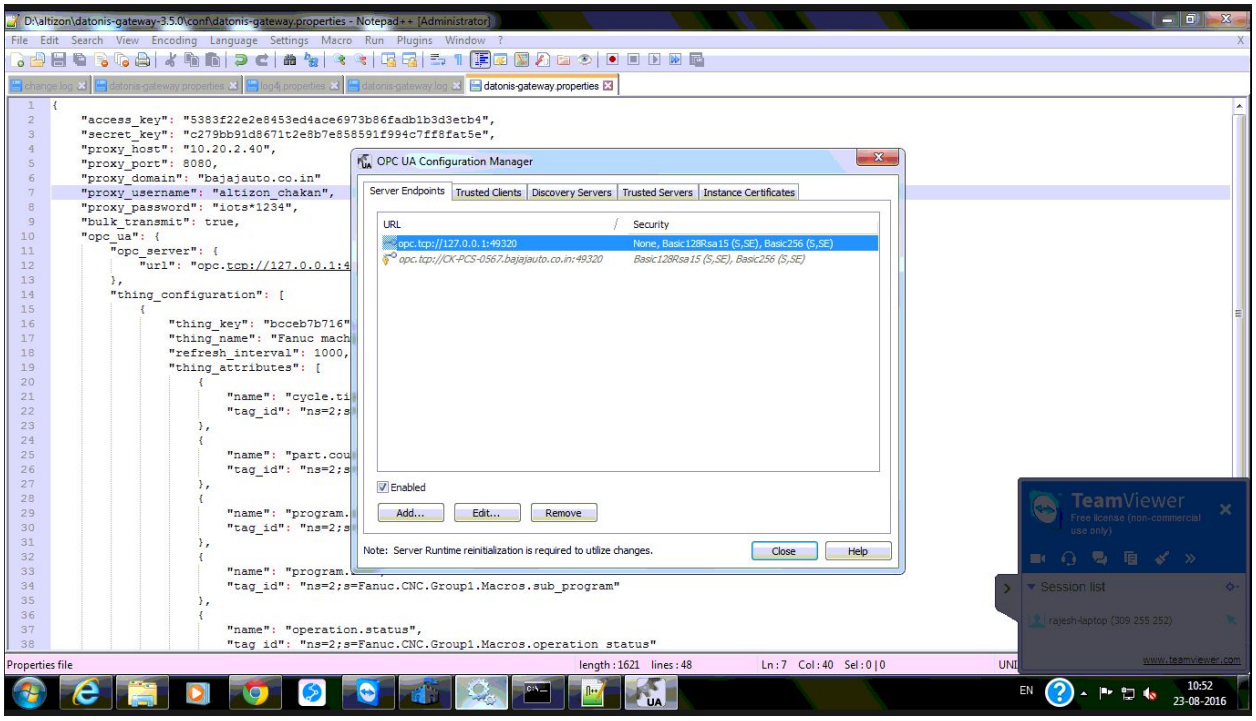
For OPC UA we need to configure the **opc_ua** adapter. Please refer the datonis-gateway.properties.opc-ua-template for a sample.
Under this adapter, there is a **opc_server** section where we need to configure the OPC server details

| **Parameter** | **Description** |
|---|---|
| url | The TCP URL at which the OPC UA server is running.<br>For example:<br><br>opc.tcp://127.0.0.1:49320<br>This is a standard URL for Kepware UA<br><br>opc.tcp://127.0.0.1:48050<br>This is a standard URL for Bechoff Twincat UA<br><br>Please refer the Launching OPC UA configuration and Noting OPC UA Url sections below |

## Launching OPC UA configuration screen



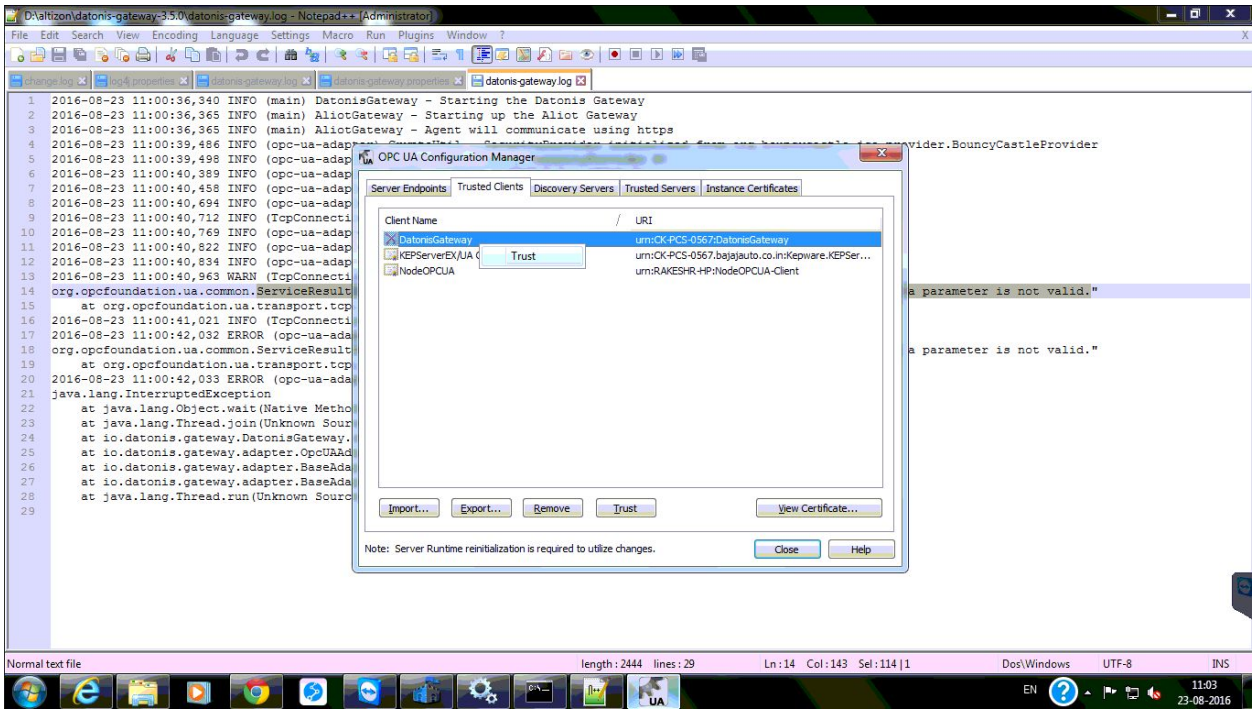

## Noting OPC UA Url



## Adding Datonis Gateway certificate to the trusted list

On restarting the Datonis IOT Gateway, it will error out with a certificate error:
**ServiceResultException: Bad_CertificateInvalid (0x80120000) "The certificate provided as a parameter is not valid.**

In the OPC UA configuration window, select the Trusted Clients tab, you will find a new entry for Datonis Gateway. Right click it and select “Trust”.




Restart the Datonis IOT Gateway from the Windows services console again.

## Thing configuration for OPC UA

The thing configuration is generic and is applicable in the same way to all adapters. Please see ADAPTER THING CONFIGURATION.

The format for tag is as follows:

ns=<namespace id>;s=<string identifier of the tag>
OR
ns=<namespace id>;i=<integer identifier of the tag>

For example:
ns=2;s=Channel1.Machine1.Tag1

When you configure the OPC UA adapter, on (re)starting the gateway, you will see the entire object hierarchy printed in the log. You can locale the tag you want in that information.

**Note that you will need to restart the Datonis IOT Gateway service from the services view every time you add a new metric/tag in the datonis-gateway.properties file**

# MTCONNECT CONFIGURATION

TBD

# MODBUS TCP CONFIGURATION

The Modbus TCP Adapter does the work of the Master in the master-slave model of the protocol while the devices act as Slaves. The Master is responsible for reading data from certain addresses in the devices. The details regarding what data is to be read from which addresses, need to be configured in the datonis-gateway.properties file. The adapter will run with configured options and send data to the platform.

## Datonis Gateway configuration for Modbus TCP

As mentioned in the ADAPTER THING CONFIGURATION section, each adapter in the **datonis-gateway.properties** file has its own section. For Modbus, we need to configure the **modbus** adapter. Please refer the datonis-gateway.properties.modbus-tcp-template for a sample.

The **modbus_devices** section of the properties file, contains an array of the modbus devices to be connected to the adapter. Each device configuration has fields as follows:

| **Parameter** | **Description** |
|---|---|
| deviceId | It is used for numbering the different devices connected to the adapter. For example, in case of two devices the “deviceId” will be 0 and 1 respectively. |
| ip_address | It is the IP address of the modbus device. |
| port | It indicates the port number which the device is communicating on. |

The **thing_configuration** section in the properties file has a **tag_id** property which contains a number of key value pairs that need specific configuring. Details of the values needed to be configured are given below.

| Parameter | Description |
|---|---|
| deviceId | It is mapped with the “deviceId” from the above **modbus_devices** section and helps to identify which device the given attribute is to be read from. |
| type | It indicates the type of data to be read from the device. Its possible values are Register, InputRegister, Coil and InputDiscrete. |
| readCount | It indicates the number of data items to be read for the given attribute. For example: to read two 16-bit registers for a 32-bit value, the readCount will be set to 2. |
| refAddr | This indicates the starting address in the device after which the values are to be read for the given attribute. For example, a Register storing a value at address 50001 will have the refAddr set to 50000 and an InputRegister with a value stored at 50002 will have refAddr set to 50001. |
| endian | This value is applicable **only** for Register and InputRegister type. In case of 32/64-bit numbers, it is used to define whether values are to be read in little endian or big endian format. 0 = little endian, 1 = big endian. |
| bitIndex | This value is applicable **only** for Coil and InputDiscrete type. It indicates the index number of the bit to be read from the fetched bitmask. Values can range from 0-7. |
| valType | This flag can be set to “float” or “integer”. Default value chosen will be “integer” such that the values read from the registers will be treated as Integers. We currently support 32-bit floating point numbers. To enable 32-bit floating point values, set valType=float, readCount=2 and endian to 1 or 0 (usually floating point values are stored in the big endian format i.e. 1). |

A 32-bit floating point value can be configured as follows:

For example: a metric named ‘voltage’

```
"thing_attributes": [
        {
          "name": "voltage",
          "type": "number",
          "tag_id": "deviceId=0;type=Register;readCount=2;refAddr=0100;endian=1;valType=float"
         }
 ]
```