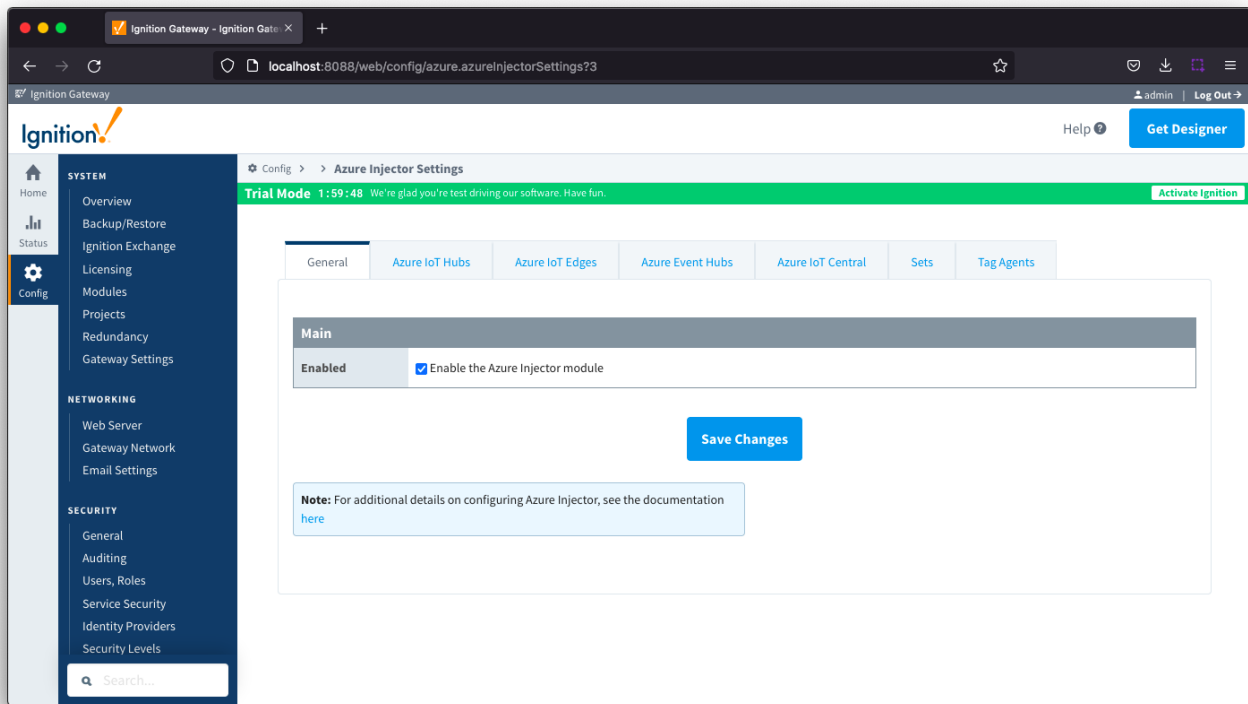
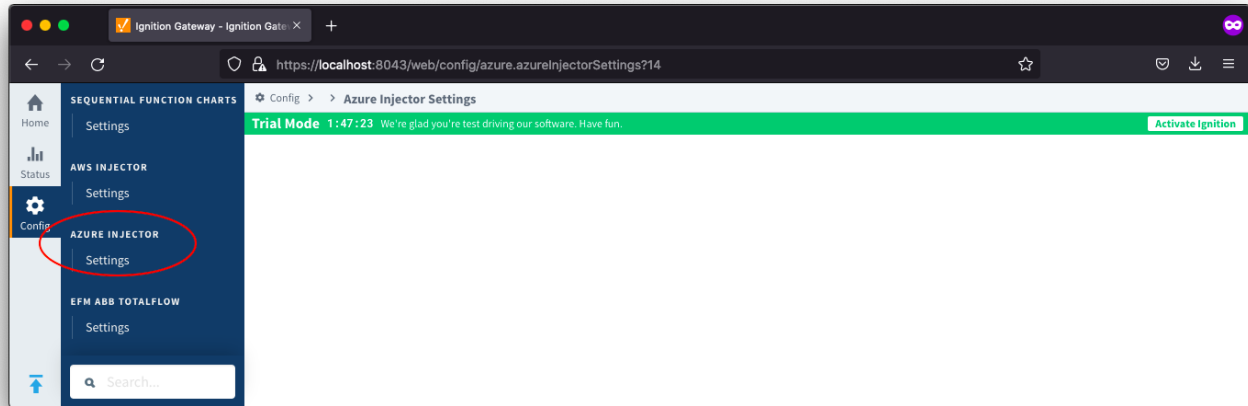


AZI: Configuration

The Azure Injector module provides the ability to push Tag data to an Azure IoT Hub, Azure IoT Edges, Azure Event Hubs and Azure IoT Central endpoints.

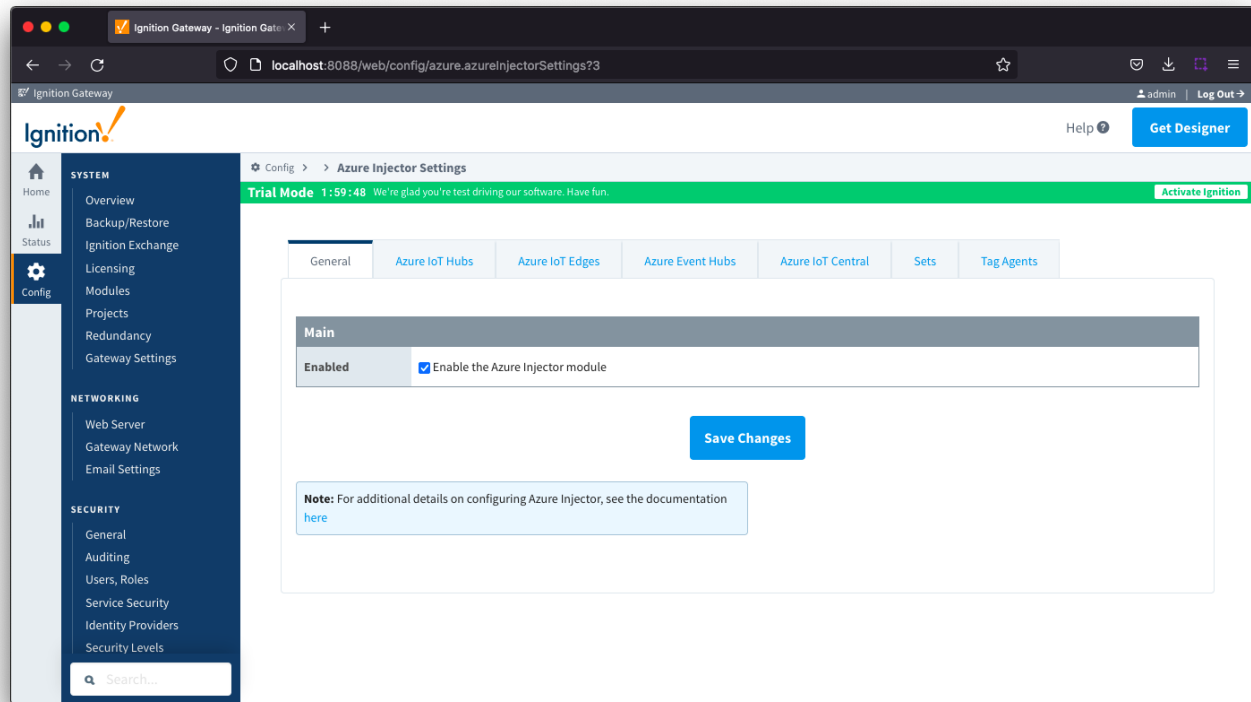
The configuration for this module are located under the Configure tab of the Ignition Gateway web UI in the left hand navigation pane under 'Azure Injector Settings'.



The configuration options for each of the seven tabs - [General](#), [Azure IoT Hubs](#), [Azure IoT Edges](#), [Azure Event Hubs](#), [Azure IoT Central](#), [Sets](#) and [Tag Agents](#) - are detailed below.

General

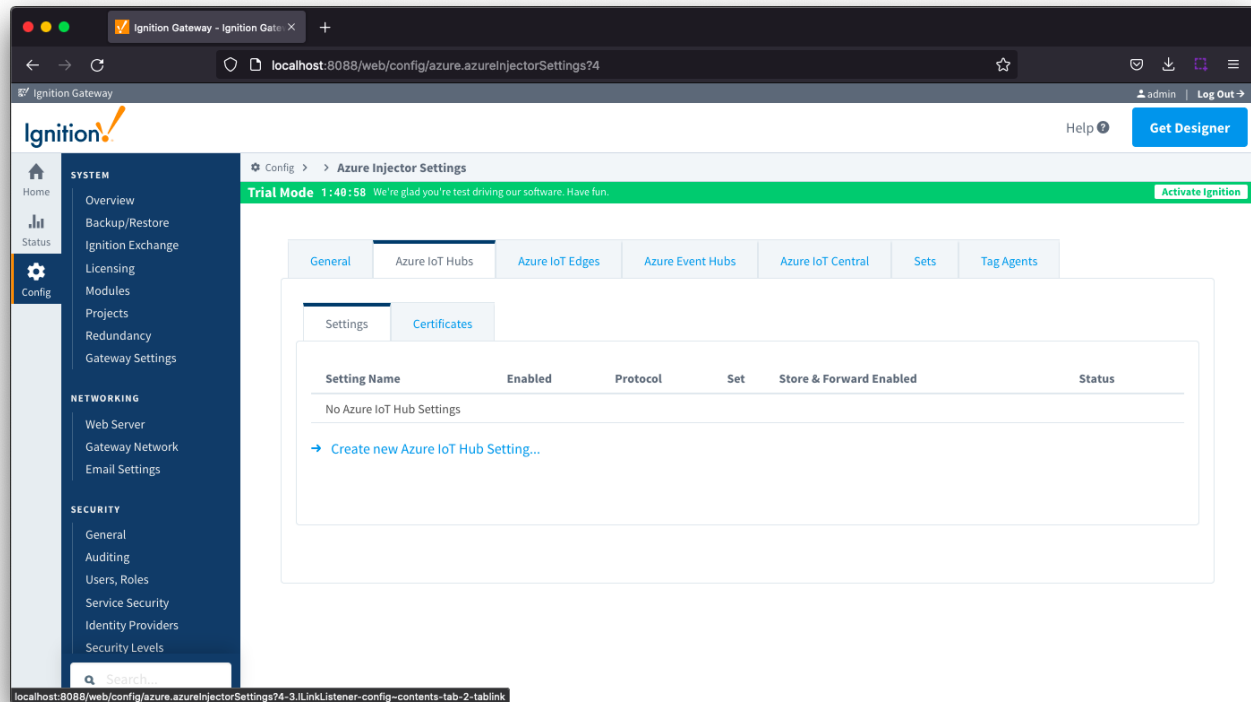
The configuration section available is Main.



- **Enabled**
 - Sets whether the module is enabled or disabled. If disabled, the Tag Agents will not run and now data will be pushed to any configured endpoints.

Azure IoT Hubs

The Azure IoT Hubs tab has two parts - [Settings](#) and [Certificates](#)

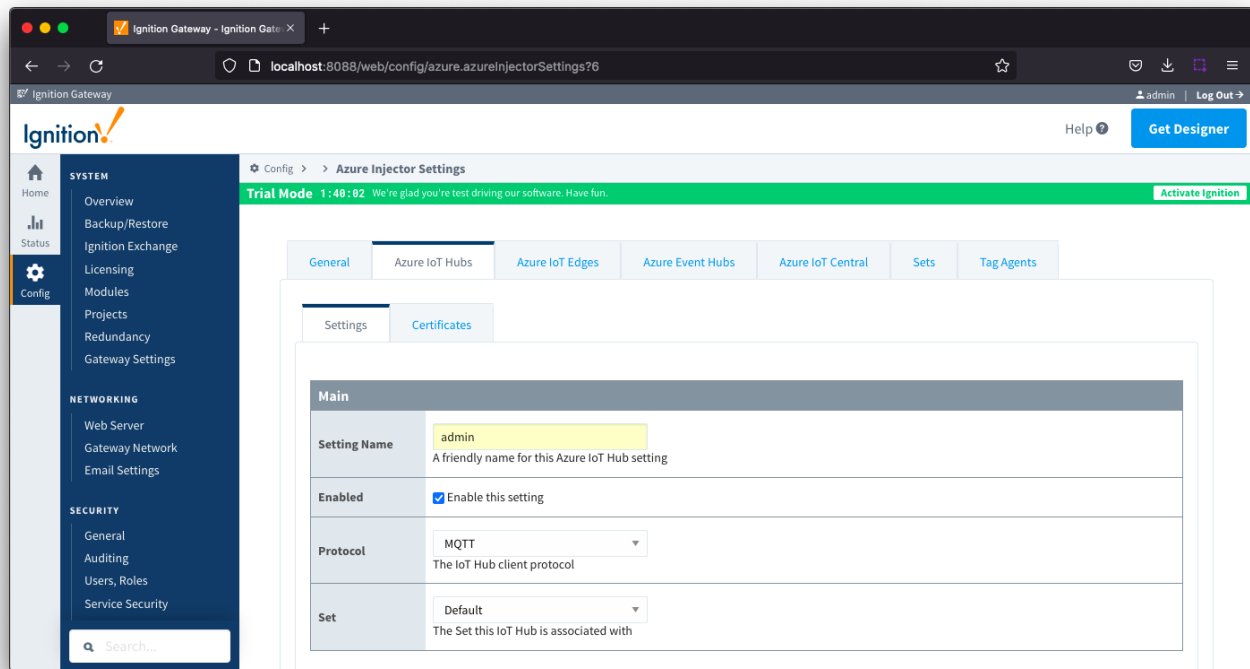


Azure IoT Hubs - Settings

This tab provides a list of the Azure IoT Hub endpoints that the module should connect to to push tag data. One or more Azure IoT Hub endpoints can be configured on this tab.

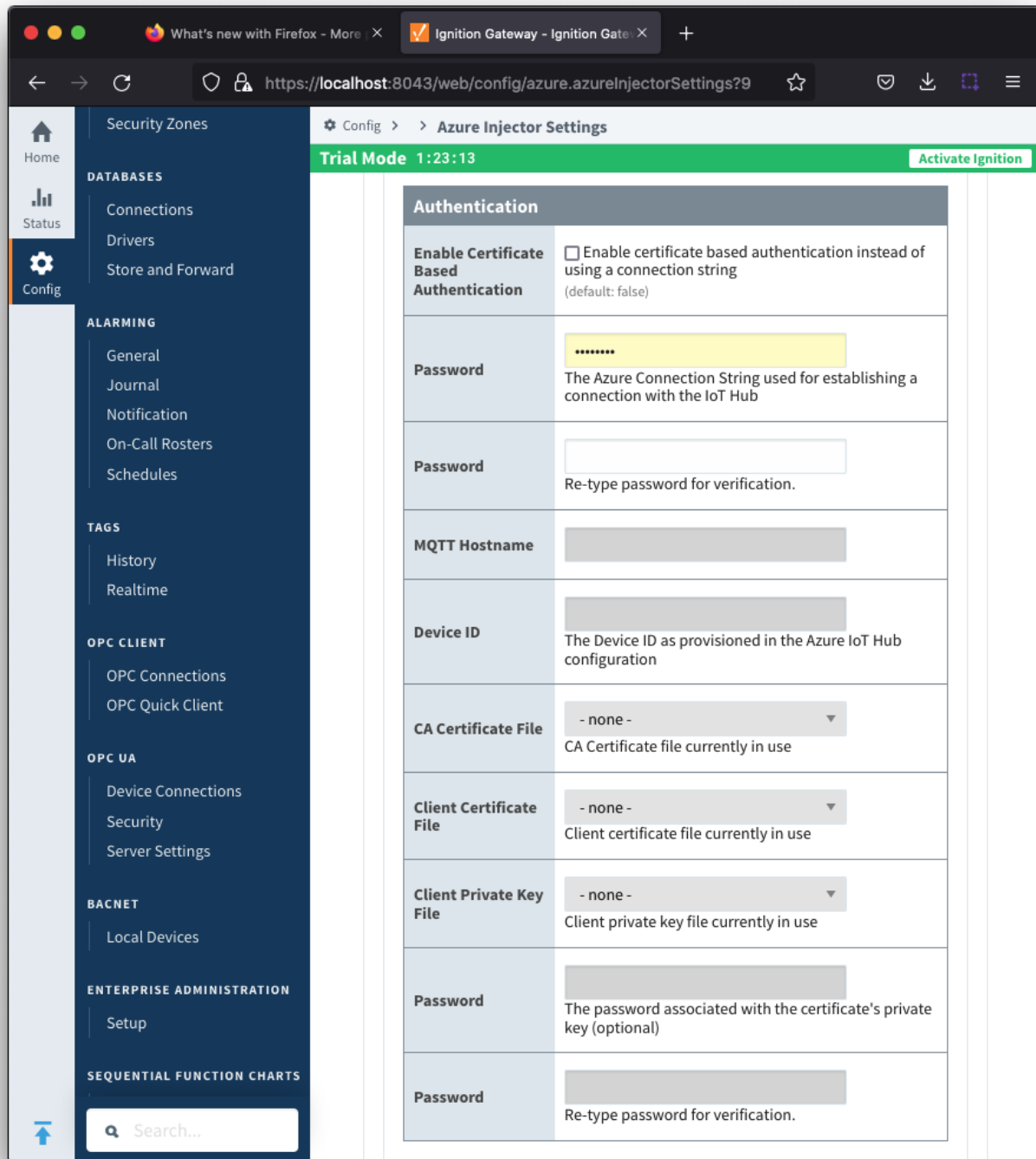
Clicking on the 'Create new Azure IoT Hub ..' link will bring up the following form to add a new Azure IoT Hub. The configuration sections available are [Main](#), [Authentication](#), [Store & Forward](#) and [Advanced](#)

Azure IoT Hub Settings - Main



- **Setting Name**
 - This is a friendly name of the Azure IoT Hub used to easily identify it. This must also be unique.
- **Enabled**
 - Whether or not this connection is enabled.
- **Protocol**
 - The protocol to use when connecting to the Azure IoT Hub.
 - Currently MQTT only is supported.
- **Set**
 - The Set to associate this Azure IoT Hub connection with.

Azure IoT Hub Settings - Authentication



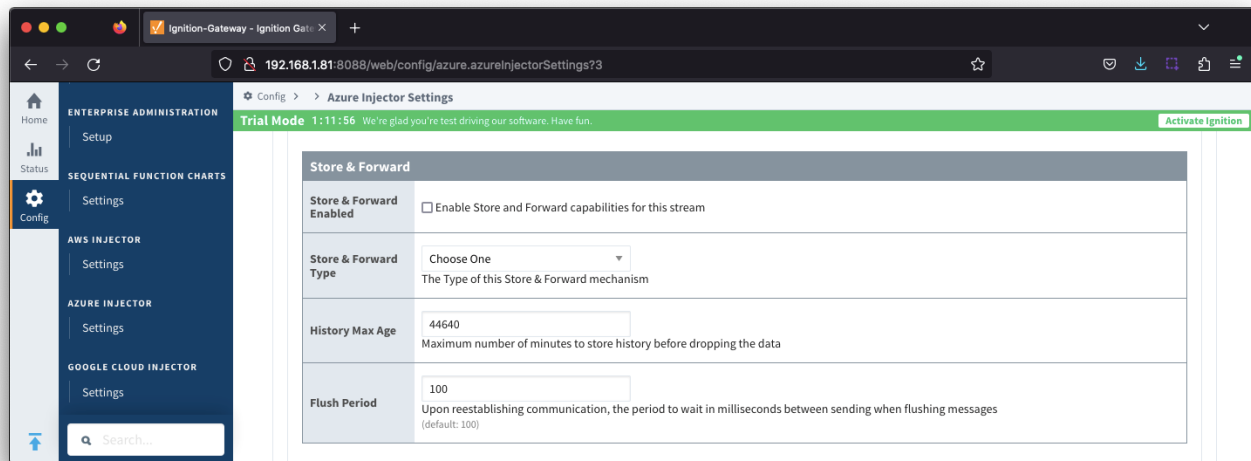
- **Enable Certificate Based Authentication**
 - Whether or not to use certificate based authentication.
 - This determines the authentication fields available for use.
- **Password**
 - Available if not using certificate based authentication
 - This is the Azure IoT Hub device connection string used to connect in the following format:
HostName=<Host Name>;DeviceId=<Device Name>;SharedAccessKey=<Device Key>
- **MQTT Hostname**
 - Available if using certificate based authentication
 - This is the DNS endpoint name of your IoT Hub
- **Device ID**
 - Available if using certificate based authentication
 - The Device ID to connect to as provisioned in the IoT Hub
- **CA Certificate File**

- Available if using certificate based authentication
- The CA certificate that signed the SSL certificate being used in the IoT Hub server. See [this document](#) for more information.
- The drop down is populated from a list of files that have been uploaded to the IoT Hub/Certificates tab.
- **Client Certificate File**
 - Available if using certificate based authentication
 - The client certificate file as provisioned for the Device ID specified above. See this [Connecting to Azure IoT Hub with Certificate Based Authentication](#) for details on creating the client certificate
 - The drop down is populated from a list of files that have been uploaded to the IoT Hub/Certificates tab.
- **Client Private Key File**
 - Available if using certificate based authentication
 - The client private key file that was used in generating the certificate for the Device ID specified above. See this [Connecting to Azure IoT Hub with Certificate Based Authentication](#) for details on creating the client private key
 - The drop down is populated from a list of files that have been uploaded to the IoT Hub/Certificates tab.
- **Private key password**
 - Available if using certification based authentication
 - The password used for the private key if one was specified for the Client Private Key File

Azure IoT Hub Settings - Store & Forward

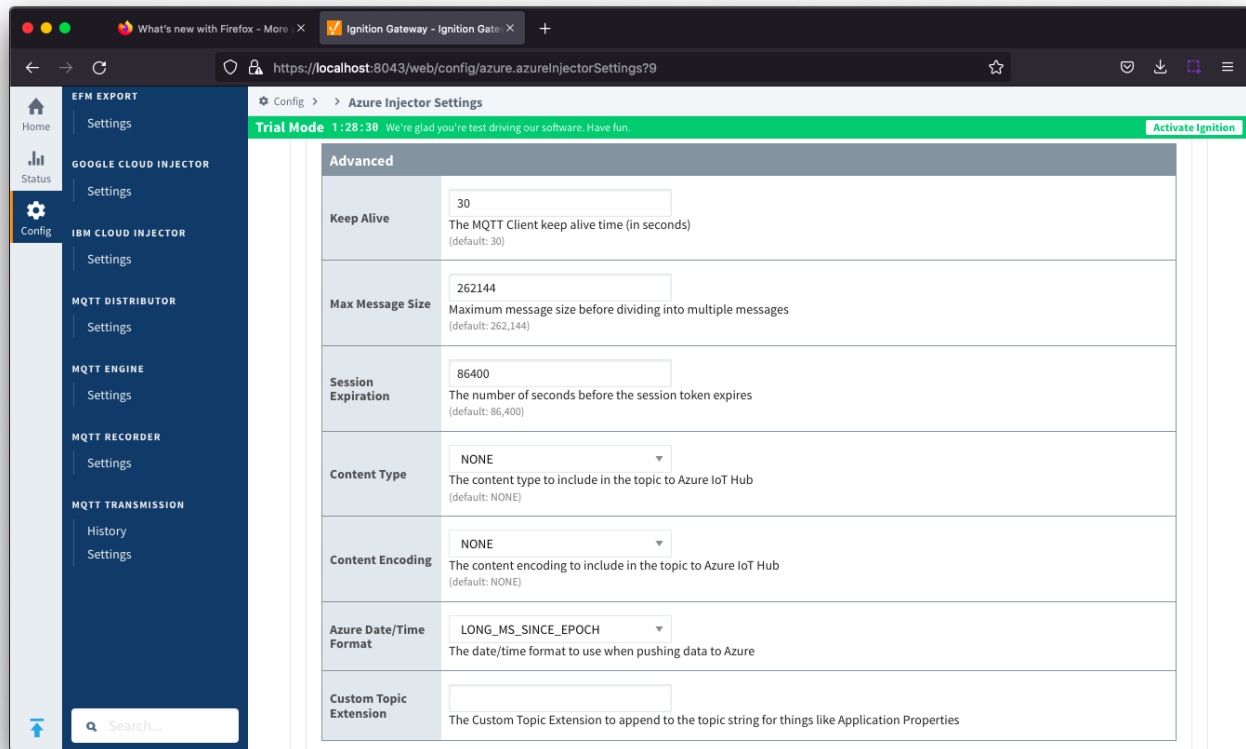


From release 4.0.19, major improvements have been made to the disk-backed History Store. As a result, Message Capacity has been deprecated and History Max Age added



- **Store & Forward Enabled**
 - Whether to enable Store & Forward capabilities for this endpoint
- **Store & Forward Type**
 - The type of the Store & Forward mechanism options: In_Memory and Disk_Backed (available in release 4.0.17 and higher)
 - Data stored with an In_Memory Store & Forward will not be persisted across a module configuration change, module disable/enable, module restart or power loss.
 - Data stored with a Disk_Backed Store & Forward will persist across a module configuration change, module disable/enable, module restart or power loss
- **Message Capacity - deprecated in 4.0.19**
 - The maximum number of messages to store before dropping the oldest historical messages
- **History Max Age**
 - The maximum number of minutes to store history before dropping the data
- **Flush Period**
 - The period of time to wait (in milliseconds) between sending when flushing messages

Azure IoT Hub Settings - Advanced



- **Keep Alive**
 - The MQTT keep alive timeout in seconds
- **Max Message Size**
 - The maximum message size in bytes that any message can be when pushing to IoT Hub. Generally, this should match the max message size allowed by IoT Hub.
- **Session Expiration**
 - How long in seconds to specify for session token timeouts when not using certificate based authentication
- **Content Type**
 - The content type to include in the topic to Azure IoT Hub
 - NONE (default) - No content type header will be included with the message
 - APPLICATION_JSON - The application/json header will be included with the message and make the body of the message available for routing if content encoding is also not 'NONE'
 - See [Using IoT Hub Message Based Routing](#) tutorial for more details
- **Content Encoding**
 - The content encoding to include in the topic to Azure IoT Hub
 - NONE (default) - No content encoding header will be included with the message
 - UTF_8 - The 'utf-8' header will be included with the message and make the body of the message available for routing if the content type is also set to APPLICATION_JSON
 - UTF_16 - The 'utf-16' header will be included with the message and make the body of the message available for routing if the content type is also set to APPLICATION_JSON
 - UTF_32 - The 'utf-32' header will be included with the message and make the body of the message available for routing if the content type is also set to APPLICATION_JSON
 - See [Using IoT Hub Message Based Routing](#) tutorial for more details
- **Azure Date/Time Format**
 - The date/time format to use when pushing messages to IoT Hub
 - LONG_MS_SINCE_EPOCH (default) - The timestamp values will all be as numbers in milliseconds since epoch (Jan 1, 1970) in UTC
 - STRING_AZURE_COMPAT - The timestamp will be pushed as described [here](#). This is useful when wanting to use 'edge' timestamps in Azure Time Series insights.
 - See [Pushing Data to Azure Time Series Insights](#) tutorial for more details

Azure IoT Hubs - Certificates

This tab provides a list of the certificate or private keys if loaded and available for certificate based authentication.

This will include the CA Certificate that signed the SSL cert being used on the IoT Hub server along with any device(s) certificate and private key files.

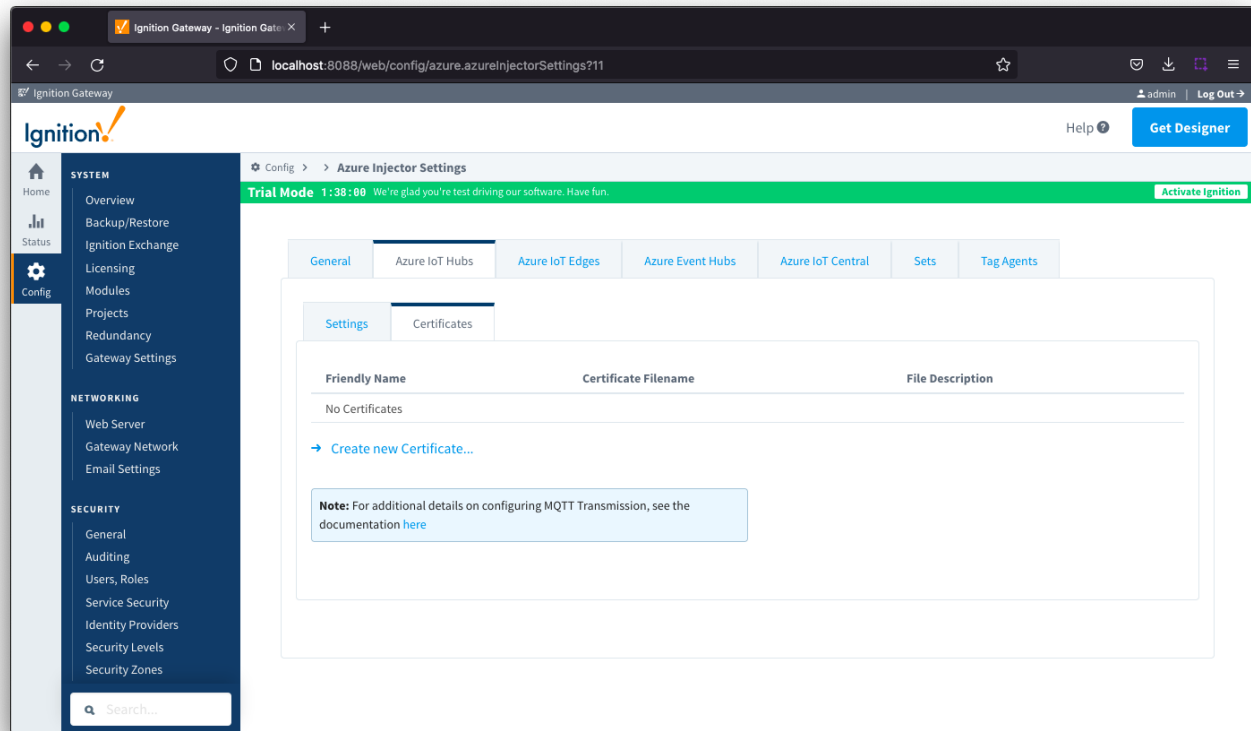


All certificate or private keys must be in PEM format.

For modules pre 4.0.9, only RSA PKCS1 format private keys are supported.

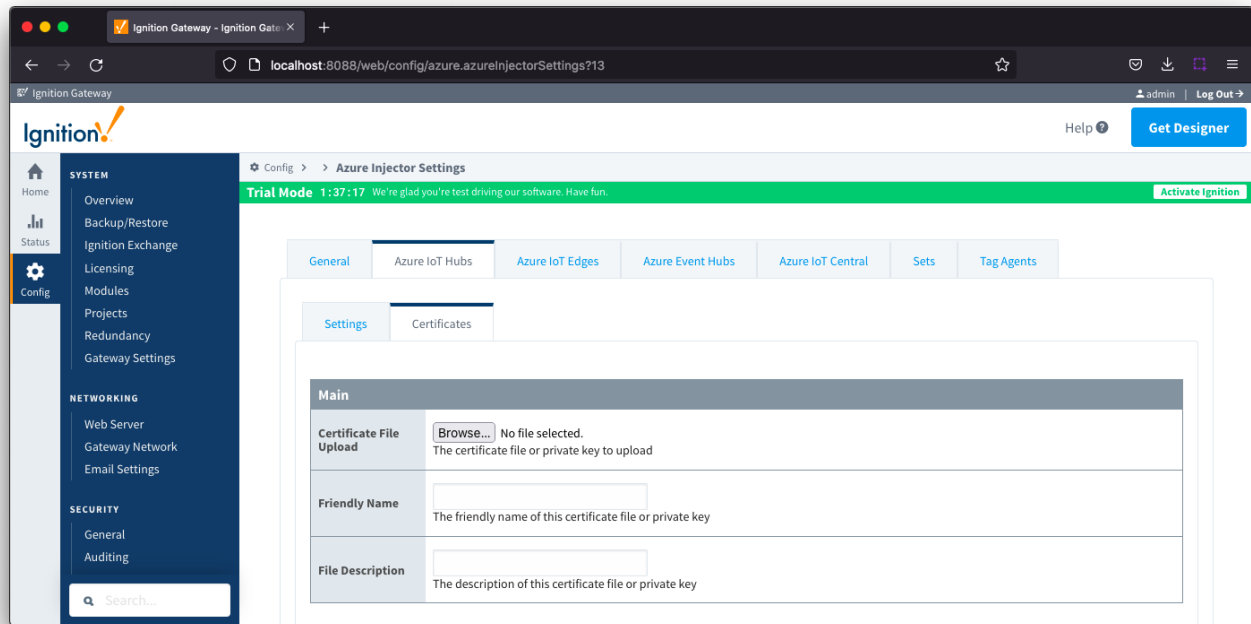
For modules 4.0.9 to 4.0.16, RSA PKCS8 format private keys are also supported.

For modules 4.0.17 or higher, password encrypted PKCS8 private keys are also supported.



Clicking on the 'Create new Certificate ..' link will bring up the following form to add a new Certificate. The Certificates tab contains a single [Main](#) section.

Azure IoT Hub Certificates - Main



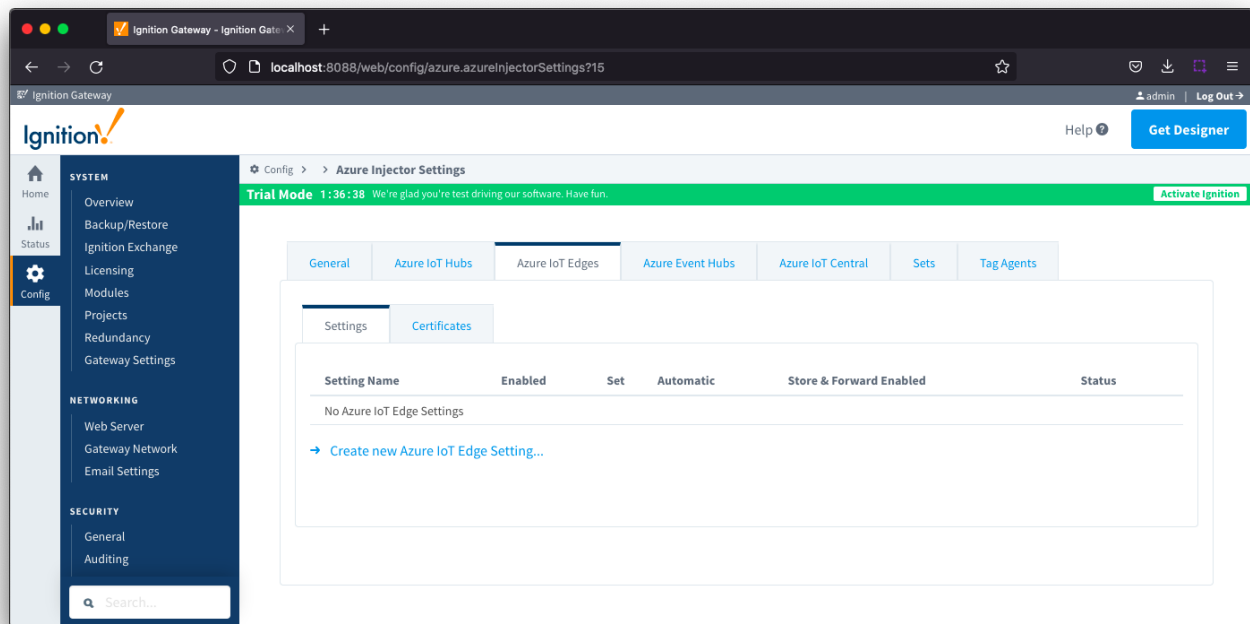
- **Certificate File Upload**
 - Browse to the certificate file or private key to upload.
- **Friendly Name**
 - The friendly name of the certificate file or private key.
- **File Description**
 - The description of the certificate file or private key.

Azure IoT Edges

The Azure IoT Edges tab has two parts - [Settings](#) and [Certificates](#)

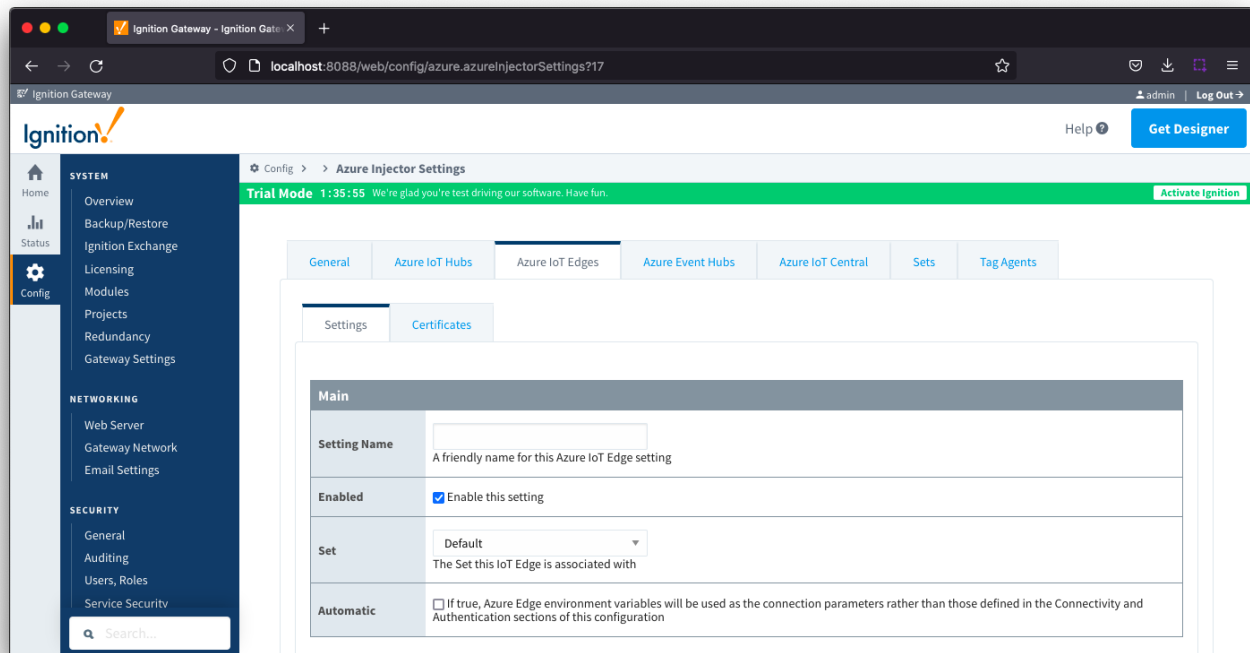
Azure IoT Edge - Settings

This tab provides a list of Azure IoT Edge endpoints that the module should connect to to push tag data. One or more Azure IoT Edge endpoints can be configured on this tab.



Clicking on the 'Create new Azure IoT Edge...' link will bring up the following form to add a new Azure IoT Edge. The configuration sections available are [Main](#), [Connectivity](#), [Authentication](#), [Store & Forward](#) and [Advanced](#).

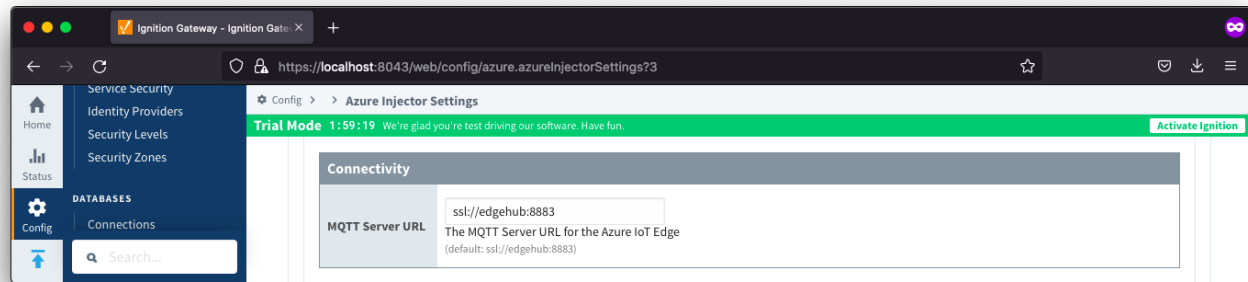
Azure IoT Edges Settings - Main



- **Setting Name**
 - This is the friendly name of the Azure IoT Edge used to easily identify it. This must be unique.
- **Enabled**
 - Whether or not this connection is enabled.
- **Set**

- The Set to associate this Azure IoT Edge connection with
- **Automatic**
 - Whether the Azure Edge environment variables will be used as the connection parameters rather than those defined in the connectivity and Authentication sections of the configuration.

Azure IoT Edges Settings - Connectivity



- **MQTT Server URL**
 - The MQTT Server URL for the Azure IoT Edge. Default: ssl://edgehub:8883

Azure IoT Edges Settings - Authentication

What's new with Firefox - More X

Ignition Gateway - Ignition Gate X

+

← → ↻

🔒 <https://localhost:8043/web/config/azure.azureinjectorSettings?2>

☆ 📄 🔍 ☰

🏠
Home

📊
Status

⚙️
Config

ALARMING

General

Journal

Notification

On-Call Rosters

Schedules

TAGS

History

Realtime

OPC CLIENT

OPC Connections

OPC Quick Client

OPC UA

Device Connections

Security

Server Settings

BACNET

Local Devices

ENTERPRISE ADMINISTRATION

Setup

SEQUENTIAL FUNCTION CHARTS

Settings

AWS INJECTOR

Settings

AZURE INJECTOR

Settings

EFM ABB TOTALFLOW

Settings

EFM EMERSON ROC

Settings

EFM EXPORT

Settings

GOOGLE CLOUD INJECTOR

Settings

IBM CLOUD INJECTOR

Settings

↑

🔍 Search...

⚙️ Config > > Azure Injector Settings

Trial Mode 1:56:38

Activate Ignition

Authentication

Enable Certificate Based Authentication

☐ Enable certificate based authentication instead of using a connection string

Password

The Connection String used for establishing a connection with the IoT Edge when using 'Connection String Authentication'. This is either the Connection String associated with the Child Device or with the Azure Edge Module.

Password

Re-type password for verification.

CA Certificate File

- none -

CA Certificate file currently in use on the IoT Edge instance. It is used for both Certificate and Connection String based authentication. This is the CA Device Certificate that was uploaded to the Azure Edge instance.

Client Certificate File

- none -

Client certificate file currently in use

Client Private Key File

- none -

Client private key file currently in use

Password

The password associated with the certificate's private key (optional)

Password

Re-type password for verification.

Hostname Verification

☐ Enable TLS Hostname Verification. This should be true on production systems.

MQTT Hostname

Device ID

The Device ID as provisioned in the Azure IoT Edge configuration

Module ID Config Option

ENVIRONMENT_VARIABLE

The method to use to configure the Module ID. This should be 'NONE' if using a 'child device' connection to Edge. It should be 'ENVIRONMENT_VARIABLE' if you want to pick up the Module ID from the 'IOTEDGE_MODULEID' environment variable. Otherwise, specify IGNITION_CONFIG and specify the Module ID in this configuration page.

Module ID

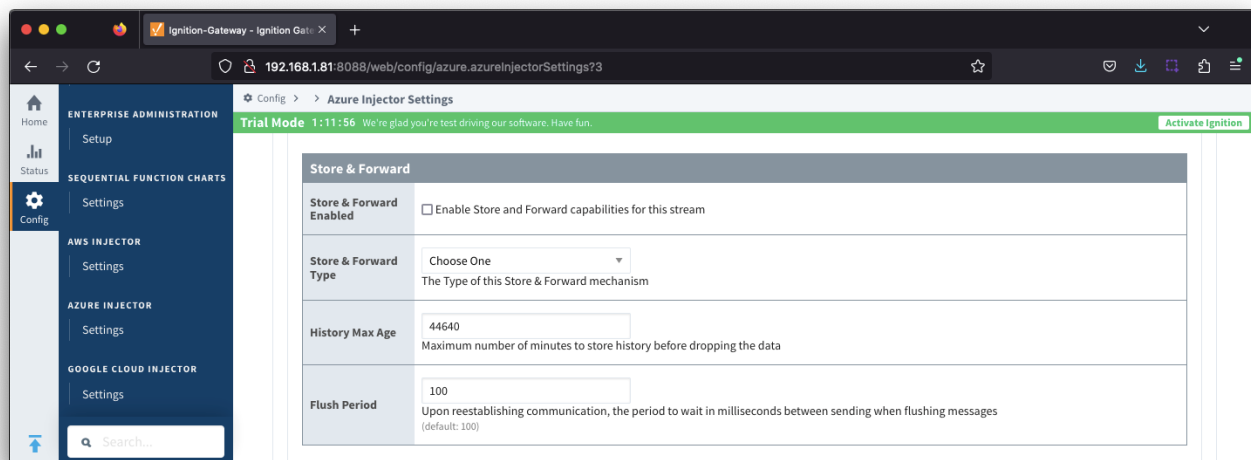
The Module ID as provisioned in the Azure IoT Edge configuration. This is only used if the 'Module ID Config Option' is IGNITION_CONFIG.

- **Enable Certificate Based Authentication**
 - Whether or not to use certificate based authentication.
 - If not using certificate based authentication, the 'Password' field must be used.
 - If certificate based authentication is used, the other Authentications fields must be used.
- **Password (required if not using certificate based authentication)**
 - This is the Azure IoT Edge connection string used to connect.
 - This is either the Connection string associated with the Child Device or with the Azure Edge Module
- **CA Certificate File**
 - The CA certificate file currently in use on the IoT Edge instance.
 - It is used for both Certificate and Connection String based authentication and is the CA Device Certificate that was uploaded to the Azure Edge instance.
 - The drop down is populated from a list of files that have been uploaded to the IoT Edge/Certificates tab.
- **Client Certificate File (required if using certificate based authentication)**
 - The client certificate file currently in use
 - The drop down is populated from a list of files that have been uploaded to the IoT Edge/Certificates tab.
- **Client Private Key File (required if using certificate based authentication)**
 - The client private key file currently in use
 - The drop down is populated from a list of files that have been uploaded to the IoT Edge/Certificates tab.
- **Password/Private key password**
 - The password used for the private key if one was specified for the key
- **MQTT Hostname (required if using certificate based authentication)**
 - This is the DNS endpoint name of your IoT Hub
- **Device ID (required if using certificate based authentication)**
 - The Device ID as provisioned in the Azure IoT Edge configuration
- **Module ID Config Option**
 - The method to use to configure the Module ID. This should be 'NONE' if using a 'child device' connection to Edge. It should be 'ENVIRONMENT_VARIABLE' if you want to pick up the Module ID from the 'IOTEDGE_MODULEID' environment variable. Otherwise, specify IGNITION_CONFIG and specify the Module ID in this configuration page.
- **Module ID**
 - The Module ID as provisioned in the Azure IoT Edge configuration. This is only used if the 'Module ID Config Option' is IGNITION_CONFIG.

Azure IoT Edges Settings - Store & Forward



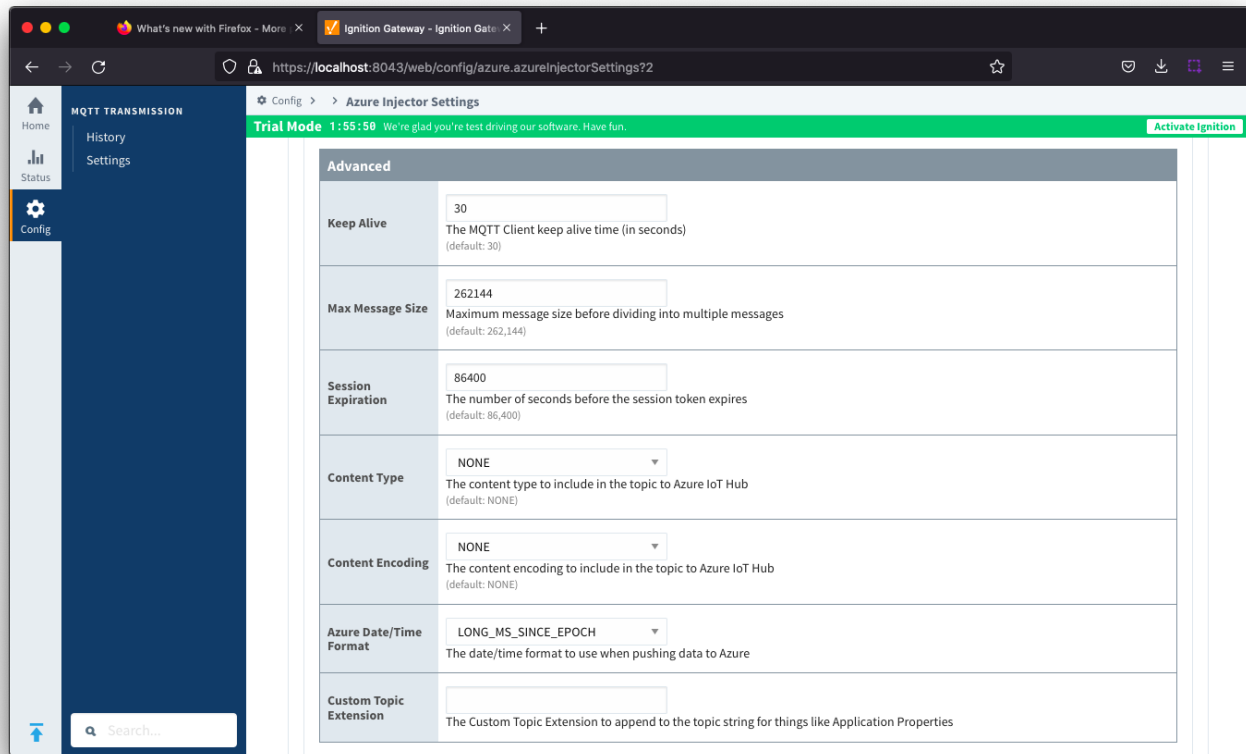
From release 4.0.19, major improvements have been made to the disk-backed History Store. As a result, Message Capacity has been deprecated and History Max Age added



- **Store & Forward Enabled**
 - Whether to enable Store & Forward capabilities for this endpoint
- **Store & Forward Type**
 - The type of the Store & Forward mechanism options: In_Memory and Disk_Back (available in release 4.0.17 and higher)
 - Data stored with an In_Memory Store & Forward will not be persisted across a module configuration change, module disable/enable, module restart or power loss.
 - Data stored with a Disk_Back Store & Forward will persist across a module configuration change, module disable/enable, module restart or power loss
- **Message Capacity - deprecated in 4.0.19**
 - The maximum number of messages to store before dropping the oldest historical messages

- **History Max Age**
 - The maximum number of minutes to store history before dropping the data
- **Flush Period**
 - The period of time to wait (in milliseconds) between sending when flushing messages

Azure IoT Edges Settings - Advanced



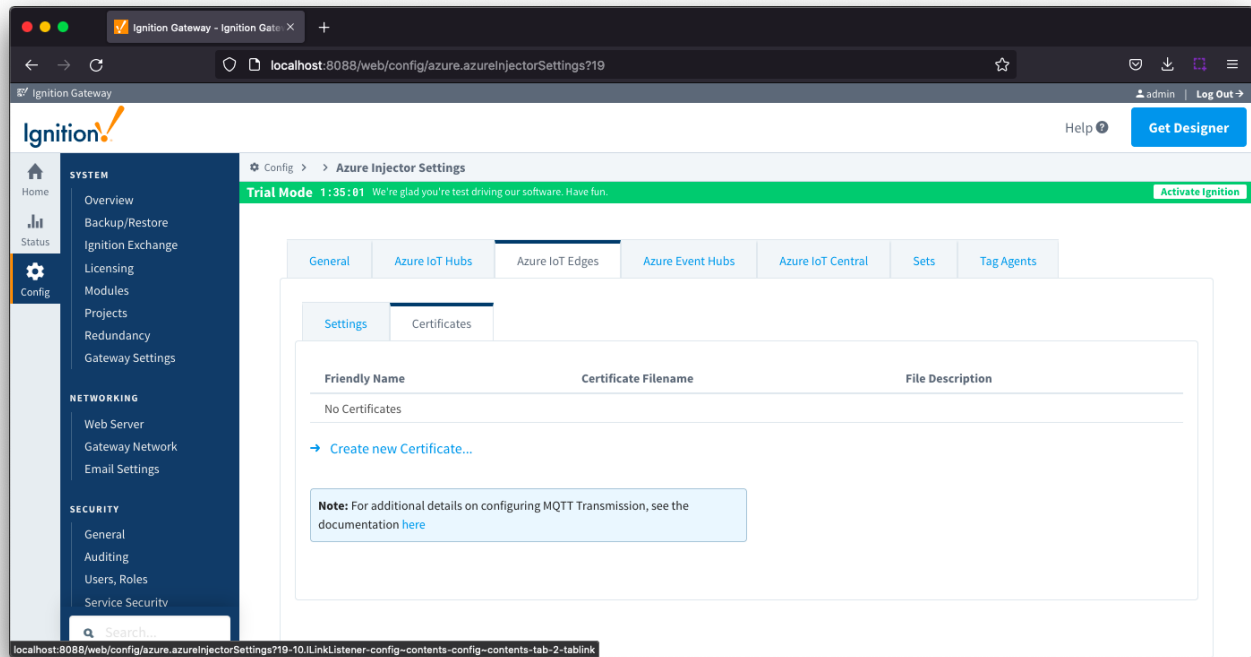
- **Keep Alive**
 - The MQTT keep alive in seconds
- **Max Message Size**
 - The maximum message size in bytes that any message can be when publishing to IoT Edge. Generally, this should match the max message size allowed by IoT Edge.
- **Session expiration**
 - How long in seconds to specify for token timeouts when not using certificate based authentication
- **Content Type**
 - The content type to include in the topic to Azure IoT Edge
 - NONE (default) - No content type header will be included with the message
 - APPLICATION_JSON - The application/json header will be included with the message and make the body of the message available for routing if content encoding is also not 'NONE'
- **Content encoding**
 - The content encoding to include in the topic to Azure IoT Edge
 - NONE (default) - No content encoding header will be included with the message
 - UTF_8 - The 'utf-8' header will be included with the message and make the body of the message available for routing if the content type is also set to APPLICATION_JSON
 - UTF_16 - The 'utf-16' header will be included with the message and make the body of the message available for routing if the content type is also set to APPLICATION_JSON
 - UTF_32 - The 'utf-32' header will be included with the message and make the body of the message available for routing if the content type is also set to APPLICATION_JSON
- **Azure Date/Time Format**
 - The date/time for mat to use when pushing messages to IoT Edge
 - LONG_MS_SINCE_EPOCH (default) - The timestamp values will all be as numbers in milliseconds since epoch (Jan 1, 1970) in UTC
 - STRING_AZURE_COMPAT - The timestamp will be pushed as described [here](#). This is useful when wanting to use 'edge' timestamps in Azure Time Series Insights.
- **Custom Topic Extension**
 - The Custom Topic Extension to append to the topic string for things like Application Properties

Azure IoT Edges - Certificates

This tab provides a list of the certificate or private keys loaded and available for certificate based authentication. This should generally include the root CA for your IoT Edge, the client certification file and the client private key file.

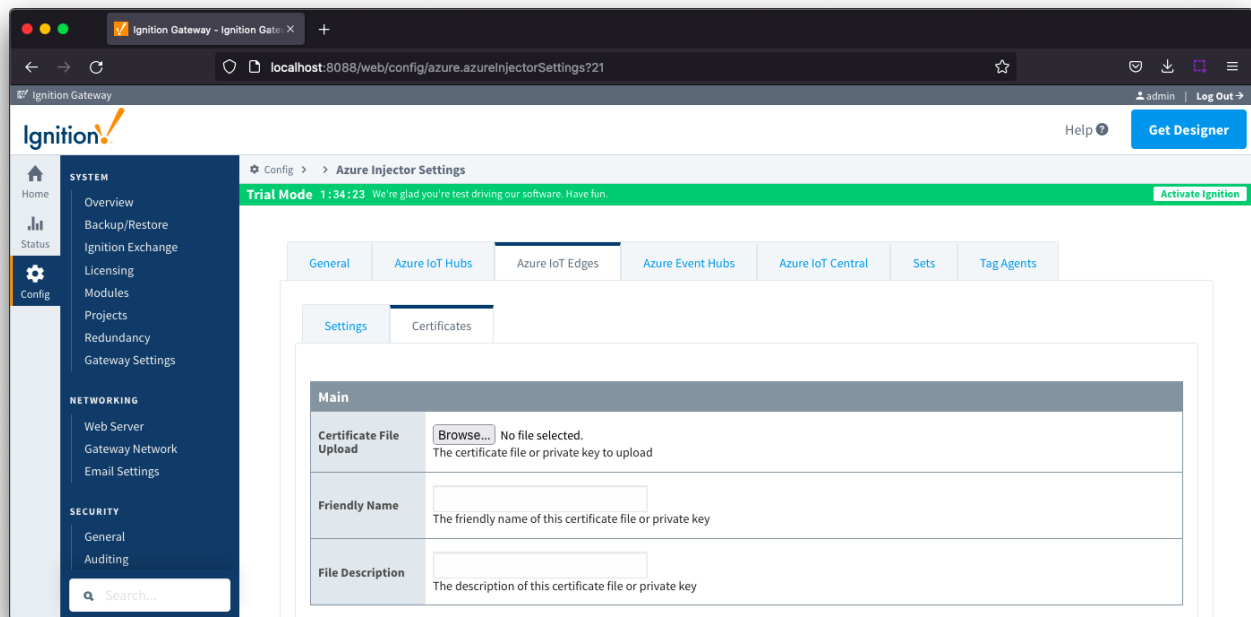


All certificate or private keys must be in PEM format. If using modules pre 4.0.9, any private key must also be in RSA PKCS1 format. If using modules 4.0.9 or greater, any private key must also be in either RSA PKCS1 or PKCS8 format.



Clicking on the 'Create new Certificate...' will bring up the following form to add a new certificate. The Certificates tab contains only a single [Main](#) section.

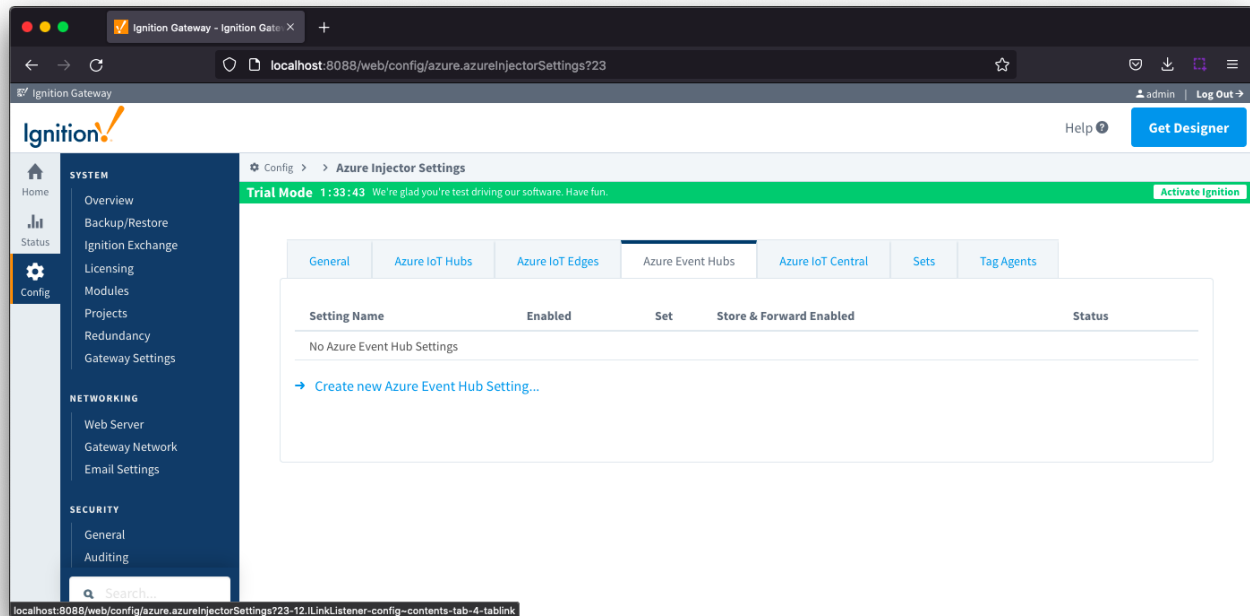
Azure IoT Edges Certificates - Main



- **Certificate File Upload**
 - Browse to the certificate or private key to upload
- **Friendly Name**
 - The friendly name of the certificate file or private key
- **File Description**
 - The description of the certificate file or private key

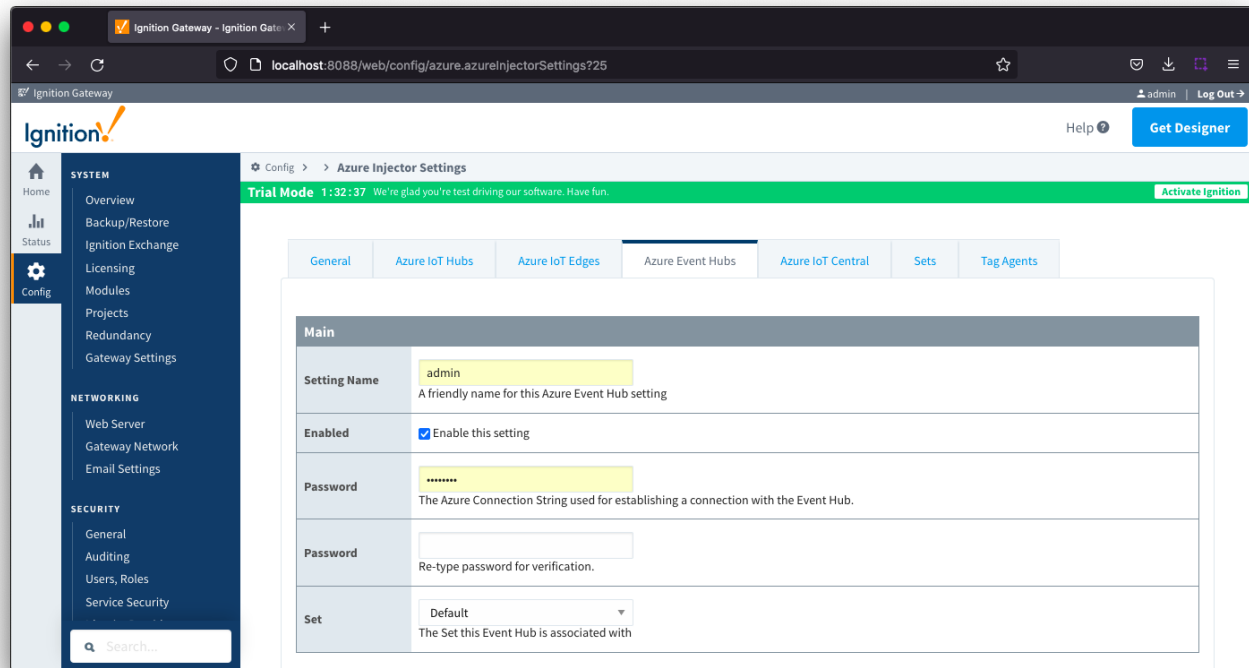
Azure Event Hubs

This tab provides a list of Azure Event Hub endpoints that the module should connect to to push tag data. One or more Azure Event Hub endpoints can be configured on this tab.



Clicking on the 'Create new Azure Event Hub ..' link will bring up the following form to add a new Azure Event Hub. The configuration sections available are [Main](#), [Store & Forward](#) and [Advanced](#)

Azure Event Hub - Main



- **Setting Name**
 - This is a friendly name of the Azure Event Hub used to easily identify it. This must also be unique.
- **Enabled**
 - Whether or not this connection is enabled.
- **Password/Connection String**
 - This is the Azure Event Hub connection string used to connect.
 - **NOTES:** Be sure to provide the connection string for the EventHub itself and not the EventHub Namespace. They both have connection strings, but the one for the EventHub will be of the form -

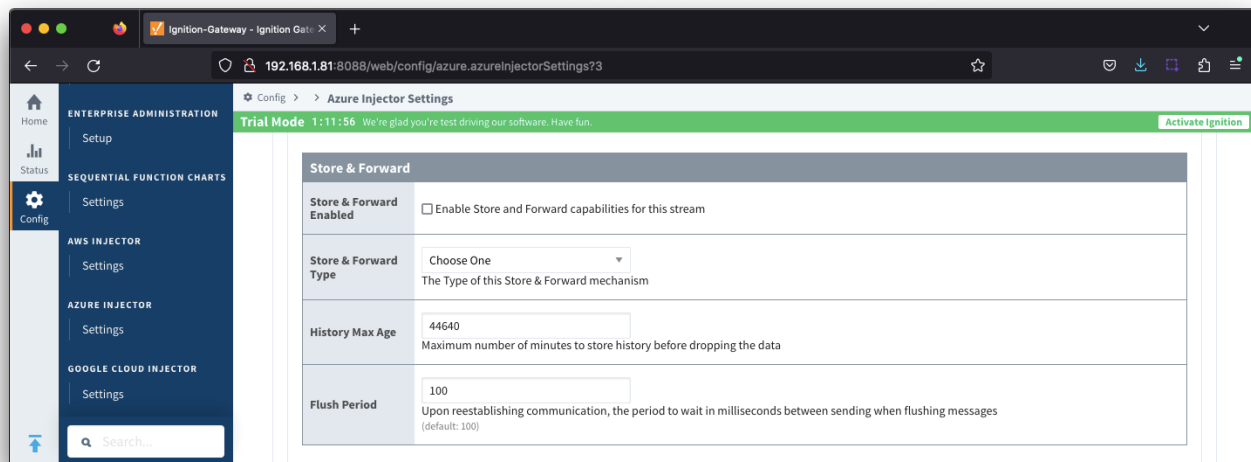
Endpoint=<YOUR_ENDPOINT>;SharedAccessKeyName=<YOUR_KEYNAME>;SharedAccessKey=<YOUR_KEY>;;
EntityPath=<YOUR_EVENTHUB_ENTITYPATH>

The Namespace connection string will not contain the entity path.
- **Set**
 - The Set to associate this Azure Event Hub connection with.

Azure Event Hub Settings - Store & Forward

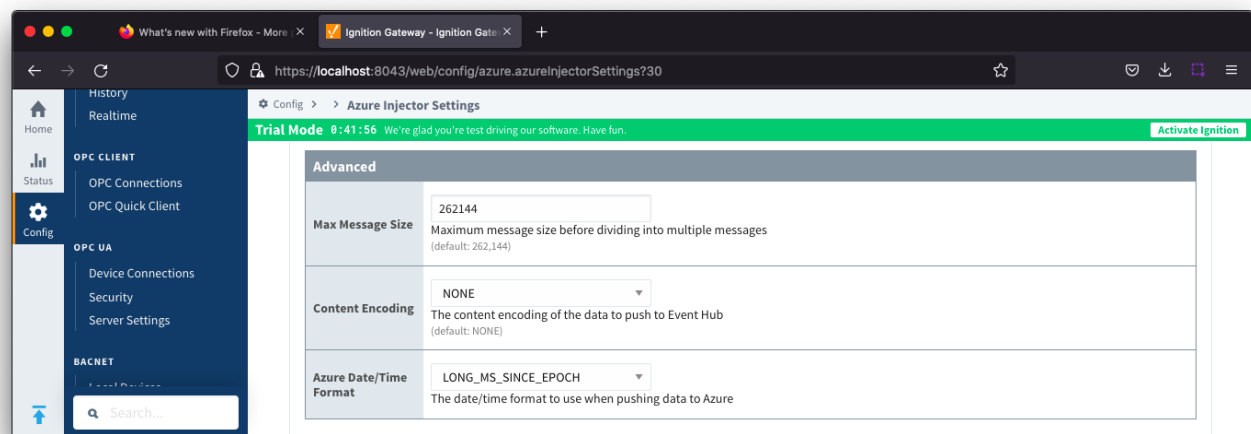


From release 4.0.19, major improvements have been made to the disk-backed History Store. As a result, Message Capacity has been deprecated and History Max Age added



- **Store & Forward Enabled**
 - Whether to enable Store & Forward capabilities for this endpoint
- **Store & Forward Type**
 - The type of the Store & Forward mechanism options: In_Memory and Disk_Backed (available in release 4.0.17 and higher)
 - Data stored with an In_Memory Store & Forward will not be persisted across a module configuration change, module disable/enable, module restart or power loss.
 - Data stored with a Disk_Backed Store & Forward will persist across a module configuration change, module disable/enable, module restart or power loss
- **Message Capacity - deprecated in 4.0.19**
 - The maximum number of messages to store before dropping the oldest historical messages
- **History Max Age**
 - The maximum number of minutes to store history before dropping the data
- **Flush Period**
 - The period of time to wait (in milliseconds) between sending when flushing messages

Azure Event Hub Settings - Advanced

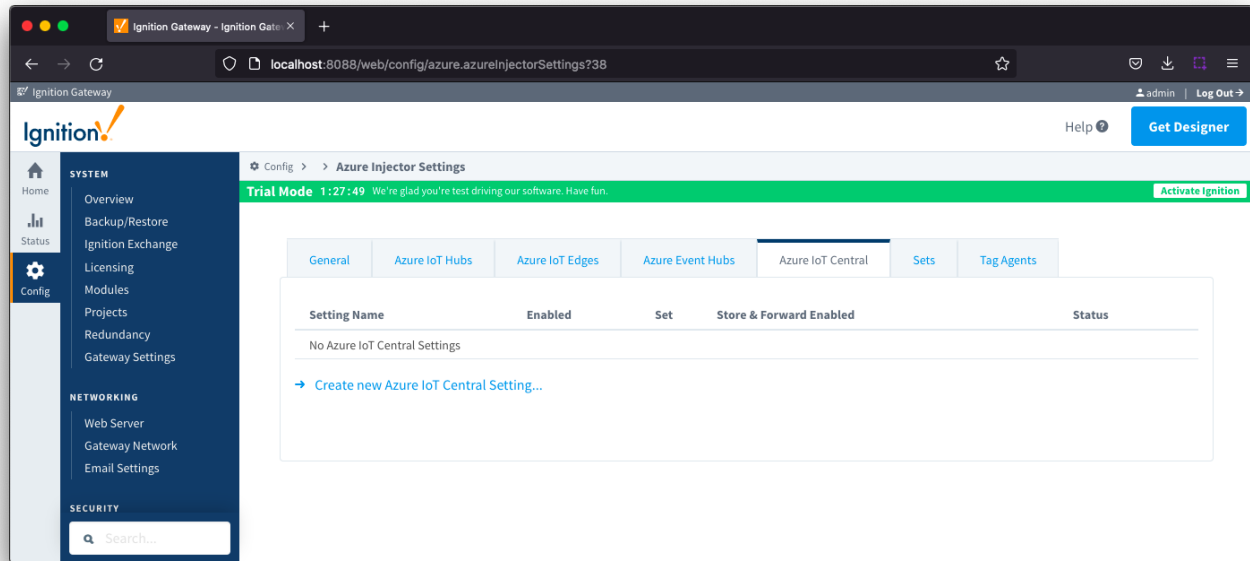


- **Max Message Size**
 - The maximum message size for the Azure Event Hub. Default is 262144 bytes (256KB). Generally, this should match the max message size allowed by the Event Hub.
 - EventHub Basic: 262144 bytes (256KB)
 - EventHub Standard or better: 1048576 bytes (1MB)
- **Content Encoding**
 - The content encoding of the data to push to Event Hub.
 - Current options are UTF_8, UTF_16 and UTF_32
- **Azure Date/Time Format**
 - The date/time format to use when pushing messages to IoT Hub
 - LONG_MS_SINCE_EPOCH (default) - The timestamp values will all be as numbers in milliseconds since epoch (Jan 1, 1970) in UTC

- **STRING_AZURE_COMPAT** - The timestamp will be pushed as described [here](#). This is useful when wanting to use 'edge' timestamps in Azure Time Series insights.
 - See [Pushing Data to Azure Time Series Insights](#) tutorial for more detail

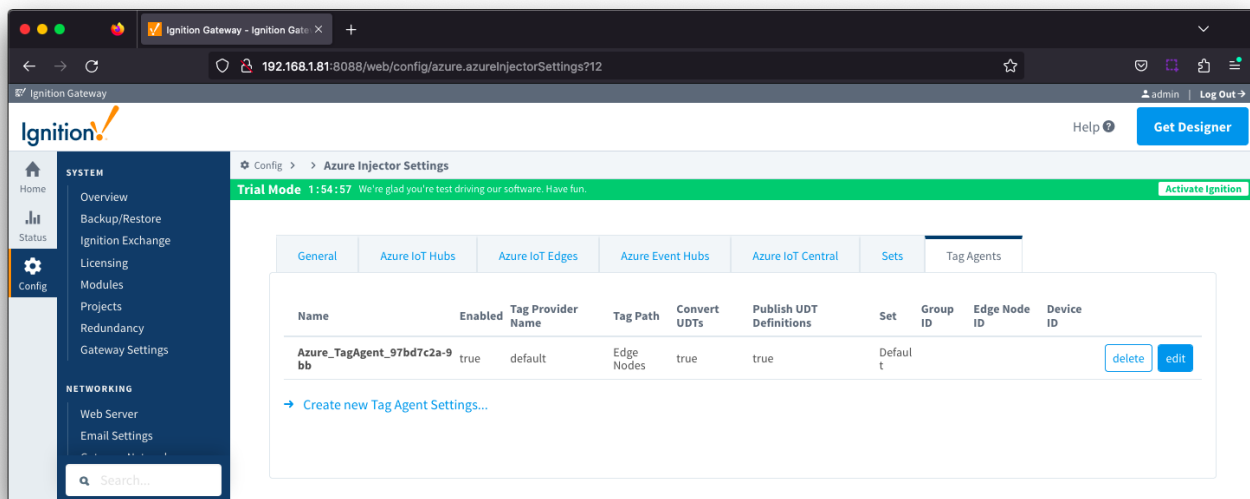
Azure IoT Central

This tab provides a list of Azure IoT Central endpoints that the module should connect to to push tag data. One or more Azure IoT Central endpoints can be configured on this tab.



Clicking on the 'Create new Azure IoT Central Setting..' link will bring up the following form to add a new Azure IoT Central. The configuration sections available are [Main](#), [Store & Forward](#) and [Advanced](#).

Azure IoT Central - Main



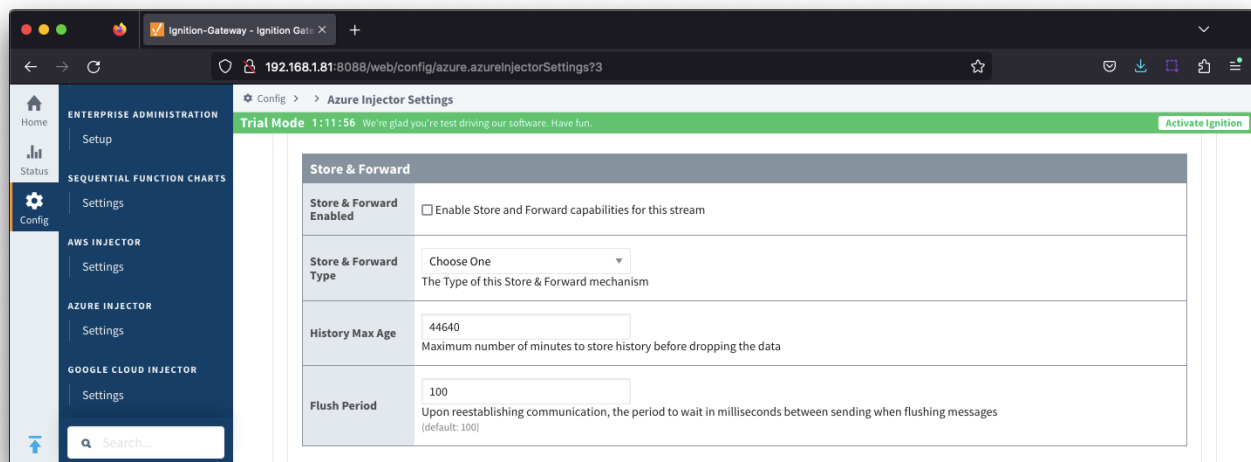
- **Setting Name**
 - This is a friendly name of the Azure IoT Central used to easily identify it. This must also be unique.
- **Enabled**
 - Whether or not this connection is enabled.
- **Scope ID**
 - The Azure IoT Central Scope ID.

- Found in the IoT Central -> Administration Device Settings.
- **Password**
 - The Azure Enrollment Group Symmetric Key.
 - Found in the IoT Central -> Security -> Permissions -> Device Connection Groups -> [SAS-IoT-Devices] -> SAS -> Primary/Second key.
- **Global Endpoint**
 - The global endpoint of the IoT Central connection. Default is global.azure-devices-provisioning.net
- **Provisioned Device ID**
 - The provisioned Device ID associated with this IoT Central connection.
- **Model ID**
 - The Model ID associated with this IoT Central connection.
- **Set**
 - The Set associated with this IoT Central connection.

Azure IoT Central - Store & Forward

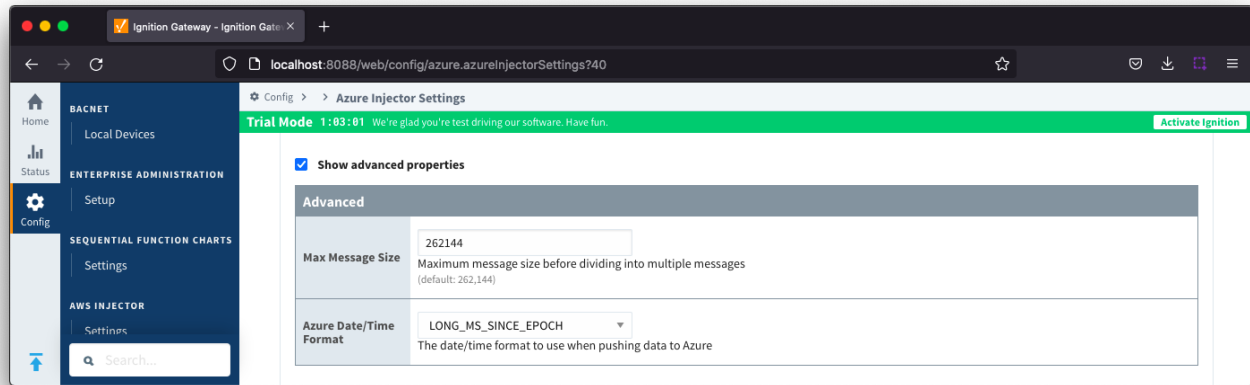


From release 4.0.19, major improvements have been made to the disk-backed History Store. As a result, Message Capacity has been deprecated and History Max Age added



- **Store & Forward Enabled**
 - Whether to enable Store & Forward capabilities for this endpoint
- **Store & Forward Type**
 - The type of the Store & Forward mechanism options: In_Memory and Disk_Backed (available in release 4.0.17 and higher)
 - Data stored with an In_Memory Store & Forward will not be persisted across a module configuration change, module disable/enable, module restart or power loss.
 - Data stored with a Disk_Backed Store & Forward will persist across a module configuration change, module disable/enable, module restart or power loss
- **Message Capacity - deprecated in 4.0.19**
 - The maximum number of messages to store before dropping the oldest historical messages
- **History Max Age**
 - The maximum number of minutes to store history before dropping the data
- **Flush Period**
 - The period of time to wait (in milliseconds) between sending when flushing messages


Azure IoT Central - Advanced



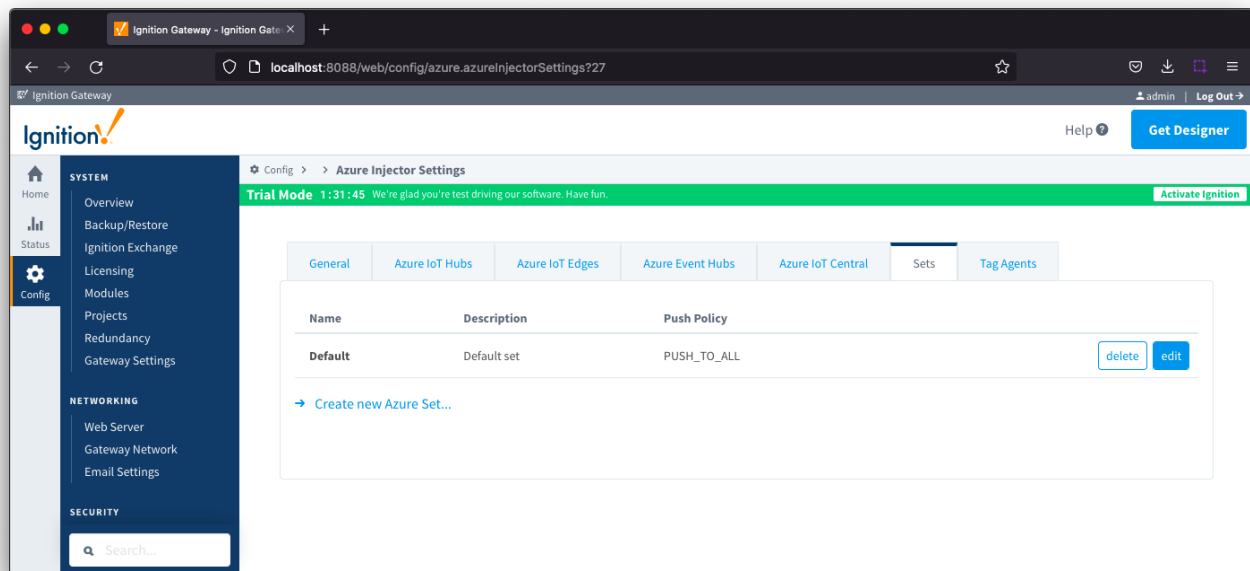
- **Max Message Size**
 - The maximum message size in bytes that any message can be when pushing to IoT Central.
- **Azure Date/Time Format**
 - The date/time format to use when pushing messages to IoT Hub
 - LONG_MS_SINCE_EPOCH (default) - The timestamp values will all be as numbers in milliseconds since epoch (Jan 1, 1970) in UTC
 - STRING_AZURE_COMPAT - The timestamp will be pushed as described [here](#). This is useful when wanting to use 'edge' timestamps in Azure Time Series insights.
 - See [Pushing Data to Azure Time Series Insights](#) tutorial for more details

Sets

This tab contains a list of Azure Sets. Each set represents a grouping of Azure IoT/Event Hub endpoints. When a set is referenced by a Tag Agent, the Agent will push Tag data to all Azure IoT/Event Hub endpoints contained within that Set.

 The Sets are disjoint, meaning that a single Azure IoT/Event Hub endpoint cannot be in more than one set.

Out of the box the Azure Injector module will have one "Default" set defined. Additional Sets can be configured for situations where multiple Tag Agents will need to push to different Azure IoT Hub endpoints.



Clicking on the 'Create new Azure Set ..' link will bring up the following form to add a new Set. The configuration section available is [Main](#)

Sets - Main

The screenshot shows the Ignition Gateway web interface in a browser window. The address bar shows the URL: localhost:8088/web/config/azure.azureinjectorSettings?29. The page title is 'Ignition Gateway'. The left sidebar contains a navigation menu with sections: SYSTEM (Overview, Backup/Restore, Ignition Exchange, Licensing, Modules, Projects, Redundancy, Gateway Settings), NETWORKING (Web Server, Gateway Network, Email Settings), and SECURITY. The main content area is titled 'Azure Injector Settings' and has a breadcrumb 'Config > Azure Injector Settings'. A green banner at the top of the main content area says 'Trial Mode 1:31:04 We're glad you're test driving our software. Have fun.' and an 'Activate Ignition' button. Below the banner are tabs: General, Azure IoT Hubs, Azure IoT Edges, Azure Event Hubs, Azure IoT Central, Sets (selected), and Tag Agents. The 'Sets' tab is active, showing a 'Main' section with three fields: 'Name' (with a text input containing 'New Set' and a description 'The friendly name of this Set'), 'Description' (with a text input and a description 'Description of this Set'), and 'Push Policy' (with a dropdown menu set to 'PUSH_TO_ALL' and a description 'The Push Policy defines whether all cloud end-points in a set will be pushed to or only one will at a time').

- **Name**
 - This is the friendly name of the set used to easily identify it.
- **Description**
 - This is a friendly description of the set.
- **Push Policy**
 - This defines which endpoints to push to.
 - If PUSH_TO_ALL is selected, every endpoint that is part of this set will receive all messages.
 - If PUSH_TO_ANY is selected, only one of the endpoints that is part of this set will receive any given message. PUSH_TO_ANY is useful when adding endpoint configurations to increase the throughput of the Injector.

Tag Agents

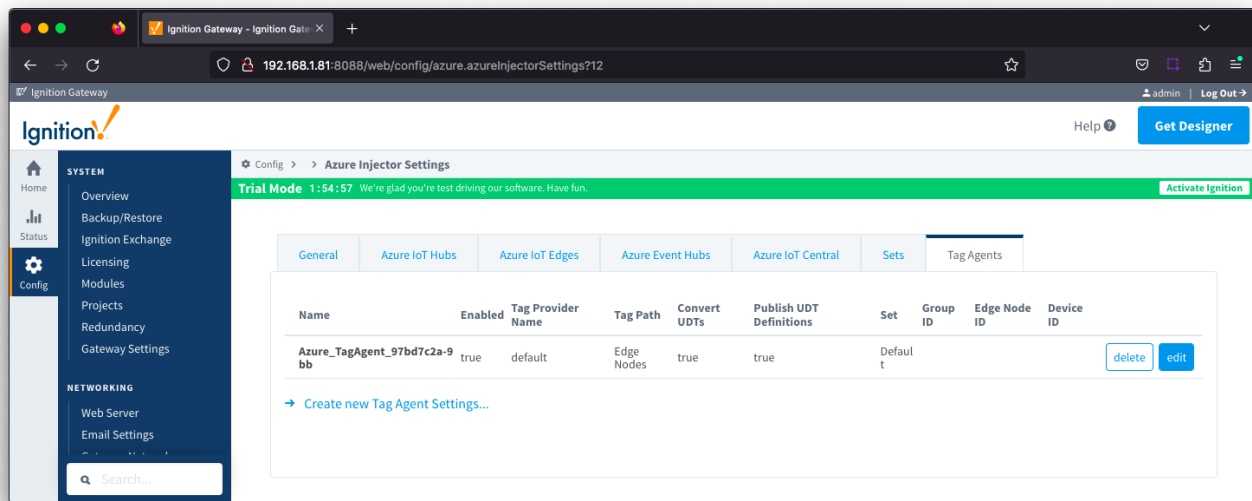
Tag Agents define which tags will be picked up from the Ignition tag tree, converted to a JSON representation and pushed to one or more Azure endpoints.

Tag Agents will monitor tags from a specific Tag Provider and, optionally, a specific Tag Path. If the tag folder hierarchy has been constructed as Group ID, Edge Node ID, and Device ID, then these will automatically be used when building up the [JSON message payload](#) which includes these in the topic.



Review the [Cloud Injector Tag Agents and Tag Trees](#) describing how Tag Agent configurations interact with Ignition tag trees

If your tag folder hierarchy does not conform to this structure, you can explicitly define these required elements under the [SparkPlug Settings](#) section to be included when building the message topic.



Clicking on the 'Create new Tag Agent Settings..' link will bring up the following form to add a new Tag Agent. The configuration sections available are [Agent Settings](#), [Sparkplug Settings](#) and [Advanced](#)

Tag Agents - Agent Settings

The screenshot shows the 'Agent Settings' form for a Tag Agent. The left sidebar is the same as in the previous image. The main content area is titled 'Config > Azure Injector Settings'. A green banner at the top indicates 'Trial Mode 1:48:54'. Below this, there are tabs for General, Azure IoT Hubs, Azure IoT Edges, Azure Event Hubs, Azure IoT Central, Sets, and Tag Agents. The 'Tag Agents' tab is active, and the 'Agent Settings' sub-tab is selected. The form contains the following fields:

- Name:** 'Azure_TagAgent_97bd7c2a-9bb'. Description: 'A unique name for the tag agent'.
- Enabled:** ☒ 'Enable Tag Agent'. Description: '(default: true)'.
- Tag Provider Name:** 'default'. Description: 'The Name of the tag provider'.
- Tag Path:** 'Edge Nodes'. Description: 'A path to the root folder where the tag tree starts (optional)'.
- Push Trigger:** 'EVENT_DRIVEN'. Description: 'The trigger to use when pushing data. EVENT_DRIVEN (default) means to push on change events. PERIODIC means to push all data on a periodic basis.'.
- Tag Pacing Period:** '1000'. Description: 'If using an EVENT_DRIVEN Push trigger, the waiting period in milliseconds after an initial tag change event before pushing all changed tags. If using PERIODIC, the number of milliseconds to wait between pushing all data. (default: 1,000)'.
- Convert UDTs:** ☒ 'Converts UDT members to normal Tags'.
- Publish UDT Definitions:** ☐ 'Publish UDT Definitions in BIRTH'.
- Optimize UDTs:** ☒ 'Optimizes UDT payload sizes in NDATA and DDATA payloads'.
- Set:** 'Default'. Description: 'The Set this Agent is associated with'.

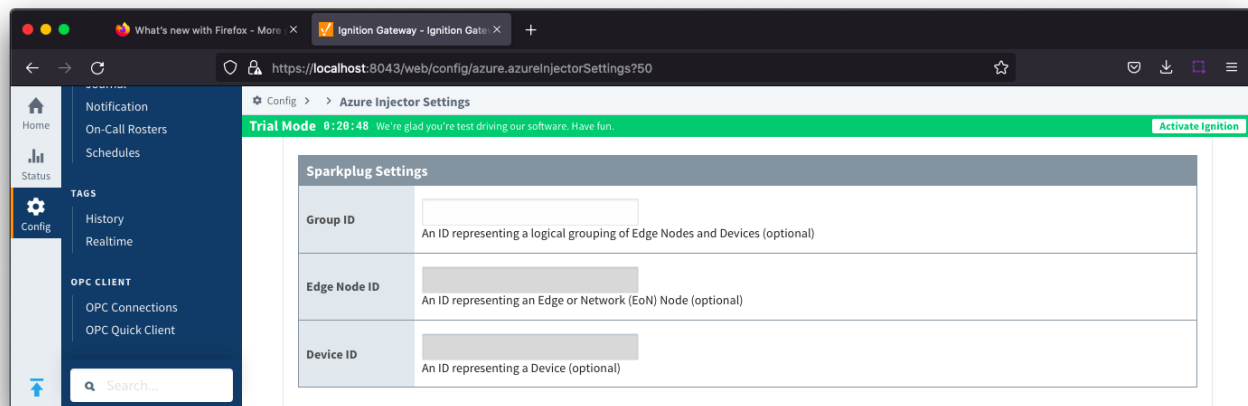
- **Name**
 - A unique name for the tag agent.
- **Enabled**
 - Sets whether the Tag Agent is enabled or disabled. If disabled, the Tag Agent will not run and no data will be pushed to any configured endpoints.
- **Tag Provider Name**
 - The name of the Tag provider containing the tags.
- **Tag Path**
 - An optional path to the root folder where the tag tree starts.
- **Push Trigger**

- Defines what triggers a push to the cloud endpoint
 - EVENT_DRIVEN (default) - when a tag change event (value or quality) occurs, and no pending push exists, tag events will be aggregated for the 'Tag Pacing Period' before being pushed.
 - PERIODIC - will push the latest data for all tags associated with the Agent every 'Tag Pacing Period'. With this option, only one event per tag will be sent and tag change events will not be captured.
- Tag Pacing Period**
 - The buffer period, in milliseconds, that Tag events will be aggregated into a single payload before pushing.
- Convert UDTs**
 - Whether to convert UDT members to normal Tags before publishing. If enabled the Tags representing the UDT member will retain their member path prefixed by the UDT Instance name.
- Publish UDT Definitions**
 - This will only be used if 'Convert UDTs' is false
 - Whether or not to push the UDT Definitions in the the NBIRTH messages
- Optimize UDTs**
 - This will only be used if 'Convert UDTs' is false
 - Whether or not to 'convert UDTs' only for DATA messages.
- Set**
 - The Set of Azure IoT Hub endpoints that the Tag Agent will push to.



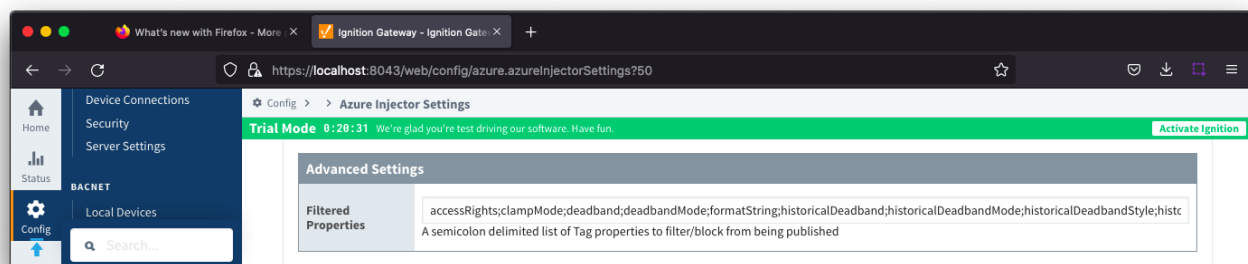
Review the [Managing UDTs through Injector Tag Agents](#) for more details on the Convert UDTs, Publish UDT Definitions and Optimize UDTs parameters

Tag Agents - Sparkplug Settings



- Group ID**
 - An ID representing a logical grouping of MQTT Edge Of Network (EoN) Nodes and Devices into the infrastructure.
- Edge Node ID**
 - An ID that uniquely identifies the MQTT Edge Of Network (EoN) Node within the infrastructure.
- Device ID**
 - An optional ID that uniquely identifies a Device within the infrastructure.

Tag Agents - Advanced



- Filtered Properties**

- A semicolon delimited list of Tag properties to filter/block from being published. These should typically not be modified unless there is an explicit requirement that a specific property is needed to be added or removed from the default.