HighByte® Intelligence Hub

Version 2.2

User Guide

Table of Contents

1. System Requirements

The following system requirements must be met to run HighByte Intelligence Hub:

1.1 Hardware

- 1.4 GHz Processor
- 1 GB RAM
- 500 MB Available Disk Space
- Network Capable (TCP)

Note: These are minimal system requirements. Actual hardware requirements will vary based on product configuration.

1.2 Operating System

- Windows Server 2012 / 2016 / 2019
- Windows 8 / 10
- Linux
- MacOS

1.3 Browser

The Intelligence Hub is configured using a browser, and by default is accessible on http://localhost:45245. The following browser versions are fully supported.

- Chrome v60+
- Firefox v60+
- Microsoft Edge v79+
- Safari v12+

2.0 Required Dependencies

The following dependencies must be installed to run HighByte Intelligence Hub:

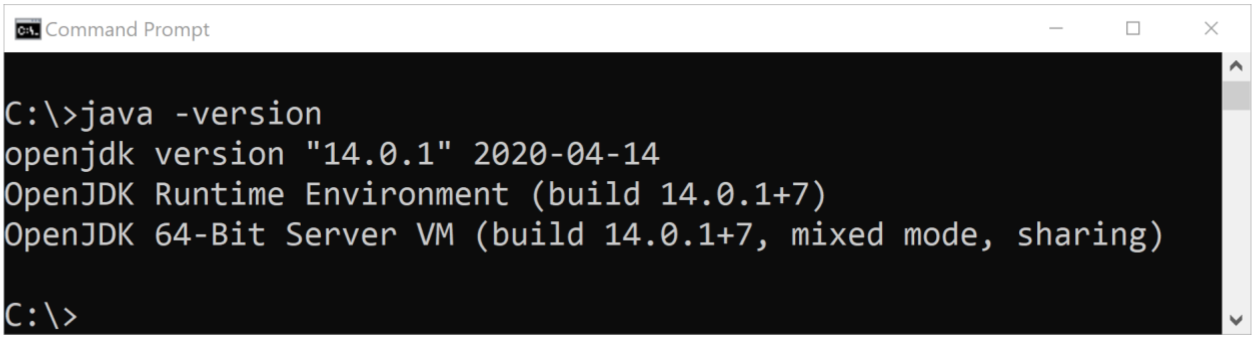
> Check out the HighByte YouTube Channel for video tutorials on how to install the Intelligence Hub and any required dependencies on Windows 10 and Ubuntu Linux.

2.1 Java Runtime Environment (JRE)

The Intelligence Hub runtime requires Java SE 11 or later. HighByte recommends using the OpenJDK 14 or Oracle JRE 1.11 distribution.

Step 1. Check to see if a JRE is already installed

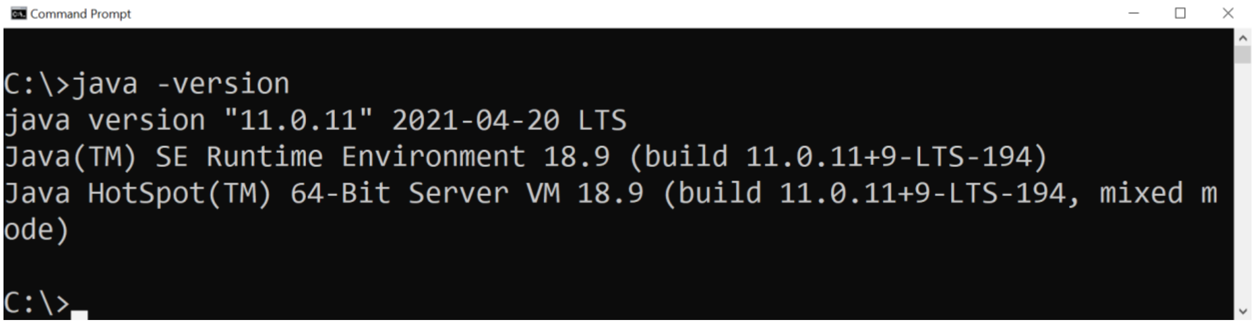
A. From a command or terminal prompt execute *java -version*
   1. If the command fails due to *java* not existing, proceed to Step 2.
   2. If *openjdk version* is 14.0.xxx or greater (Figure 1a) OR *java version* is 1.8.xxx or greater (Figure 1b), a valid JRE is already installed and you can skip the remaining steps in this section. Otherwise, proceed to Step 2a (OpenJDK install) or Step 2b (Oracle JRE install). See Figure 1.



```
C:\>java -version
openjdk version "14.0.1" 2020-04-14
OpenJDK Runtime Environment (build 14.0.1+7)
OpenJDK 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)

C:\>
```

Figure 1a. OpenJDK Distribution



```
C:\>java -version
java version "11.0.11" 2021-04-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.11+9-LTS-194)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.11+9-LTS-194, mixed m
ode)

C:\>
```

Figure 1b. Oracle Distribution

Step 2a. Install OpenJDK JRE

A. Download OpenJDK 14 from the web. Currently available at OpenJDK 14.

B. Follow the instructions that accompany the OS dependent distribution that was downloaded to install the JRE for your system.
C. Close any command or terminal prompt windows that are currently open.
D. Re-run Step 1 to verify the JRE was properly installed.

Step 2b. Install Oracle JRE

A. Download JRE SE Version 1.11.xxx from the web. Currently available at Oracle Java SE Downloads. Please review Oracle's latest license to ensure compliance for use within your organization.
B. Follow the instructions that accompany the OS dependent distribution that was downloaded to install the JRE for your system.
C. Close any command or terminal prompt windows that are currently open.
D. Re-run Step 1 to verify the JRE was properly installed.

3.0 Intelligence Hub Runtime

3.1 Installation

> Check out the HighByte YouTube Channel for video tutorials on how to install the Intelligence Hub and any required dependencies on Windows 10 and Ubuntu Linux.

Step 1. Install Required Dependencies

A. See section *Required Dependencies* above.

Step 2. Install Runtime

A. Extract *HighByte-Intelligence-Hub-2.x* (provided with this documentation) to a location where you have read/write privileges.
B. Copy the extracted contents of *HighByte-Intelligence-Hub-1.x/runtime* to a location where you have read/write/execute privileges in order run the HighByte Intelligence Hub runtime.

3.1.1 Upgrading

> Check out the HighByte YouTube Channel for video tutorials on how to upgrade the Intelligence Hub to the latest version.

Method A: Preserving prior installation(s)

Step 1. Install a fresh copy of the latest version to a new folder (e.g., /HighByte/IntelligenceHub-2.x) by following the instructions in 3.1.

Step 2. Stop the previous installation's runtime service by running the *stop-[OS]* command file for your operating system.

Step 3. Redirect the new installation's application and store and forward data directories

A. If you have not already done so in an earlier install or upgrade, create a common directory for application and store and forward directories that all HighByte Intelligence Hub installations will share (e.g., /HighByte/IntelligenceHub/AppData and /HighByte/IntelligenceHub/StoreForwardData). Next, locate and copy the preexisting application data files (including intelligencehub-certificatestore.pkcs12, intelligencehub-configuration.json, intelligencehub-events.log, intelligencehub-state.dat and intelligencehub-users.dat) to the application data directory. Finally, if you are using the store and forward capabilities of the product, locate and copy the preexisting store and forward files (including all files with a .db extension) to the store and forward data directory.
B. Rename the *intelligencehub-settings.json.template* file located in the new installation folder to *intelligencehub-settings.json*. Open the *intelligencehub-settings.json* file and set the application and store and forward data directories. An example is shown below:

```
{
 "settings": {
   "directories" : {
     "appData" : "/HighByte/IntelligenceHub/AppData ",
     "storeForwardData" : "/HighByte/IntelligenceHub/StoreForwardData "
   },
```

…

Note: On Windows you will need to escape all forward slashes as they have special meaning in JSON (the format of this settings file). For example, "appData" : "C:\\HighByte\\IntelligenceHub\\AppData".

When complete, save and close *intelligencehub-settings.json*.

Method B: Overwrite prior installation

Step 1. Extract *HighByte-Intelligence-Hub-2.x* (provided with this documentation) to a location where you have read/write privileges.

Step 2. Stop the previous installation's runtime service by running the *stop-[OS]* command file for your operating system.

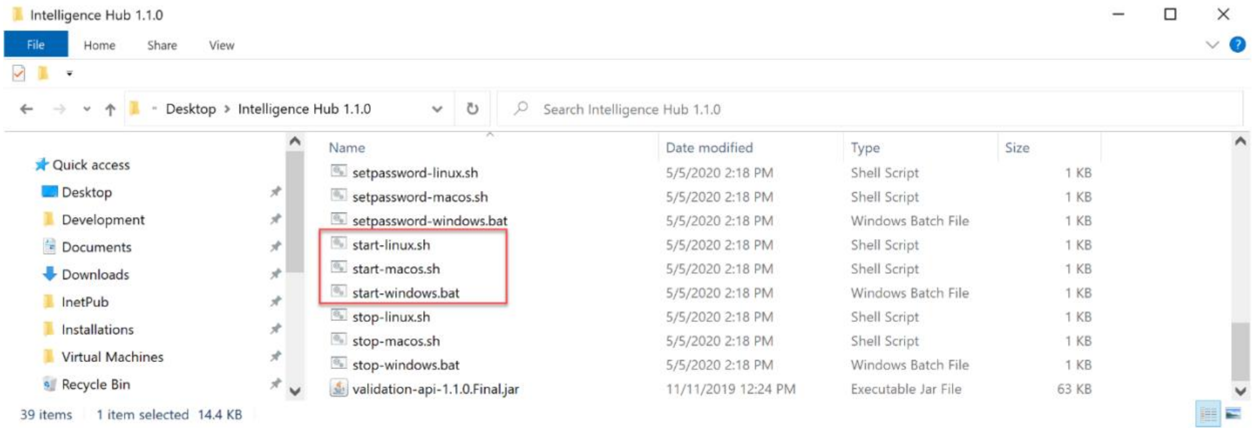
Step 3. Copy the extracted contents of *HighByte-Intelligence-Hub-1.x/runtime* to the location where you previously installed the HighByte Intelligence Hub runtime and overwrite all pre-existing files.

3.2 Usage

The runtime currently executes as a console-based service that requires the user to start and stop it manually through command files included with the product. During execution, the runtime will emit messages that indicate informational, warning, and error events to the console window to provide the user with relevant feedback. These messages are also written to a log file for archival purposes.

3.2.1 Starting the Runtime

A. Go to the directory where you extracted the contents of *HighByte-Intelligence-Hub-2.x/runtime*.
B. Start the runtime by *executing* the *start-[OS]* file associated with your OS (e.g., start-windows.bat, ./start-linux.sh, ./start-macos.sh). See Figure 2.



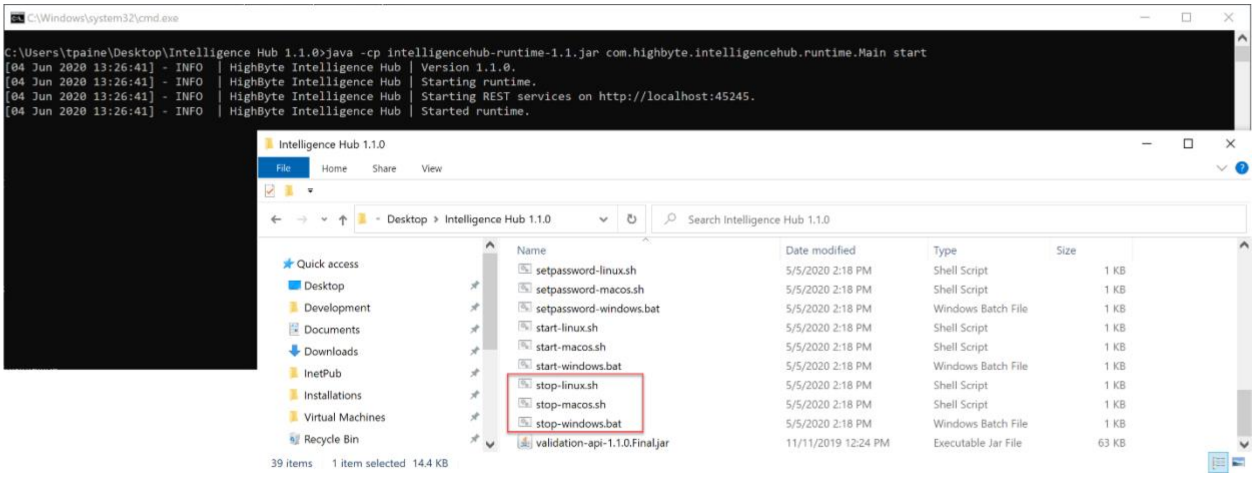
Name	Date modified	Type	Size
setpassword-linux.sh	5/5/2020 2:18 PM	Shell Script	1 KB
setpassword-macos.sh	5/5/2020 2:18 PM	Shell Script	1 KB
setpassword-windows.bat	5/5/2020 2:18 PM	Windows Batch File	1 KB
start-linux.sh	5/5/2020 2:18 PM	Shell Script	1 KB
start-macos.sh	5/5/2020 2:18 PM	Shell Script	1 KB
start-windows.bat	5/5/2020 2:18 PM	Windows Batch File	1 KB
stop-linux.sh	5/5/2020 2:18 PM	Shell Script	1 KB
stop-macos.sh	5/5/2020 2:18 PM	Shell Script	1 KB
stop-windows.bat	5/5/2020 2:18 PM	Windows Batch File	1 KB
validation-api-1.1.0.Final.jar	11/11/2019 12:24 PM	Executable Jar File	63 KB

Figure 2

C. The runtime will continue to run until stopped.

3.2.2 Stopping the Runtime

A. Go to the directory where you extracted the contents of *HighByte-Intelligence-Hub-2.x/runtime*.
B. Stop the runtime by *executing* the *stop-[OS]* file associated with your OS (e.g., stop-windows.bat, ./stop-linux.sh, ./stop-macos.sh). See Figure 3.



```
C:\Users\tpaine\Desktop\Intelligence Hub 1.1.0>java -cp intelligencehub-runtime-1.1.jar com.highbyte.intelligencehub.runtime.Main start
[04 Jun 2020 13:26:41] - INFO  | HighByte Intelligence Hub | Version 1.1.0.
[04 Jun 2020 13:26:41] - INFO  | HighByte Intelligence Hub | Starting runtime.
[04 Jun 2020 13:26:41] - INFO  | HighByte Intelligence Hub | Starting REST services on http://localhost:45245.
[04 Jun 2020 13:26:41] - INFO  | HighByte Intelligence Hub | Started runtime.
```

Name	Date modified	Type	Size
setpassword-linux.sh	5/5/2020 2:18 PM	Shell Script	1 KB
setpassword-macos.sh	5/5/2020 2:18 PM	Shell Script	1 KB
setpassword-windows.bat	5/5/2020 2:18 PM	Windows Batch File	1 KB
start-linux.sh	5/5/2020 2:18 PM	Shell Script	1 KB
start-macos.sh	5/5/2020 2:18 PM	Shell Script	1 KB
start-windows.bat	5/5/2020 2:18 PM	Windows Batch File	1 KB
stop-linux.sh	5/5/2020 2:18 PM	Shell Script	1 KB
stop-macos.sh	5/5/2020 2:18 PM	Shell Script	1 KB
stop-windows.bat	5/5/2020 2:18 PM	Windows Batch File	1 KB
validation-api-1.1.0.Final.jar	11/11/2019 12:24 PM	Executable Jar File	63 KB

Figure 3

C. The runtime will finish any remaining tasks and shutdown.

3.2.3 Application-Level Settings

Application-level settings are set in the *intelligencehub-settings.json* file located in the runtime directory. The installation package includes a template file for these settings named *intelligencehub-settings.json.template* to ensure future updates do not overwrite any custom application settings.

A. Go to the directory where you extracted the contents of *HighByte-Intelligence-Hub-2.x/runtime*.
B. Open the *intelligencehub-settings.json* file with any text editor. If the file does not exist, copy or rename *intelligencehub-settings.json.template* to *intelligencehub-settings.*json. Available settings are as follows:

Setting	Description
directories.appData	Specifies the location for loading and saving configuration, event log, user management, and other application data files. If this setting is not specified or is invalid, application data files will be stored in the runtime directory. Note: Specifying a Windows path requires any path delimiters ('\\') to be properly escaped. For example, a path to C:\Program Data\HighByte\Intelligence Hub should be set as "C:\\\\ProgramData\\\\HighByte\\\\Intelligence Hub".
directories.storeForwardData	Specifies the location for loading and saving store and forward data files. If this setting is not specified or is invalid, store and forward data files will be stored in the runtime directory. Note: Specifying a Windows path requires any path delimiters ('\\') to be properly escaped. For example, a path to C:\ProgramData\HighByte\Intelligence Hub should be set as "C:\\\\ProgramData\\\\HighByte\\\\Intelligence Hub".

configuration.scheme	Specifies whether the REST based configuration API should be exposed via *http* or *https*.
configuration.port	Specifies which http/https *port* the REST based configuraton API listens on.
configuration.rootDirectory	Specifies the location for the product configuration component files for self-hosting. If this setting is not specified it defaults to a relative path used in the packaged installer (i.e., ../configuration). If the setting is invalid or the path is invalid, the runtime will not self-host the configuration component. Note: Specifying a Windows path requires any path delimiters ('\') to be properly escaped. For example, a path to C:\ProgramData\HighByte\Intelligence Hub should be set as "C:\\ProgramData\\HighByte\\Intelligence Hub".
configuration.centralConfig	Specifies if the hub is in central configuration mode, allowing it to be used to remotely configure other hubs. Set to True to enable this capability. The default setting is False.
configuration.autoSaveInterval	Specifies the maximum interval in seconds that the configuration file is saved after a configuration change is made. Defaults to 60 seconds. The minimum is 5 seconds and the maximum is 3600 seconds (1 hour).
configuration.backupCopies	Specifies the maximum number of configuration copies to backup. The default is 50. The minum value is 0 (no backups) and the maximum is 100. When enabled, a 'backups' directory is created in the application directory. The existing configuration file is copied to the backup directory prior to saving any new changes to the configuration. If the maximum backups already exist, the oldest backup (based on the last time the file was editted) is deleted from the backup directory.
log.fileSizeMB	Specifies the maximum event log file size in megabytes. The valid range is 10–1000. If this setting is not specified or is invalid, a value of 100 will be used. Once the file reaches the limit, the oldest events are moved into a backup file (.bak) and a new event log file is created. If a backup file already exists, it will be overwritten.
log.logAuditEvents	Specifies if the hub logs audit events for objects (Connections, Models, etc) that are changed via the REST API. When enabled, audit events are logged for creation, update, and delete events. These events are logged to the event log and are of type AUDIT. Defaults to False.
redundancy.backup.enabled	Specifies if the hub is running as a redudancy backup to another primary hub. See Application-Level Redundancy for more details.
redundancy.backup.primary.uri	Specfies the URI for the primary hub's configuration interface. This will be based on the primary's *configuration.scheme* and *configuraton.port* settings described above and its *host address*. For example, http://127.0.0.1:45245.
redundancy.backup.primary.pingIntervalSeconds	Specifies how often to ping the primary (in seconds) to ensure it is operational. The valid range is 1-3600. If this setting is not specified, a value of 10 will be used.
redundancy.backup.primary.pingAttempts	Specifies how many failed attempts should occur before promoting the backup to active. The valid range is 1-10. If this setting is not specified, a value of 1 will be used.
authentication.providers	Specifies the list of identity providers that the application will support. Identity providers will be specified using an array that contains field specific names. Note: The application only supports internal (pre-existing HighByte Intelligence Hub specific authentication, more information can be found in section 4.2) and saml2 login as of right now.
authentication.saml2	Specifies all SAML 2.0 specific settings. More information about supported SAML 2.0 settings can be found in the External Identity Provider Setup under Appendix A.

A. Save any changes and restart the runtime service for the updated application-level settings changes to take effect.

3.2.4 Project Configuration Backup

To back up the project configuration, use the configuration.backupCopies setting (see Application-Level Settings). This will backup existing configurations any time the project is changed, allowing you to recover an existing project in the event that a project is corrupted.

To recover an old project configuration, perform the following steps:

A. Stop the runtime.
B. Delete the intelligencehub-configuration.json from the application directory.
C. Copy the desired configuration.json file from the backup directory to the application directory and rename it to intelligencehub-configuration.json.
D. Start the runtime.

3.2.5 Application-Level Redundancy

To setup application-level redundancy, you will need perform the following steps:

A. Install two copies of the HighByte Intelligence Hub on separate machines. Identify one as the primary hub and the other as the backup hub.
B. Configure and start the primary hub
    a. There are no application-level settings that need to be set for the runtime identified as the primary hub.
    b. The primary hub should be started and will run with no knowledge of any backup hub.
C. Configure and start the backup hub
    a. Enable the backup hub and set the primary uri, pingIntervalSeconds and pingAttempts (see Application-Level Settings).
    b. Restart the backup hub for these changes to take effect.

With the primary and backup hubs running, application-level redundancy operates as follows:

A. The backup will start in standby mode. No flows will be started.
B. The backup will ping the primary at a rate set by pingIntervalSeconds. In the event the primary does not respond, the backup will continue to ping the primary until the number of successive failures is equal to pingAttempts.
C. Once failed pingAttempts is reached, the backup will transition to active mode and any flows that are enabled will be started.
D. In active mode, the backup will continue to ping the primary at a rate set by pingIntervalSeconds.
E. Once the backup successfully pings the primary, all flows will be stopped and the backup will transition to standby mode.
F. The process will repeat until the backup hub is stopped.

Note: There is no automatic project synchronization between the primary and backup hubs. The user will need to setup or deploy (through central hub management) a project for the primary and backup hubs. This can be the same project or different project in the event the backup should use alternate connections, etc.

4.0 Intelligence Hub Configuration

By default, the Intelligence Hub Configuration is self-hosted, meaning it doesn't require a 3rd party web server. By unzipping the default installer to disk, and launching the runtime, the configuration is available via the browser address http://localhost:45245.

It is possible to host the configuration separately via IIS, Tomcat, or any other webserver. To do this, please see the appendix for installation instructions.

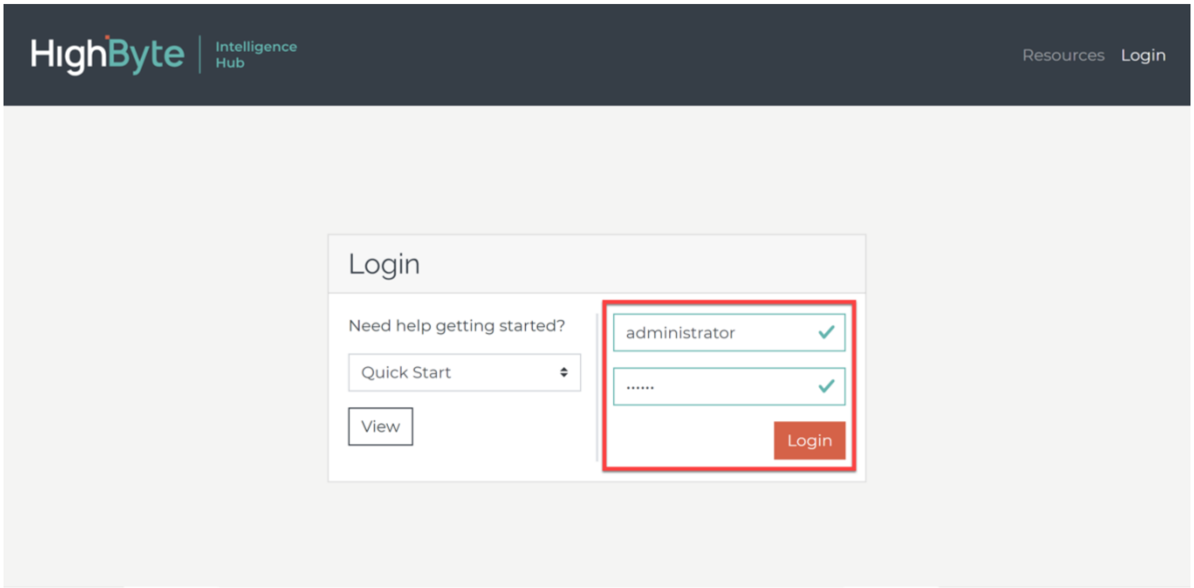
4.1 Administration

The usage instructions assume the runtime and configuration are installed and running per the instructions above.

4.1.1 Login

The runtime requires an administrative password to login and make configuration changes.

A. To log into the runtime and make configuration changes, type in the administrative *password* and click the *Login* button. By default, the administrative password is empty and should be set after the first login (see 4.2.2). See Figure 4.

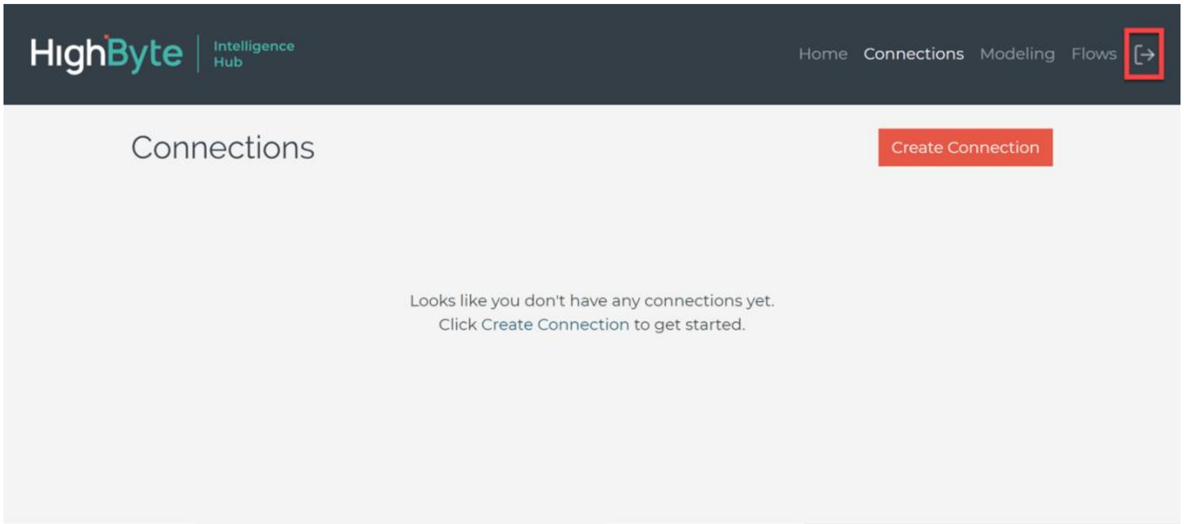


administrator

••••••

Figure 4

B. To logout, click the *Logout* icon. See Figure 5.



Looks like you don't have any connections yet.
Click Create Connection to get started.

Figure 5

4.1.2 Set Administrative Password

It is highly recommended that a strong administrative password be set after installation.

A. *Login* to the runtime with the current administrative password (default is empty).
B. Click *Admin* in the configuration's Main Menu, click the *Users* tab, and then click the *Select* the *Details* for the built-in administrator account. See Figure 6.

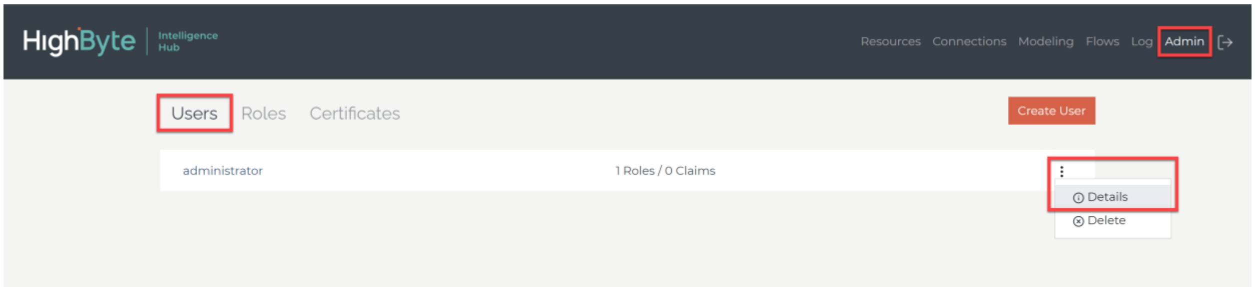
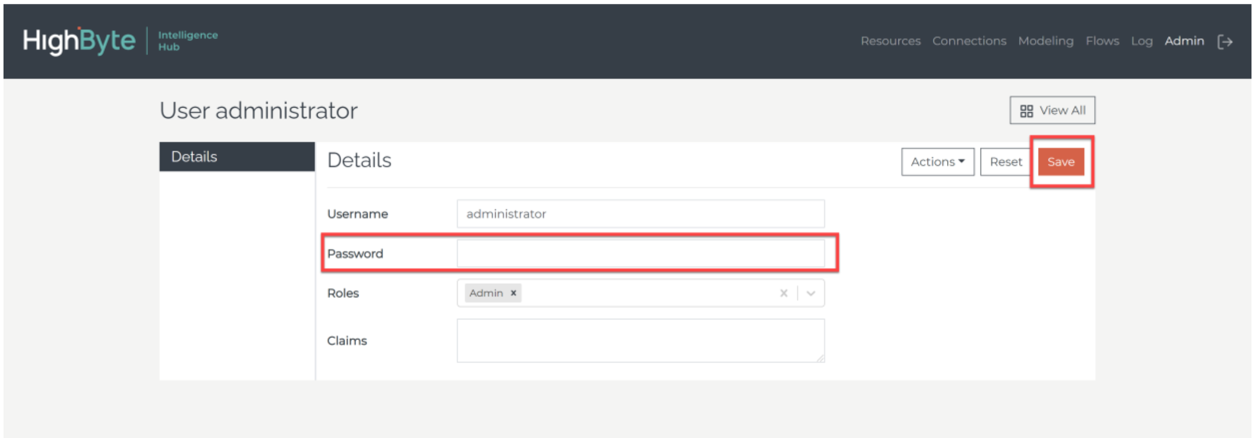


Figure 6

C. *Set* the new password and Click the *Save* button to commit the password. *Logout* and re-*Login* with the new credentials. See Figure 7.



Username: administrator

Password:

Roles: Admin ×

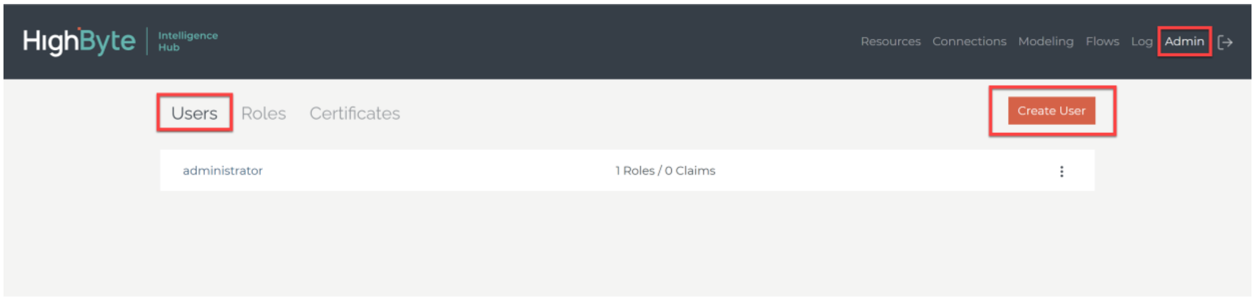
Claims:

Figure 7

4.1.3 Create a User

Users can be created and assigned certain roles and/or claims to allow them to perform certain types of operations.

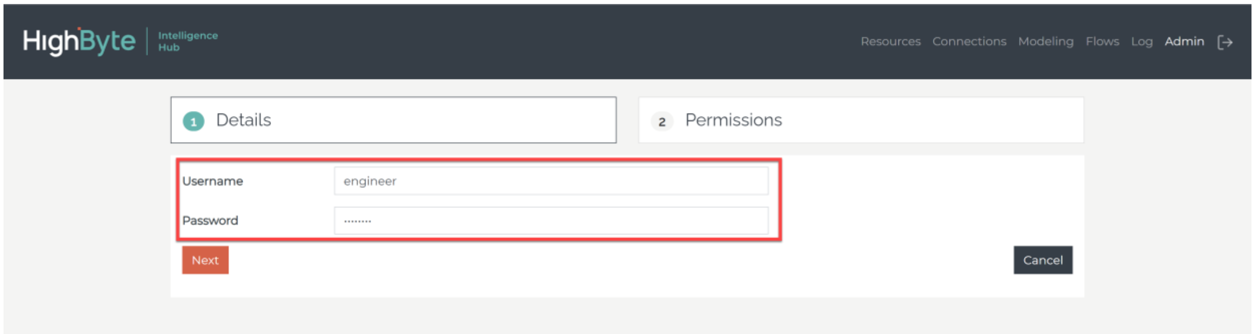
A. Click *Admin* in the configuration's Main Menu, click the *Users* tab, and then click the *Create User* button. See Figure 8.



administrator — 1 Roles / 0 Claims

Figure 8

B. Enter the *Username* and *Password*. *Click Next* to continue. See Figure 9.



Username: engineer

Password: ········

Figure 9

C. Select the *Roles* and/or set the *Claims* the user should inherit. Click *Submit* to add the user. See Figure 13. Claims take the form of resource:action, where resource is the object (e.g., connection) and action is the operation that can be performed in CRUD terminology (create, read, update, delete). Wildcard syntax like connection:* is also supported.

A list of *Claim resources* that can be set is as follows:

Resource	Description
connection	Connections, includes inputs and outputs
model	Models
instance	Instances
flow	Flows
network	Network hubs, groups, and sync operations
log	The event log
user	Users, roles, and claims
certificate	Certificates

Below are example claims.

Claim	Description
*	Admin claim that provides access to all resources and operations
connection:*	Run any operation on connections
*:read	Access to read all resources
connection:read	Read only connections
user:create	Create new users
user:update	Edit existing users
user:delete	Delete users
network:create	Create new network groups, sync objects between groups

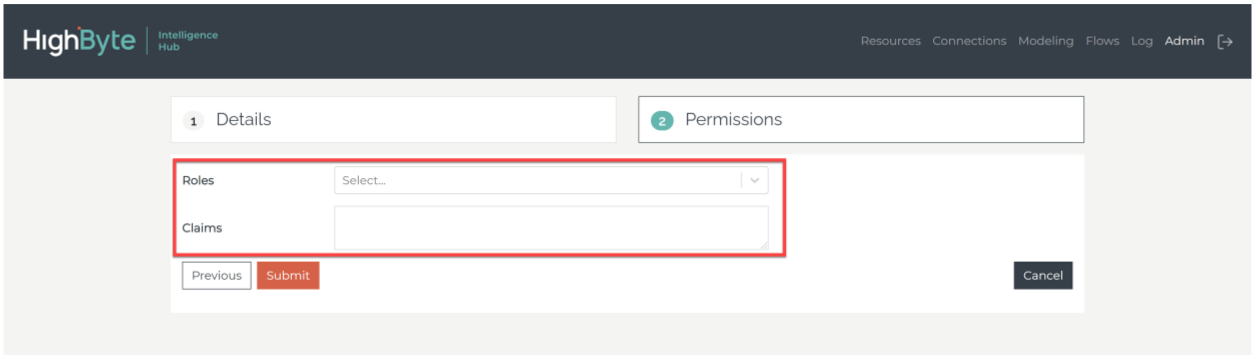
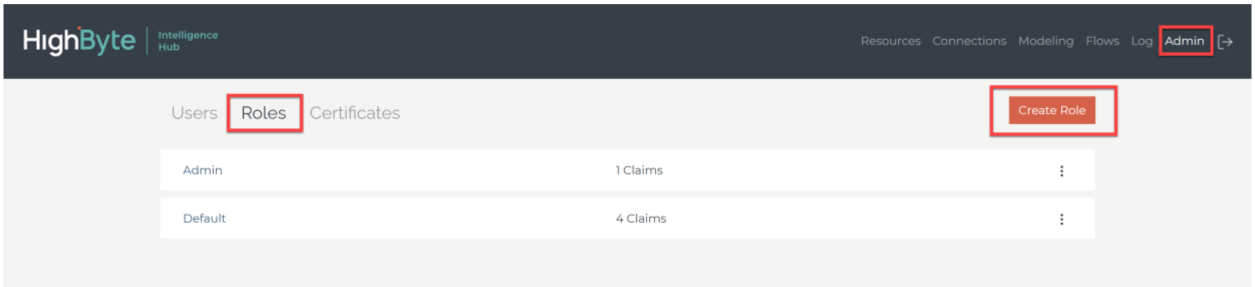


Figure 10

4.1.4 Create a Role

Roles can be created and assigned claims that can be inherited by one or more users. The roles allow the type of operations a class of users may perform.

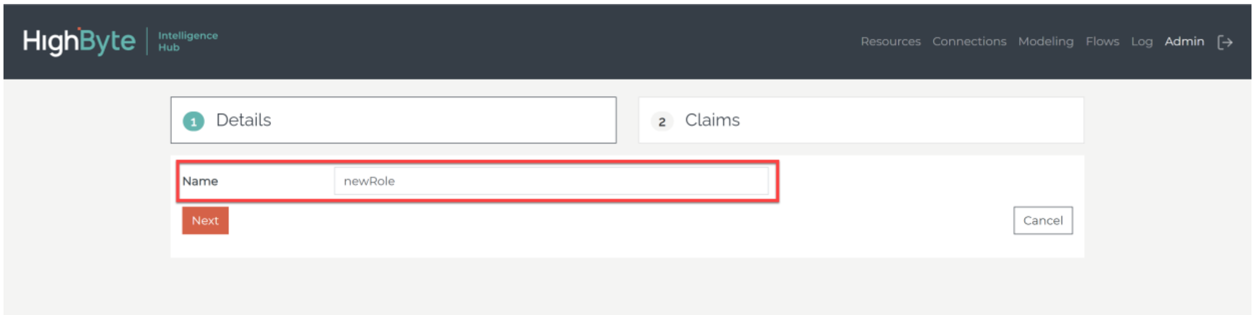
A. Click *Admin* in the configuration's Main Menu, click the *Users* tab, and then click the *Create Role* button. See Figure 11.



Admin	1 Claims
Default	4 Claims

Figure 11

B. Enter a *Name* to represent the new role. See Figure 12.



Name: newRole

Figure 12

C. Set the *Claims* the user assigned to this role should inherit. Click *Submit* to add the role. See Figure 13.

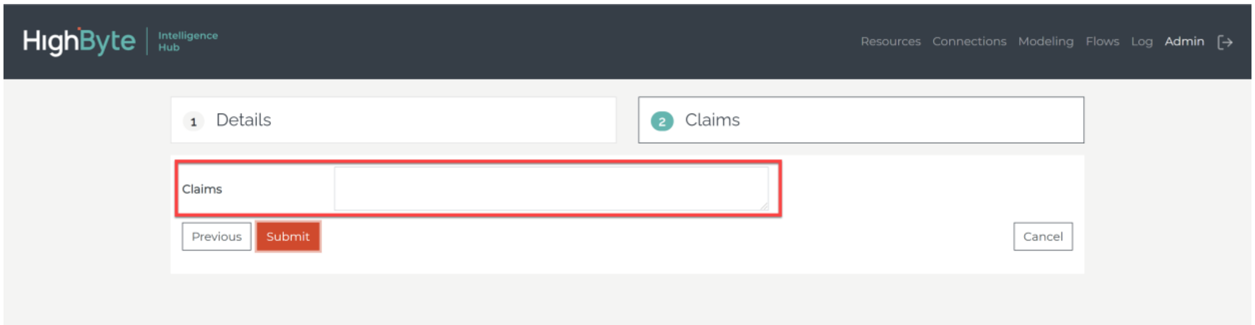


Figure 13

4.1.5 Create a Certificate

Certificates are commonly used to secure communications and authenticate clients. An example is using certificates with the MQTT connector to send and receive data from AWS IoT Core or AWS IoT Greengrass. In this case, the self-signed certificate for AWS is used to secure the connection and AWS provides a public and private key to authenticate the hub. Use the steps in the preceding sections to import certificates into the hub and use them in connectors.

A. Click *Admin* in the configuration's Main Menu, click the *Certificates* tab, and then click the *Create Role* button. See Figure 14.

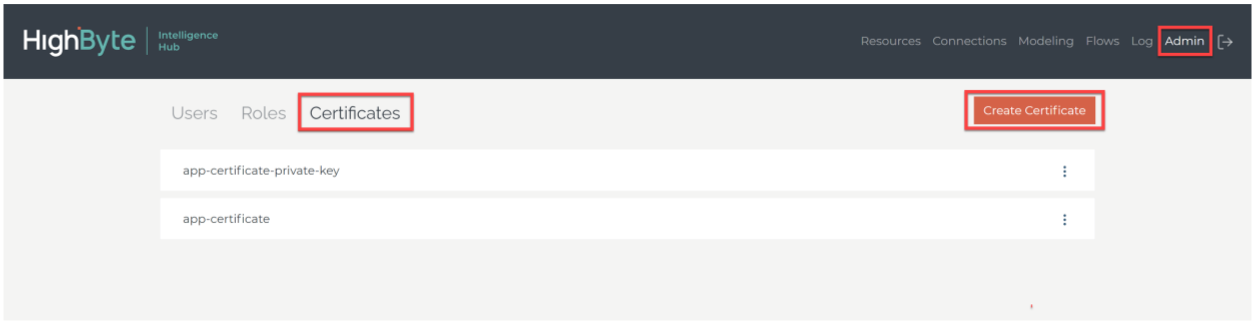
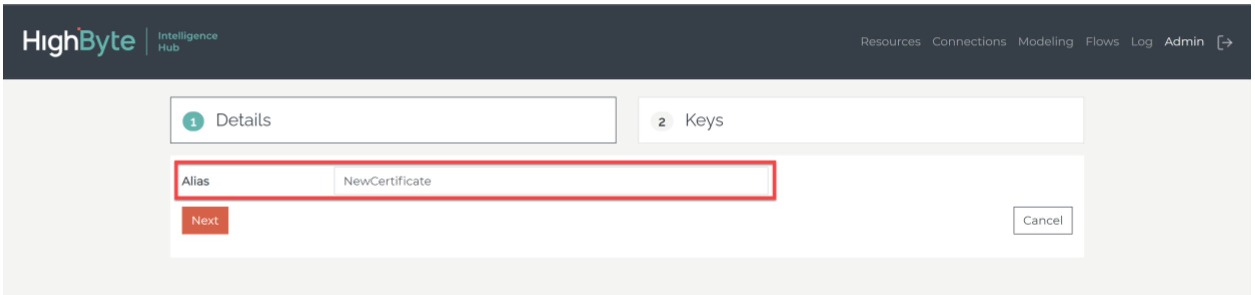

HighByte Intelligence Hub
Resources Connections Modeling Flows Log Admin
Users Roles Certificates
Create Certificate
app-certificate-private-key
app-certificate

Figure 14

HighByte Intelligence Hub may automatically generate some certificates/keys on your behalf (e.g., app-certificate* are application instance specific certificates leveraged by OPC UA connections). You may set these named certificates to your own public/private keys based on your company's internal IT policies.

B. Enter an *Alias* to represent the new certificate. See Figure 15.


HighByte Intelligence Hub
Resources Connections Modeling Flows Log Admin
1 Details
2 Keys
Next
Cancel

Alias: NewCertificate

Figure 15

C. Insert the textual representation of the *Public* and/or *Private* keys. See Figure 16.

Use this field to import public certificates, often used to secure TLS/SSL and HTTPS connections. Open the certificate file in a text editor and copy all of the text into this field. An example Public Key will be decorated with text that begins with -----BEGIN CERTIFICATE----- and ends with -----END CERTIFICATE-----. Copy all of the text (including the BEGIN and END parts) into the field.

Private Key is used in addition to Public Key to import public/private key pairs into the hub. To do this, open the private key file in a text editor and copy all of its contents into the Private Key field. It should start with -----BEGIN RSA PRIVATE KEY----- and end with -----END RSA PRIVATE KEY-----. Also, make sure you've included the public part of the key in Public Key.

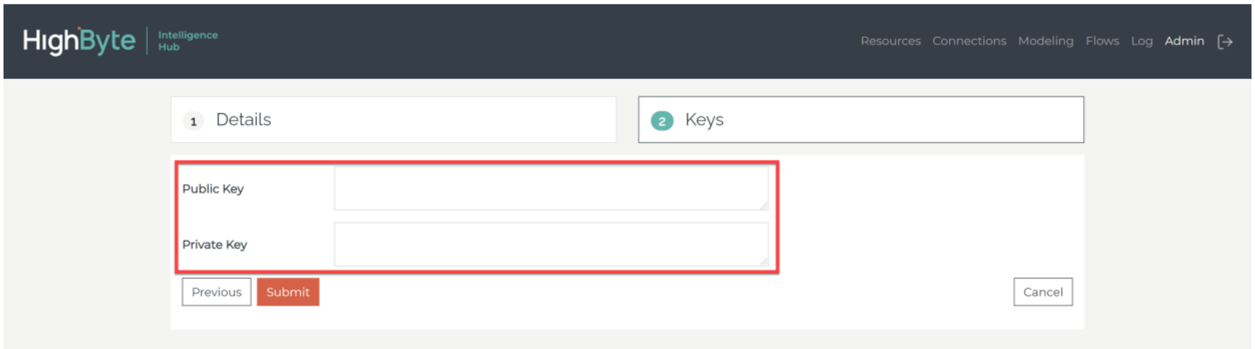

HighByte | Intelligence Hub
Resources Connections Modeling Flows Log Admin
1 Details
2 Keys
Public Key
Private Key
Previous
Submit
Cancel

Figure 16

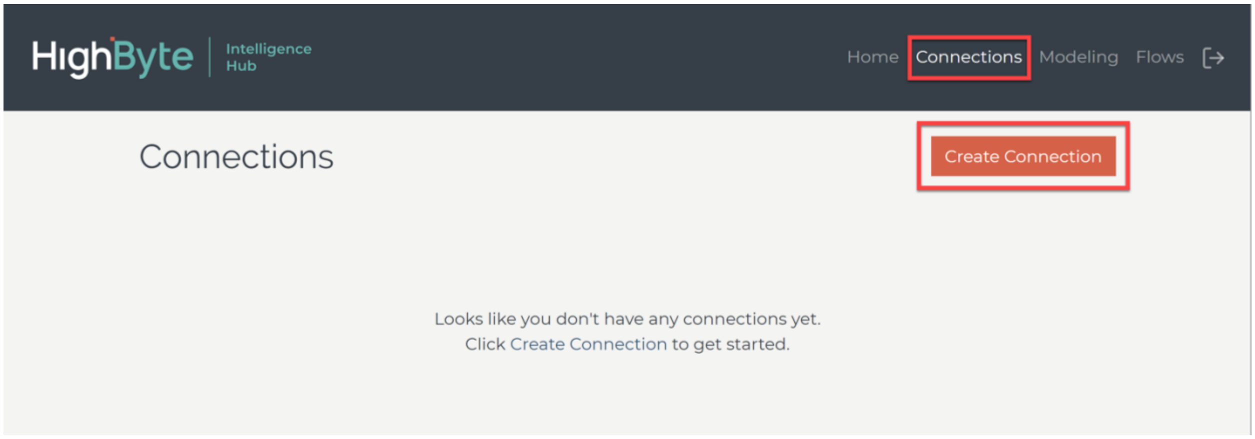
4.2 Project Configuration

The usage instructions assume the runtime and configuration are installed and running per the instructions above.

4.2.1 Create a Connection

A connection represents a path to a source that contains inputs that can be read or outputs that can be written to.

A. Click *Connections* in the configuration's Main Menu and then Click *Create Connection* to get started. See Figure 17.



Connections

Looks like you don't have any connections yet.
Click Create Connection to get started.

Figure 17

B. Enter a *Name* to represent the connection. Names can only contain alphanumeric and underscore characters (e.g., A-Z, a-z, 0-9 or _). Optionally enter in a *Description* for the connection. Click *Next* to continue. See Figure 18.

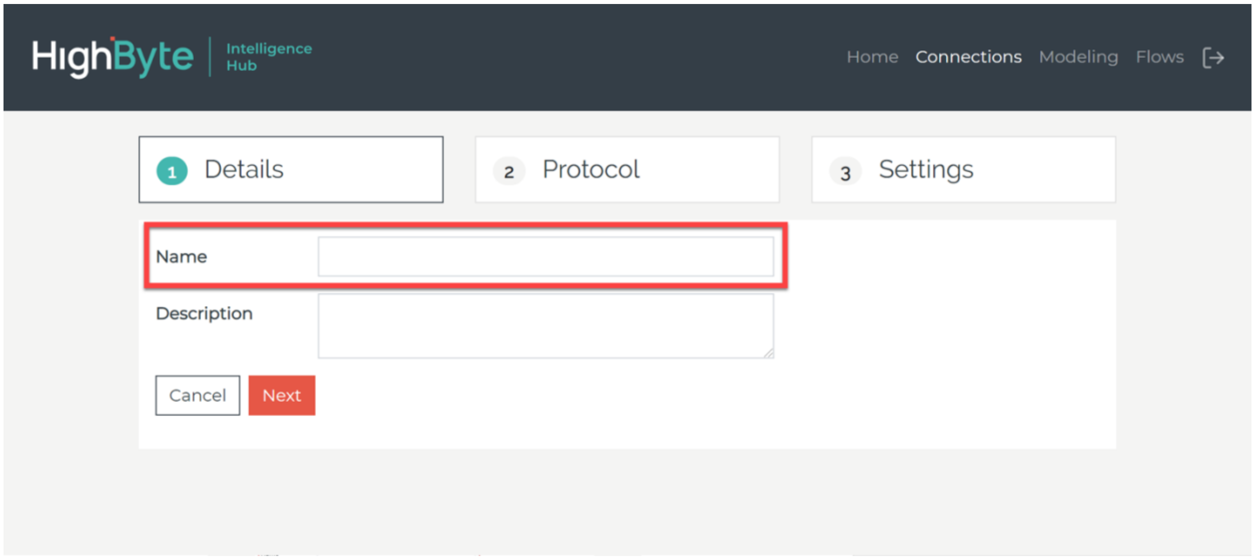


Figure 18

C. Select the *Protocol*. Depending on the protocol, set the *Hostname*, *Port* and optional *Path* to set the desired *URI* for the connection Click *Next* to continue. See Figure 19.

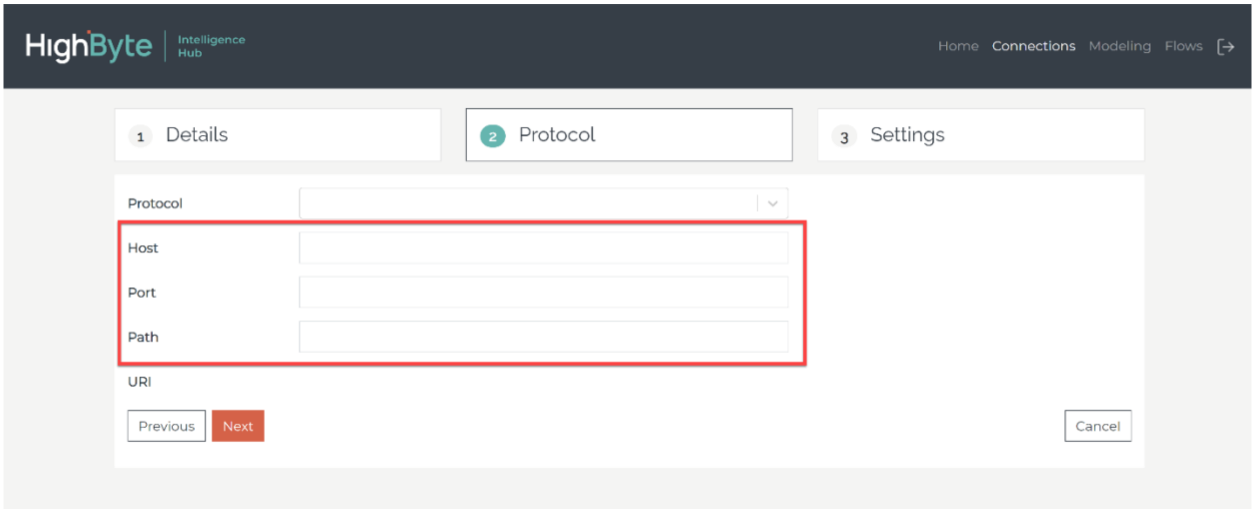


Figure 19

D. Set any Connection *General* and *Protocol Specific Settings*. Click *Submit* to add the connection and return to the main Connection page. General and Protocol Specific Settings are described below.

General Settings

Store and Forward	Description
Enabled	Specifies whether store and forward is enabled for the connection.
Max Entries Per Output	Maximum number of store and forward entries per output in the event writing to a connection's output fails. Once the maximum is reached, any new output data is discarded and a message is logged. Upon re-establishing a connection, stored outputs are written in the same order they were entered into the store (FIFO). An output's write entry is only removed from the store once it has been successfully acknowledged by the destination application. All stored output writes are written as fast as possible to ensure the delivery of newer data is not stalled.
Failure Wait Interval	How long to wait before retrying to write the next batch of stored entries when a connection has failed. The interval can be specified in milliseconds, seconds, minutes, hours, or days. Valid range is between 10 milliseconds and 36 days. Once the connection is re-established, writes are handled as described above.

Protocol Specific Settings

AWS IoT SiteWise Settings	Description
Access Key	IAM created user access key with AWSIoTSiteWiseFullAccess permission.
Secret Key	IAM provided secret key.
Region	Region of the AWS IoT SiteWise instance (e.g., us-east-1)
Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to AWS.
Amazon Kinesis Data Streams Settings	
Access Key	IAM created user access key with AmazonKinesisFullAccess permission.
Secret Key	IAM provided secret key.
Region	Region of the Amazon Kinesis Data Streams instance (e.g., us-east-1)
Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to Amazon.
Amazon Kinesis Data Firehose Settings	
Access Key	IAM created user access key with AmazonKinesisFirehoseFullAccess permission.
Secret Key	IAM provided secret key.
Region	Region of the Amazon Kinesis Data Firehose instance (e.g., us-east-1)
Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to Amazon.
Apache Parquet Settings	
File Directory	Specifies the directory where files should be sourced for processing.
Processed File Directory	Specifies the directory where files should be moved to after processing successfully. This is only used for inputs set to Indexed.
Error Directory	Specifies the directory where files should be moved to after processing unsuccessfully. This is only used for inputs set to Indexed.
Azure Event Hubs Settings	
Connection String	The primary connection string for the Event Hubs Namespace. Found under Share access policies in the Azure Console's Event Hubs view.
Event Hub Name	The name of the event hub to publish to.
Request Timeout	Maximum time to wait for a send request to respond before failing. Valid range is 100–60000 milliseconds.

Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to MQTT as a JSON object.
Azure IoT Hub Settings	
Connection String	The primary connection string of the IoT Hub’s IoT Device, found under IoT devices in Azure Console’s IoT Hub configuration.
Protocol	Specifies the underlying protocol to use for connectivity. Options include AMQPS, HTTPS and MQTT.
Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to MQTT as a JSON object.
CSV Settings	
File Directory	Specifies the directory where files should be sourced for processing.
Processed File Directory	Specifies the directory where files should be moved to after processing successfully. This is only used for inputs set to Indexed.
Error Directory	Specifies the directory where files should be moved to after processing unsuccessfully. This is only used for inputs set to Indexed.
Google Cloud Pub/Sub Settings	
Project Id	The Project ID as defined in the Google Cloud Console.
Service Account JSON	The Google Console Service Account Key JSON file, used to grant access to the Service Account that has permission to publish to the Project and Topic. Paste the contents of the key .json file directly in this field. As an example, the file should start with "type": "service_account", "project_id": "myproject-312317", "private_key_id": "12345...",
InfluxDB Settings	
URL	Specifies the URL to connect to. This is the base URL to the InfluxDB API.
Token	Specifies the token provided by InfluxDB to authenticate the connection. This token limits what HighByte Intelligence Hub has access to in InfluxDB.
Organization Name	The organization name in Influx. This is optional, and when left blank the default organization for the account is used.
JDBC Settings	
JDBC Connection String	The full JDBC connection string required by the driver. Please consult your driver documentation.
Class Path	Class path of the JDBC driver. Please consult your driver documentation.
Microsoft SQL Server Settings	
Database	Name of the database to connect to.
Username	Username for authentication with the database.
Password	Password for authentication with the database.
Additional JDBC Options	Additional JDBC options to include in the JDBC connection string. This is MSSQL specific. Options are entered exactly as they would be if the JDBC connection string was created manually.
Flatten Modeled Values	Enable this setting to log models with hierarchy to a SQL table. As an example, if ModelA contained ModelB, this option would flatten the names of ModelB attributes to ModelA_ModelB_Attribute.
MQTT Settings	
Client ID	The ID to send to identify the connection with the broker.
Username and Password	The username and password required by the MQTT broker.
Connection Timeout	Maximum time to wait for a connection to respond. Valid range is 1–300 seconds.
Keep Alive	Maximum time to wait without sending a request to the broker. Valid range is 10–43200 seconds.
Use SSL	Enable for MQTT brokers that require transport layer security. See *Appendix A* for more information on MQTT security.

	CA Certificate: The name of the certificate file (e.g., groupCA.pem) imported into the certificate store. Client Certificate: Used if the broker requires client authentication. This is the public certificate for the client (e.g., mycert.pem). Client Key: Used if the broker requires client authentication. This is the private key for the client (e.g., Mycert.key.pem).

Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to MQTT as a JSON object.
	Model Value (JSON)
	{
	“Machine” : {
	“AssetInfo” : {
	“ID” : “MACH-L1-102”,
	“SN” : “B320-3R821N9-D”,
	“Date” : “2018-09-29”
	},
	“Online” : true,
	“Utilization” : 40.0
	}
	}
	Flattened Modeled Value (JSON)
	{
	“Machine” : {
	“AssetInfo_ID” : “MACH-L1-102”,
	“AssetInfo_SN” : “B320-3R821N9-D”,
	“AssetInfo_Date” : “2018-09-29”,
	“Online” : true,
	“Utilization” : 40.0
	}
	}

MySQL Settings	
Database	Name of the database to connect to.
Username	Username for authentication with the database.
Password	Password for authentication with the database.
Additional JDBC Options	Additional JDBC options to include in the JDBC connection string. This is MySQL specific. Options are entered exactly as they would be if the JDBC connection string was created manually.
Flatten Modeled Values	Enable this setting to log models with hierarchy to a SQL table. As an example, if ModelA contained ModelB, this option would flatten the names of ModelB attributes to ModelA_ModelB_Attribute.
OPC UA Settings	
Security	Level of security required by the OPC UA Sever. Valid options are None, Basic256Sha256-Sign and Basic256Sha256-SignEncrypt. See *Appendix A* for more information on OPC UA security.
Authentication Type	Level of authentication required by the OPC UA Server. Valid options are Anonymous and Username-Basic256 (requires a username and password to be set).
Connection Timeout	Maximum time to wait for a connection to be established. Valid range is 1–60 seconds.
Request Timeout	Maximum time to wait for a connection to respond. Valid range is 100–60000 milliseconds.
Flatten Modeled Values	*Not Applicable For OPC UA Connections*
Mode	Specifies how the data collection (reading input data) is performed with the OPC Server. Valid options are Poll and Subscribe. When Poll is set, reads will be performed on-demand. This mode ensures the most current data is obtained based on time or flow events, but may cause stress and the inability for the OPC UA server to optimize communications. When Subscribe is set, the OPC UA server will determine how best to collect the data and make it available when data changes.
Subscription Rate	Specifies the default rate that connection’s inputs should be monitored (read) for change. An input can override this setting by setting its sampling rate. This setting only applies when the mode is set to Subscribe.
OSIsoft PI AF SDK Settings	
Token	The connection token used by the OSIsoft AF SDK Agent to authenticate the HighByte Intelligence Hub connection. See the Appendix for details on how to setup the OSIsoft AF SDK Agent, which is required to connect to PI.
PostgreSQL Settings	
Database	Name of the database to connect to.
Username	Username for authentication with the database.

Password	Password for authentication with the database.
Additional JDBC Options	Additional JDBC options to include in the JDBC connection string. This is PostgreSQL specific. Options are entered exactly as they would be if the JDBC connection string was created manually.
Flatten Modeled Values	Enable this setting to log models with hierarchy to a SQL table. As an example, if ModelA contained ModelB, this option would flatten the names of ModelB attributes to ModelA_ModelB_Attribute.
REST Client Settings	
Base URL	The base URL for the request (e.g., https://myurl.com)
Authentication Type	Authentication type used for REST client connection. None: No authentication used. Basic Auth: Set the username and password for authenticating REST client calls. OAUTH 2.0: Set the login Grant Type, Login URL, Scope, Audiences, and Resources for authentication. The only Grant Type currently supported is client credentials. With client credentials, users provide a Username and Password.
Header	Name=Value header pairs separated by &. (e.g., Content-Type=application/json&key=123). Passed on each request. If a value contains a ‘&’ or ‘=’ character, you must URL encode the value (e.g., = becomes %3D).
Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to a REST output.
Sparkplug Settings	
Client ID	The ID to send to identify the connection with the broker.
Username and Password	The username and password required by the MQTT broker.
Connection Timeout	Maximum time to wait for a connection to respond. Valid range is 1–300 seconds.
Keep Alive	Maximum time to wait without sending a request to the broker. Valid range is 10–43200 seconds.
Request Timeout (ms)	Maximum time to wait for a subscription to arrive. Valid range is 100–60000 milliseconds.
Group Id	The ID that provides a logical grouping of Edge of Network (EoN) nodes.
Edge Node Id	The ID that uniquely identifiers an EoN node.
Use SSL	Enable for MQTT brokers that require transport layer security. See *Appendix A* for more information on MQTT security. CA Certificate: The name of the certificate file (e.g., groupCA.pem) imported into the certificate store. Client Certificate: Used if the broker requires client authentication. This is the public certificate for the client (e.g., mycert.pem). Client Key: Used if the broker requires client authentication. This is the private key for the client (e.g., Mycert.key.pem).
Flatten Modeled Values	Specifies if modeled values should be flattened before publishing to Sparkplug as a list of metrics. **Model Value (Metrics)** “Machine\AssetInfo\ID” : “MACH-L1-102” “Machine\AssetInfo\SN” : “B320-3R82-D” “Machine\AssetInfo\Date” : “2018-09-29” “Machine\Online” : true, “Machine\Utilization” : 40.0 **Flattened Modeled Value (Metrics)** “Machine\AssetInfo_ID” : “MACH-L1-102” “Machine\AssetInfo_SN” : “B320-3R82-D” “Machine\AssetInfo_Date” : “2018-09-29” “Machine\Online” : true, “Machine\Utilization” : 40.0
Webhook Settings	
No protocol specific settings	

4.2.2 Create an Input

An input represents a path to a data point contained in a connection that can be read.

A. Under *Connections*, find the *Connection*, click the drop-down menu, and click the *Inputs* menu item. See Figure 20.

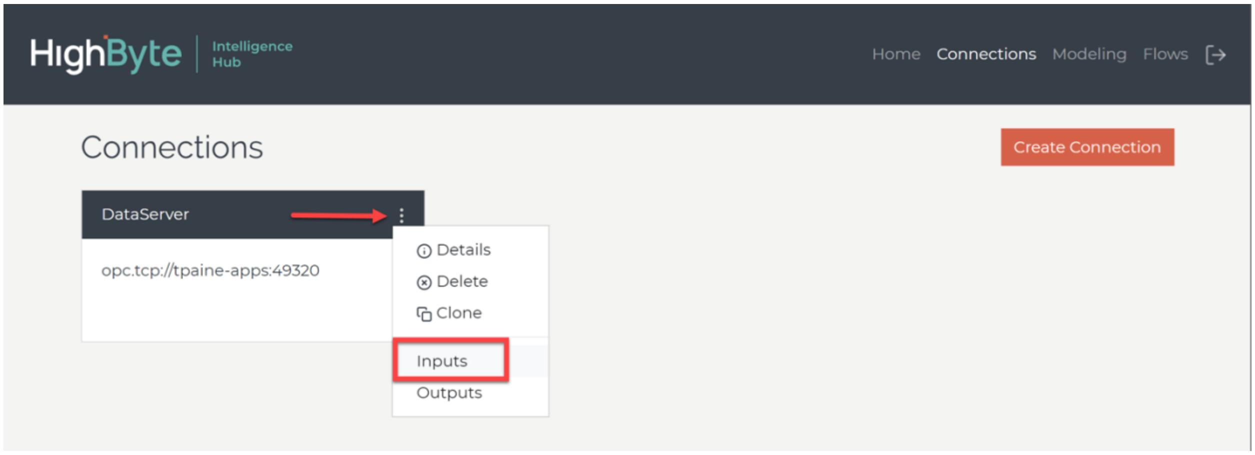
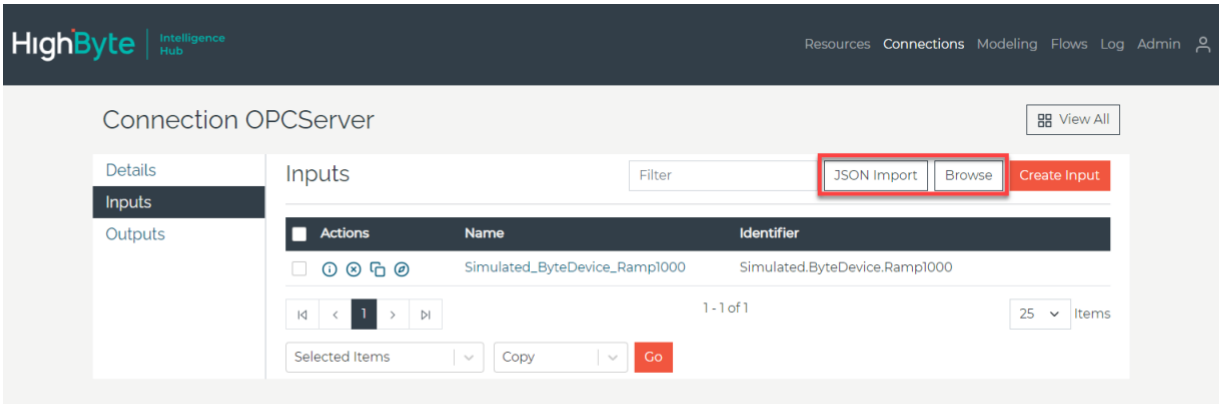


Figure 20

B. Depending on the type of connection you may be able to *Browse* for inputs or *Import* one or more inputs in JSON format. See Figure 21.



Actions	Name	Identifier
□	Simulated_ByteDevice_Ramp1000	Simulated.ByteDevice.Ramp1000

Figure 21

Click the *Browse* button to *View* and *Select* the desired inputs. When finished click the *Import Selected* button. See Figure 22.

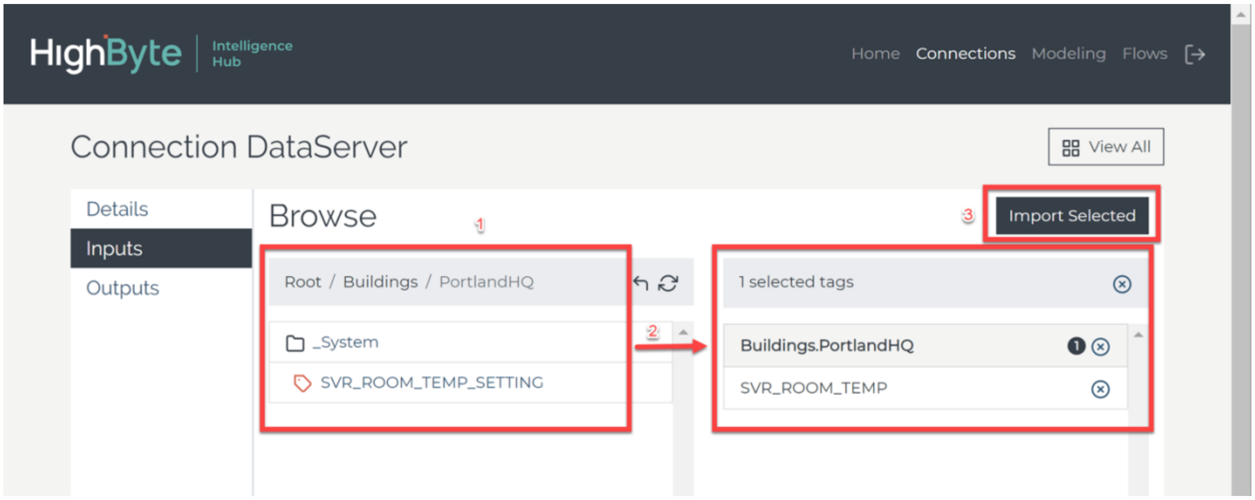
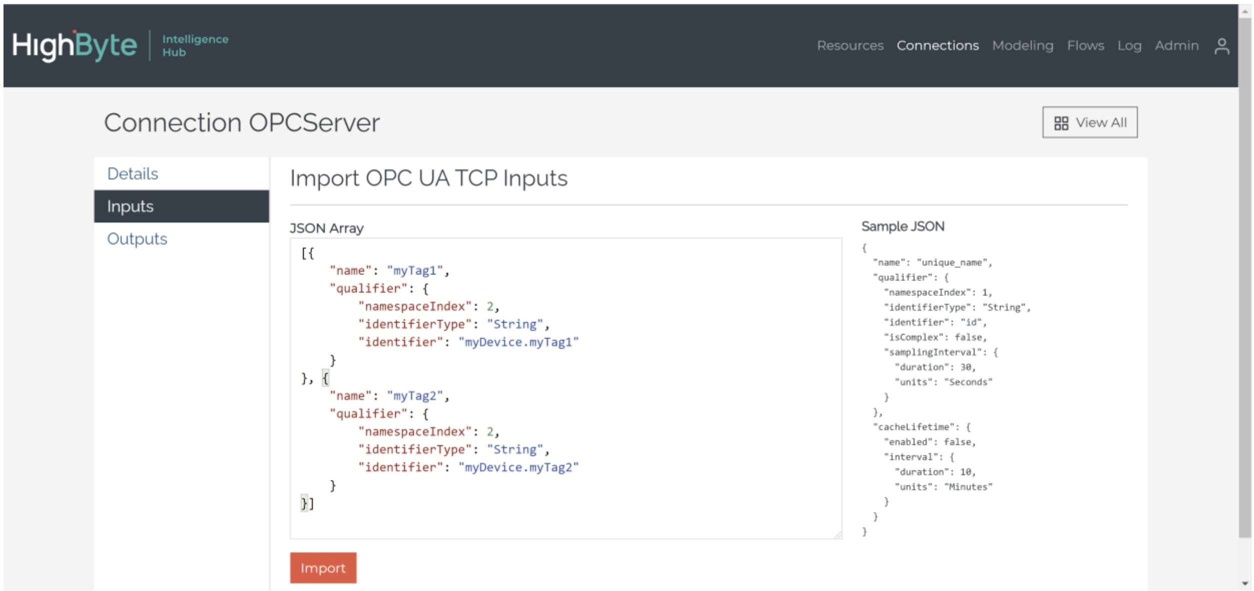


Figure 22

Click the *JSON Import* button to manually create one or more inputs using JSON. This is currently only supported for OPC UA. A sample is provided to the right of the import field. When finished click the *Import* button to add the inputs. See Figure 23.

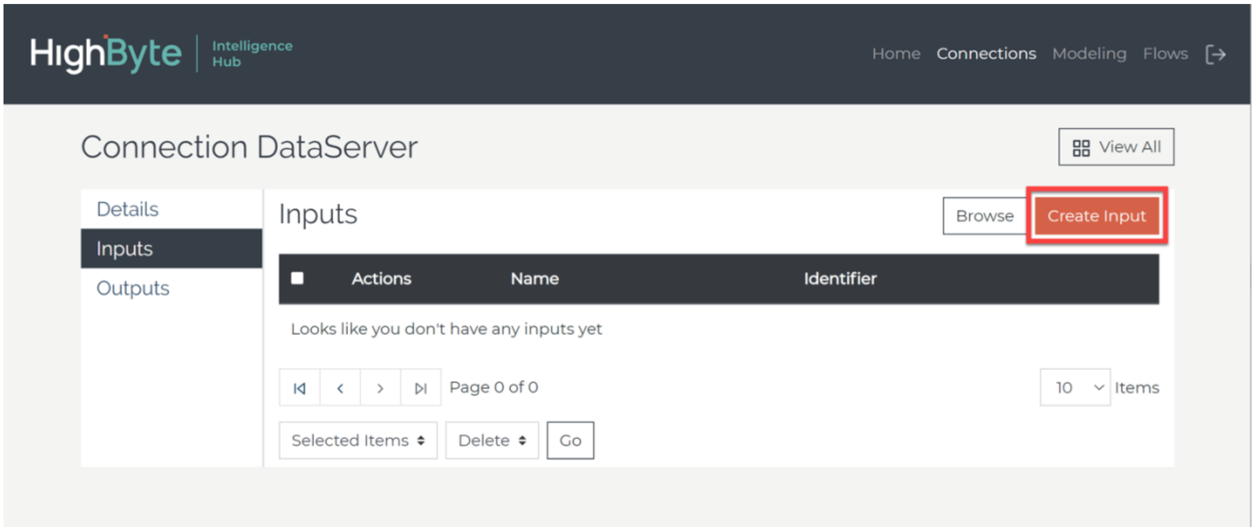


```
[{
    "name": "myTag1",
    "qualifier": {
        "namespaceIndex": 2,
        "identifierType": "String",
        "identifier": "myDevice.myTag1"
    }
}, {
    "name": "myTag2",
    "qualifier": {
        "namespaceIndex": 2,
        "identifierType": "String",
        "identifier": "myDevice.myTag2"
    }
}]
```

```
{
  "name": "unique_name",
  "qualifier": {
    "namespaceIndex": 1,
    "identifierType": "String",
    "identifier": "id",
    "isComplex": false,
    "samplingInterval": {
      "duration": 30,
      "units": "Seconds"
    }
  },
  "cacheLifetime": {
    "enabled": false,
    "interval": {
      "duration": 10,
      "units": "Minutes"
    }
  }
}
```

Figure 23

C. Alternatively, you can click the *Create Input* to manually add an input. See Figure 24.



Connection DataServer

Inputs

■	Actions	Name	Identifier

Looks like you don't have any inputs yet

Page 0 of 0

Figure 24

Set any *General* and *Protocol Specific Settings*. Click *Submit* to add the input and return to the input list. General and Protocol Specific Settings are described below.

General Settings

General	Description
Name	Specifies the name of the input. The name must be unique across all inputs for the specified connection. Names can only contain alphanumeric and underscore characters (e.g., A-Z, a-z, 0-9 or _).
Cache	When enabled, specifies how long the input's value should be cached and reused internally after a successful read from the underlying connection. Once the cached value's lifetime expires, the cache is invalidated and a new read is issued to the underlying connection. The lifetime interval can be specified in milliseconds, seconds, minutes, hours, or days. Valid range is between 10 milliseconds and 36 days. *Note: The cached value and its lifetime is persisted to disk and will be used across product runs until the value expires.*

Protocol Specific Settings

AWS IoT SiteWise Settings	Description
Inputs are not currently supported	
Amazon Kinesis Data Streams Settings	
Inputs are not currently supported	
Amazon Kinesis Data Firehose Settings	
Inputs are not currently supported	
Azure Event Hubs Settings	
Inputs are not currently supported	
Azure IoT Hub Settings	
Inputs are not currently supported	

CSV Settings	
CSV File Name	The file name to read. This can be the full name of a single file or a regular expression. As an example, (.*).csv would match the first .csv file in the directory based on alphabetical order.
Skip Lines	How many lines to skip from the front of the file. Use this setting to skip header lines or other data that is not required.
Max Rows	The max number of rows in the file. Leave this blank if you just want to read to the end of the file. Set this if you only want to read up to N rows.
Max Rows Per Read	The max number of rows to return from a single read. As an example, if there are 1000 rows and this is set to 10, only 10 rows would be read at a time. When Index File is on, the first read will be records 0 to 9, second read will be 10 to 19, etc.
Replacement Headers	Delimited separated list of headers used to replace the file headers, or provide headers (e.g., header1, header2, header3). *Note: There must be a header for each column of data.*
Delimiter	Specifies the delimiter used to separate headers and fields. The default delimiter is a comma. Only single character delimiters are supported.
Index File	When enabled, the file is moved to the connection’s processed directory after being read.
Google Cloud Pub/Sub Settings	
Inputs are not currently supported	
InfluxDB Settings	
Flux Query	The Flux query string used to query InfluxDB. See the InfluxDB documentation on supported query strings and syntax. https://docs.influxdata.com/influxdb/v1.8/guides/query_data/ Data is returned in CSV format, and this data is available in HighByte Intelligence Hub as an array of rows.
MQTT Settings	
Topic	Specifies a string that is used to subscribed to specific MQTT messages.

Include Topic	Specifies if the MQTT topic should be included in the read data. When enabled the reads look as follows, where the payload is inserted under value, and the topic is included at the root.
	{
	"_topic": "deviceid",
	"value": {
	“deviceid": "press1",
	"other": 10
	}
	}

OPC UA Settings	
Namespace Index	Numeric index of the namespace the identifier belongs to.
Identifier Type	Specifies the type of data used in the identifier field. This can be set to String or Numeric. *This is not the data type of the value associated with this OPC UA input.*
Identifier	Specifies the identifier used to reference the OPC UA input.
Sampling Interval	Specifies the rate that the input should be monitored (read) for change. By default, this value is zero, which uses the subscription rate set on the connection. Set this value to something slower than the connection’s subscription rate to sample less frequently (down sample). Setting this value to something faster than the connection’s subscription rate will be clamped to the subscription rate (over sampling is not supported).
OSIsoft PI AF SDK Settings	

Type	Specifies the type of data to read. Options include assets, event frames, and points.
Database	Specifies the PI AF database to query. This field is required for asset and event frame reads.
Query	Used for Event Frames and Assets. For Assets, this is the Asset search syntax supported by PI. Examples are provided below. Get the Boiler1 under PlantB\Boilers *Parent:PlantB\Boilers Name:Boiler1* Get all elements with a template type of Boiler *Template:Boiler* See the OSIsoft documentation or PI Explorer for supported syntax. For Event Frames, this represents the query syntax supported by PI Explorer. Get all event frames * Get all event frames that start with the name boiler name:Boiler*
Points	Used for point reads only. A list of all the points to read.
Get	Used for point reads only. Specifies the type of data to get for the point. Options include current value, raw values, interpolated, and summary queries.
Start Time	Used for point reads only. Specifies the start time for the query. . This uses the standard PI time syntax. Relative and absolute times are supported (ex. *-1h). Absolute times must include a “ Z” at the end to be treated as UTC.
End Time	Used for point reads only. Specifies the end time for the query. The supported formats are the same as Start Time.
Boundary Type	Used for point reads of raw values only. Options include inside, outside, or interpolated.
Interval	Used for point reads of type summary or interpolated. Specifies the interval to interpret data samples. For example, 5m, means every 5 minutes. Use this setting to break reads into chunks. As an example, get the average value of a point every 5 minutes over a 1-hour period When left blank the interval defaults to the time range between Start and End Time.
Calculation Basis	Used for point reads of summary types only (ex. average). Options include time weighted or event weighted.
Index	Used for point reads only. When enabled, reads are performed from the Start to the End Time, in window intervals. After each successful read, the last read index is updated and stored to disk, so that in the case of a system restart, the reads will start from where they last ended. Indexing is useful when querying a large amount of historical data.
Index Window	Used when indexing is enabled. This is the window size to read on each read request. As an example, with the following settings each read will gather one days' worth of data in 5 minute intervals and reads will stop after the full 100 days is read. Start Time = *-100d End Time = * Interval = 5m Index = true Window Size = 1d
Apache Parquet Settings	

Parquet File Name	The file name to read. This can be the full name of a single file or a regular expression. As an example, (.*).parquet would match the first .csv file in the directory based on alphabetical order.
Max Rows Per Read	The max number of rows in the file. Leave this blank if you just want to read to the end of the file. Set this if you only want to read up to N rows.
Index File	When enabled, the file is moved to the connection's processed directory after being read.
SQL (JDBC, MSSQL, MySQL, PostgreSQL) Settings	
Query	The full SQL query, which can include any SQL syntax including stored procedures. Additionally, the query can reference any connections' inputs' values by including the qualified *@ConnectionName$InputName* (e.g., SELECT * FROM AssetTable WHERE [id] = @OPC$LastAssetId). Click the *Object Explorer* button to validate the connection settings and discover Tables and Views to perform a query on. Click the *Execute* button to validate the query. A subset of the results will be displayed.
Index	This setting is used to index the SQL query and only get new data, either by time or by an index, on each query. When enabled, specifies an index *Name* and default *Value* to reference within a query. This name / value pair is cached across successive product runs and allows the next query to pick up where it left off. Note the *Name* must be the column name in the table that is being used for indexing. An example query might be *SELECT * FROM SimpleMachineX WHERE [id] > :idIndex* By enabling indexing and specifying the *Name* as *idIndex* and *Value* as *zero*, the first time the query is run *:idIndex* is set to *zero* and after the query *:idIndex* is updated to the latest value from *[id]*. The latest value is cached to disk. Successive queries will use the latest value cached and then update the cached value on success. To reset the index, change the index *Value* and save the input. The next query will start with this *Value*. Note that test reads on the input done from the UI will use the current index *Value* but will not update the index *Value*.
REST Client Settings	
Endpoint URL	The endpoint to send the request to (e.g., /routes/myendpoint). Note the URL can reference data from other inputs using the @connection$input.attribute syntax. Other inputs are read first and the string value of the returned field is inserted into the URL before reading this input. As an example /my/url/@connection$input.attribute will result a call to /my/url/123 at runtime. Use the References panel on the left to explore and drag-and-drop references to other inputs.
Content Type	The content type of the incoming request: JSON or XML.
Method	The method to use: GET or POST.
Header	Name=Value header pairs separated by & (e.g., Content-Type=application/json&key=123). Passed on each request. These are added to the connection level header. This field supports dynamic inputs.

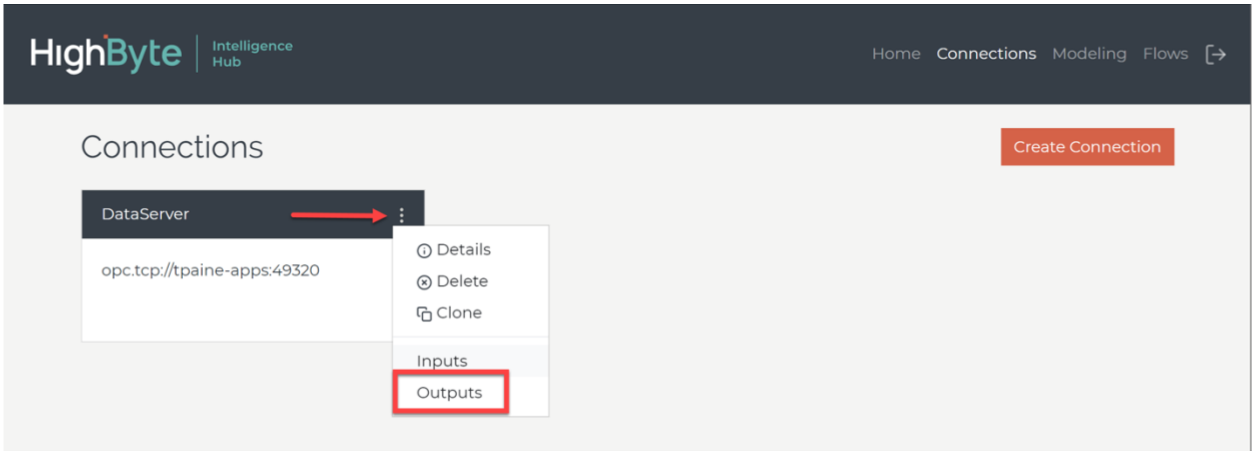
Body	The body of the request. Used with the POST method only. This can be any text including JSON. Like the Endpoint URL, this field can also referene data from other inputs using the @connection$input.attribute syntax.
Sparkplug Settings	
Device Id	The id for the attached device the metric belongs to. Do not set the device id if the metric belongs to the EoN.
Metric Name	The metric name(s) to read. Examples include: MetricA, Folder/MetricA, #, #/AssetInfo/# *Note: # is an MQTT wildcard character. When used, multiple metrics will be read and attached to a single input. This input will be treated as a complex data object that can be used as an input into a flow or attached to a modeled instance's attribute of type Any.*
Webhook Settings	

Filter	Filters incoming payloads using user defined syntax. Filter syntax is written in the form of filter1=value1&&filter2=value2. Each filter is separated by &&.
	{
	“Machine” : {
	“AssetInfo” : {
	“ID” : “MACH-L1-102”,
	“SN” : “B320-3R821N9-D”,
	“Date” : “2018-09-29”
	},
	“Online” : true,
	“Utilization” : 40.0
	}
	}
	Filter example for payload above:
	Machine.Online=true&&Machine.AssetInfo.ID=MACH-L1-102
	Note: The full path of each filter must be given every time.

4.2.3 Create an Output

An output represents a path to a data point contained in a connection that can be written to.

A. Under *Connections*, find the *Connection*, click the drop-down menu, and click the *Outputs* menu item. See Figure 25.

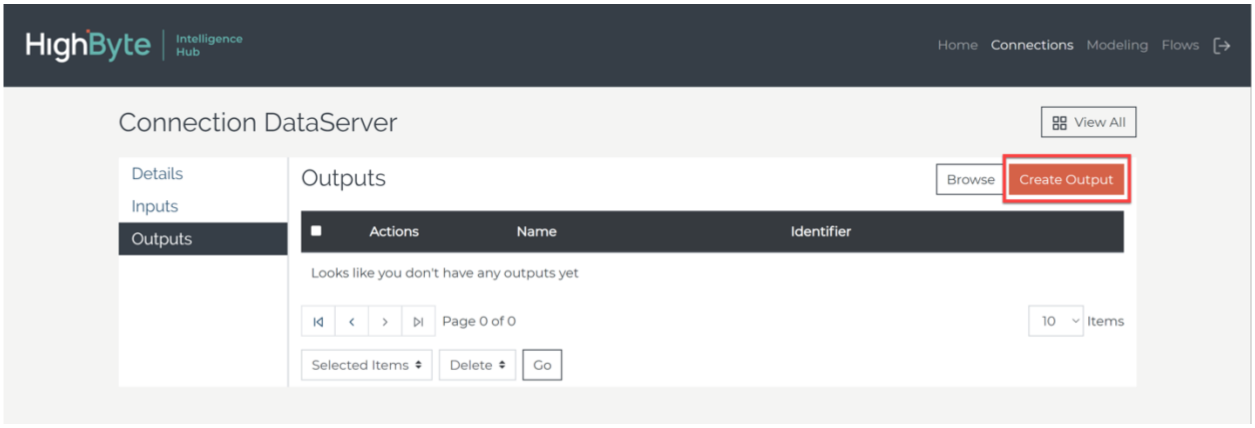


opc.tcp://tpaine-apps:49320

Figure 25

B. Depending on the type of connection (e.g., OPC UA TCP) you may be able to *Browse* for outputs. Follow the same steps described in *Creating an Input* to *View*, *Select*, and *Import Selected Items.*

C. Alternatively, you can click the *Create Output* to manually add an output. See Figure 26.



Looks like you don't have any outputs yet

Figure 26

Set any *General* and *Protocol Specific Settings*. Click *Submit* to add the output and return to the output list. Protocol and General Specific Settings are described below.

General Settings

General	Description
Name	Specifies the name of the output. The name must be unique across all outputs for the specified connection. Names can only contain alphanumeric and underscore characters (e.g., A-Z, a-z, 0-9 or _).

Protocol Specific Settings

AWS IoT SiteWise Settings	**Description**
No output protocol specific settings are required	
Amazon Kinesis Data Streams Settings	
Data Stream	The name of data stream to write to.
Partition Key	An optional field to specify a partition key for the data stream. Outputs with a specified partition key will all write to the same shard. If no partition key is given, the output will disperse requests among all existing shards for the data stream.
Amazon Kinesis Firehose Settings	
Delivery Stream	The name of the delivery stream to write to.
Azure Event Hubs Settings	
No output protocol specific settings are required	
Azure IoT Hub Settings	
No output protocol specific settings are required	
CSV Settings	
Filename	Specifies the name of the file to write to. This includes the file extension and directories. The file path can include dynamic outputs (e.g., /{@connection$input.site}/file.csv)
Delimiter	The delimiter to use in the output file. By default, this is a comma.
Create	When enabled, the file and directory structure are created if they do not exist. If they do exist, data is appeneded to the end of the file.
Google Cloud Pub/Sub Settings	
Topic	The Topic ID as defined in the Google Cloud Pub/Sub configuration. This is the topic the output will publish data to.
InfluxDB Settings	
Bucket	Specifies the the InfluxDB bucket to write data to. This bucket must exist in InfluxDB and be accessible by the connection token.

Tags	The tags associated with the data written to InfluxDB. Tags are a way to organize data in InfluxDB. Each tag is placed on a newline. As an example, tags can be used with dynamic outputs to organize data in an ISA-95 structure.
	site={@this.site}
	area={@this.area}
	Leave this blank if no tags are required.

MQTT Settings	
Topic	Specifies a string that is used to publish MQTT messages. Topics can be static or dynamic. Dynamic topics can include data from the output payload. To include the source input or instance name, use a # character. As an example. /mytopic/# will convert to /mytopic/modelname or /mytopic/opctagname at runtime. To include data from the payload use the following syntax. /mytopic/{@this.siteid} At runtime this takes the siteid attribute of the output payload and includes it in the output topic. One or more dynamic outputs can be defined. ex. /mytopic/{@this.siteid}/{@this.assetid} When using dynamic outputs, if the output payload does not contain the referenced attribute the output will fail to write.
QoS	Specifies the quality of service associated with publishing the output.

Named Root	Specifies how the instance name should show up in the resulting output value.
	When enabled, output values will take the form:
	"InstanceName" : {
	…
	}
	When disabled, output values will take the form:
	{
	"_name" : "InstanceName"
	…
	}

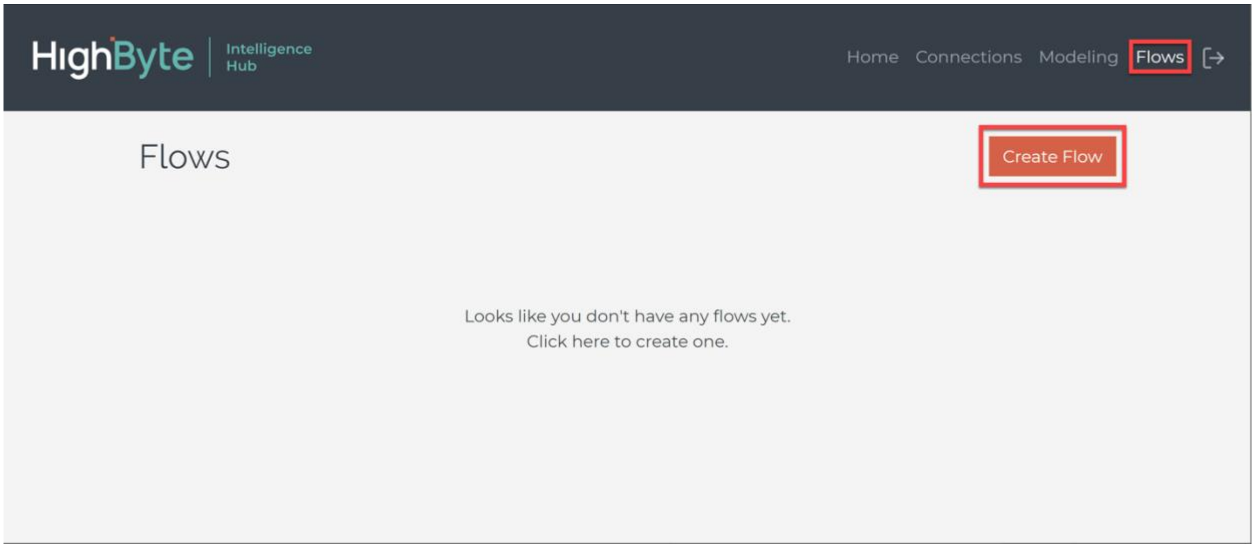
Retain	Specifies whether the MQTT Broker should cache the last value sent.
Breakup Arrays	When true, if the data being sent out is an array type (e.g., an array of SQL rows), this option will send out each element of the array individually. Otherwise the entrie array is sent in one publish. This is useful when used in conjunction with dynamic topics. Each element of the array can be published on a unique topic given data in the payload.
OPC UA Settings	
Namespace	Numeric index of the namespace the identifier belongs to.
Identifier Type	Specifies the type of data used in the identifier field. This can be set to String or Numeric. *This is not the data type of the value associated with this OPC UA output.*
Identifier	Specifies the identifier used to reference the OPC UA output.
	Numeric index of the namespace the identifier belongs to.
OSIsoft PI AF SDK Settings	
Type	Specifies the type of write; options include asset and points. Point writes write data as PI points. Arrays and simple types are not supported. Hierarchy is flattened using the parent.child.attribute syntax. All points created in PI have a source type set to HB. Asset writes data as PI points, and then builds the templates and element hierarchy inside of PI's Asset Framework.
Database	Used for asset writes only. Specifies the AF database to write to.
Path	For asset writes, this specifies the root element in the tree to write to. For example, MyData\\DataHere would add elements as children to DataHere. For point writes, this is a string to prepend to the PI point name.
Asset Name	Used for asset writes only. This overrides the element name assigned to elements. By default, when left blank, the element name is equal to the instance or input name in HighByte Intelligence Hub. This setting supports dynamic outputs, allowing elements to be named based on data from the output payload (e.g., {@this.assetid})
Point Name	Used for point writes only to override the instance or input name portion of the point name. As an example, if the point name is parent.child.attribute, a point name of 'test' would change it to test.child.attribute. Dynamic outputs are supported.
Timestamp	Provides a way to set the timestamp of the write. By default, when left blank, the timestamp is set to the time of the write. This setting is used to pull the timestamp from the payload using dynamic addressing. As an example, if the payload has a mytimestamp attribute, setting this to {@this.mytimestamp} will set the write time to the value of mytimestamp. The attribute must be in ISO8601 format.

Apache Parquet Settings	
Outputs are not currently supported	
SQL (JDBC, MSSQL, MySQL, PostgreSQL) Settings	
Table	The name of the SQL table to write to. The table must already exist and have column names that match the model attribute names. Table names containing capital letters must be placed inside quotes (e.g., "MyTable").
Write Type	Writes to the table can be Inserts, Updates, or Upserts. Inserts add the output data as new rows to the table. Updates update existing rows that have a matching value of the attribute referenced in the Where Column. If there are no matches Update does nothing. Upsert performs an Update if there are matching rows, and if there are no matching rows it performs an Insert.
Where Column	The name of the attribute/column in the output data that's used to match existing rows in the table. This setting is used if the Write Type is Update or Insert. As an example, assume this is set to 'batchId'. The output will take the value of batchId in the output payload, and update all rows in the table whos batchId column matches the batchId value.
Log as JSON	Specifies whether modeled values should be serialized as a single-column JSON blob versus expanding to multiple columns.
Create Table	Specifies whether a table should automatically be created if it does not exist.
REST Client Settings	
Endpoint URL	The endpoint to send the request to (e.g., /routes/myendpoint).
Header	Name=Value header pairs separated by &. (e.g., Content-Type=application/json&key=123). Passed on each request. These are added to the connection level header.
HTTP Method	The method to use: POST or PUT.
Template	If blank, the default JSON output format is used. Otherwise, the format of the output can be modifed using this Apache Freemarker template engine. See Appendix C for more details.
Sparkplug Settings	
Device Id	The id for the attached device the metric belongs to. Do not set the device id if the metric belongs to the EoN. This field supports dynamic oututs.
Metric Name	Optionally specify the metric name to write. Examples include: MetricA, Folder/MetricA If you do not set a metric name, the metric name will be auto generated from the input or model instance name. This allows you to route many inputs and/or instances to a single output and simplifies configuration. This field supports dynamic outputs.
Breakup Arrays	When true, if the data being sent out is an array type (ex. an array of SQL rows), this option will send out each element of the array individually. Otherwise the entrie array is sent in one publish. This is useful when used in conjunction with dynamic device id or metric names. Each element of the array can be published on a unique device or metric given data in the payload.
Webhook Settings	
Outputs are not currently supported	

4.2.4 Create a Flow for a Single Input / Output

A flow defines a mapping of data coming in and data going out of the product. A flow's sources can be one or more simple *Inputs* and/or modeled *Instances*, while its targets are one or more *Outputs*.

A. Click *Flows* in the configuration's Main Menu and then click the *Create Flow* button. See Figure 27.



Looks like you don't have any flows yet.
Click here to create one.

Figure 27

B. Enter a *Name* to represent the flow. Optionally enter in a *Description or* a *Group As* folder to categorize the flow. Click *Next* to continue. See Figure 28.

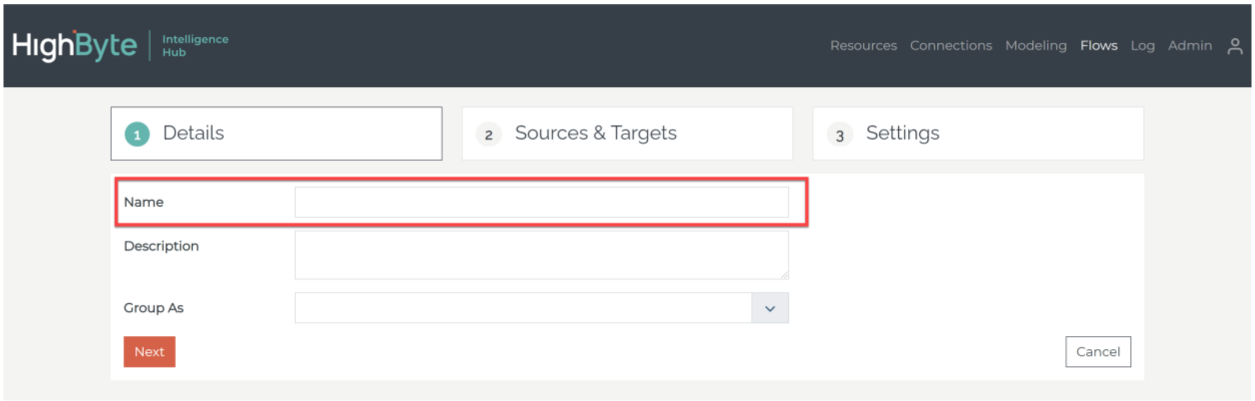
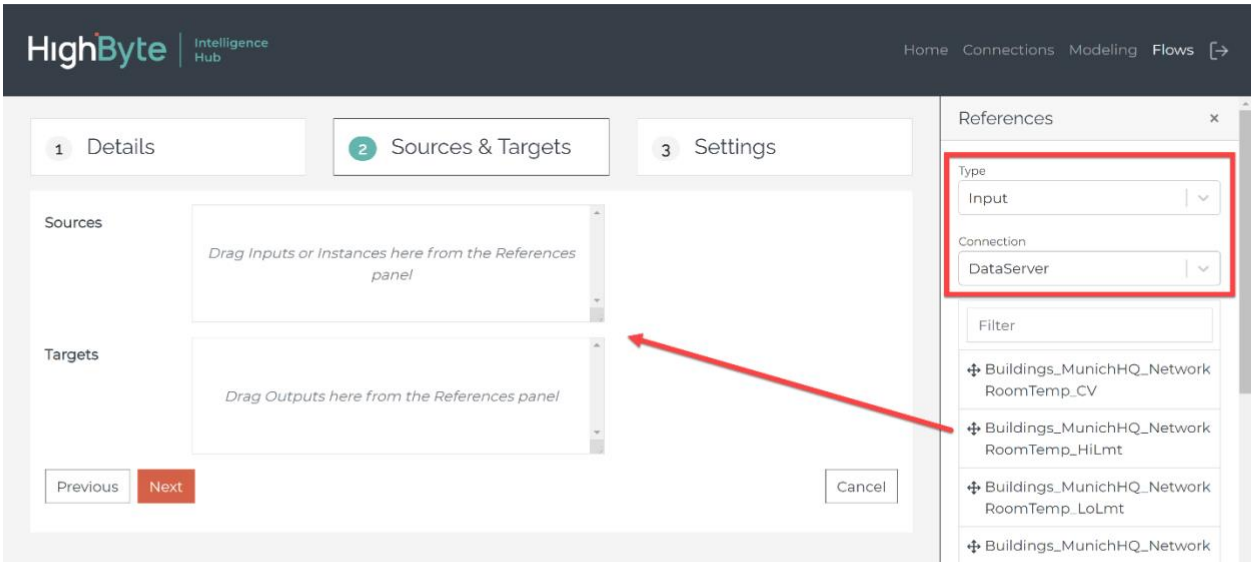


Figure 28

C. To add one or more *Sources,* use the *References* panel to view and select *Inputs* for any *Connection* and/or *Instances* and drag them to the *Sources* field. To add one or more *Targets,* use the *References* panel to view and select *Outputs* for any *Connection* and drag them to the *Targets* field. Click *Next* to continue. See Figure 29.


HighByte | Intelligence Hub
Home
Connections
Modeling
Flows
1 Details
2 Sources & Targets
3 Settings
Sources
Drag Inputs or Instances here from the References panel
Targets
Drag Outputs here from the References panel
Previous
Next
Cancel
References
×
Filter

Type: Input

Connection: DataServer

Buildings_MunichHQ_Network RoomTemp_CV

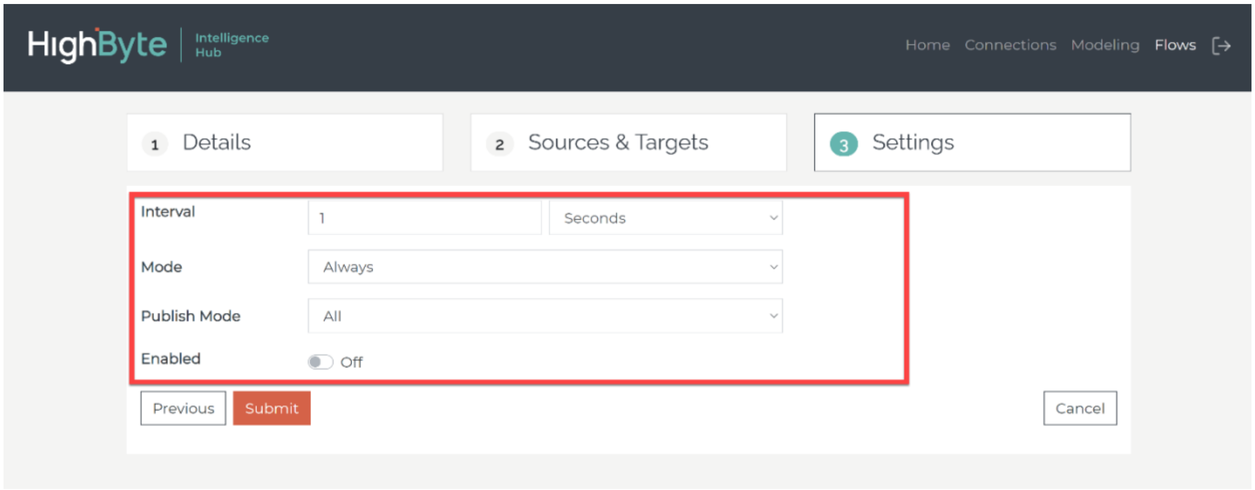
Buildings_MunichHQ_Network RoomTemp_HiLmt

Buildings_MunichHQ_Network RoomTemp_LoLmt

Buildings_MunichHQ_Network

Figure 29

D. Set the additional flow settings described below. Click the *Submit* button when finished. See Figure 30.



Interval: 1 Seconds

Mode: Always

Publish Mode: All

Enabled: Off

Figure 30

Setting	Description
Interval	Specifies how often the flow should be processed. The interval can be specified in milliseconds, seconds, minutes, hours, or days. Valid range is between 10 milliseconds and 36 days.
Mode	Specifies the flow's mode of operation.

<table>
<tr><td></td><td>
<table>
<tr><td><em>Always</em></td><td>Every interval the source will be read and made available to write to the target.</td></tr>
<tr><td><em>OnChange</em></td><td>On the first interval, the expression will be evaluated, the result will be stored, and the source will be read and made available to write to the target.

On successive intervals, the expression will be evaluated. If the result differs from the previous interval, the result will be stored, and the source will be read and made available to write to the target. Otherwise, no action is taken.</td></tr>
<tr><td><em>OnTrue</em></td><td>On the first interval, the expression will be evaluated. If the result evaluates to True, the source will be read and made available to write to the target. Otherwise, no action is taken.

On successive intervals, the expression will be evaluated. If the result evaluates to True, and on the previous interval the expression evaluated to False, the source will be read and made available to write to the target. Otherwise, no action is taken.</td></tr>
<tr><td><em>WhileTrue</em></td><td>Every interval the expression will be evaluated. If the result evaluates to True, the source will be read and made available to write to the target. Otherwise, no action is taken.</td></tr>
</table>
</td></tr>
<tr><td>Expression</td><td>Depending on the mode, a JavaScript expression that assigns a simple input or performs a calculation on one or more inputs. This field is not applicable for all modes. See <em>Appendix B</em> for more information on <em>Expressions.</em>

<em>Note: The special value of internal$lastExpressionValue can be used in flow expressions to perform deadband calculations. Internal$lastExpressionValue evaluates to the value returned by the last execution of the expression. Here is an example of how to perform deadband calculations on an input, where connection$input is the input, and the deadband is 0.1:</em>

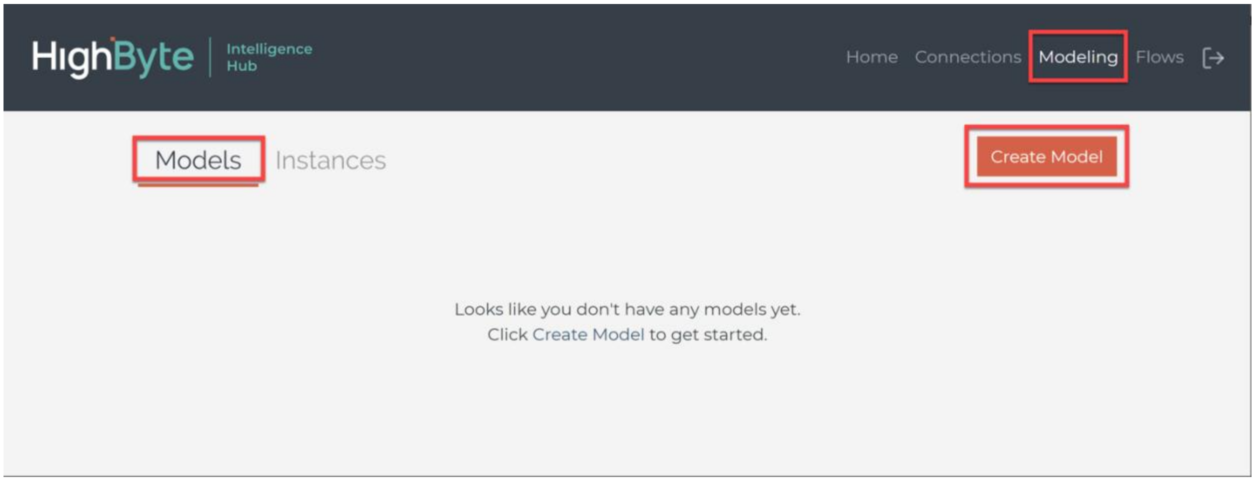
<em>(Math.abs(connection$input – internal$lastExpressionValue) > 0.1) ? connection$input : internal$lastExpressionValue</em></td></tr>
<tr><td>Publish Mode</td><td>Specifies which source values to publish to the target.
<table>
<tr><td><em>All</em></td><td>Publish all successfully read source values at each interval.</td></tr>
<tr><td><em>OnlyChanges</em></td><td>Publish successfully read source values that have changed from the previous interval. For modeled instances all attribute values are published.</td></tr>
<tr><td><em>OnlyChangesCompressed</em></td><td>Publish successfully read source values that have changed from the previous interval. For modeled instances only the attribute values that changed are published.</td></tr>
</table>
</td></tr>
<tr><td>Enabled</td><td>The <em>enabled</em> state of the flow. When disabled, sources are not read, and nothing is written to the target.</td></tr>
</table>

4.2.5 Create a Model

Models are a standard representation of logical assets, processes, products, systems, or roles. A model is comprised of a collection of attributes that are common to the logical item and forms the basis for standardizing data across a wide range of raw input data.

Models are leveraged through the creation of a model *Instance*, each of which are unique to a specific instance of an asset, process, product, system, or role.

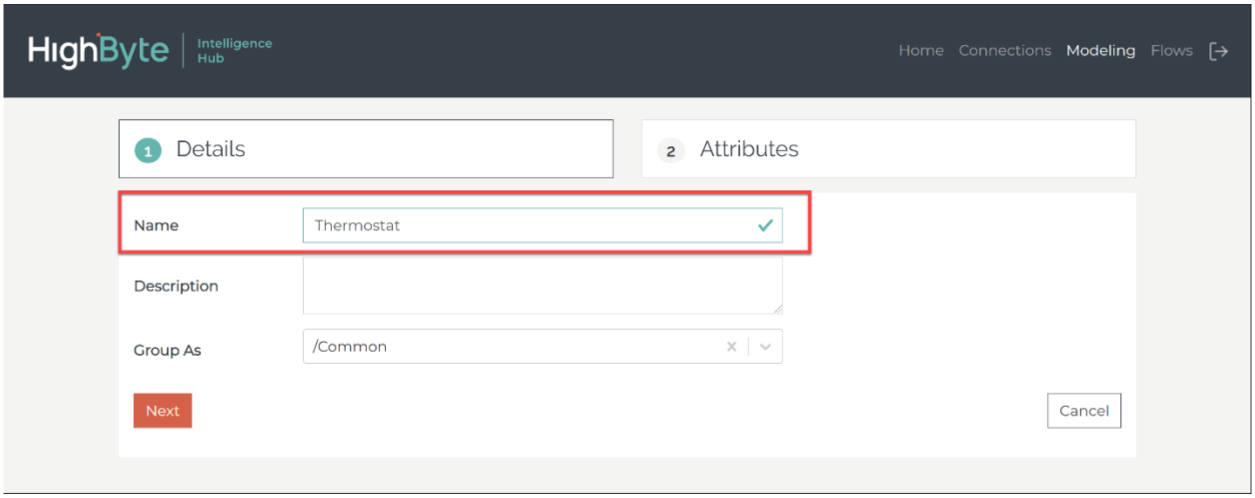
A. Click *Modeling* in the configuration's Main Menu, click the *Models* tab, and then click the *Create Model* button. See Figure 31.



Looks like you don't have any models yet.
Click Create Model to get started.

Figure 31

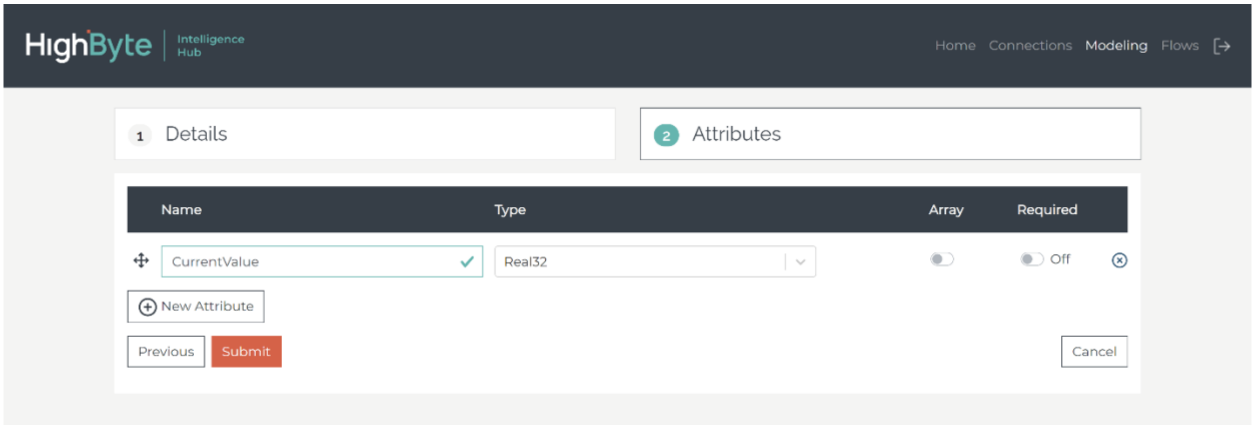
B. Enter a *Name* to represent the model. Optionally enter in a *Description* and a *Group As* folder to categorize the model. Click *Next* to continue. See Figure 32.



Name: Thermostat
Description:
Group As: /Common

Figure 32

C. Click the *New Attribute* and set the attribute's *Name*, *Type*, and *Required* fields. Continue to add additional attributes that should be assigned to this model and click the *Submit* button when finished. See Figure 33.



Name	Type	Array	Required
CurrentValue	Real32		Off

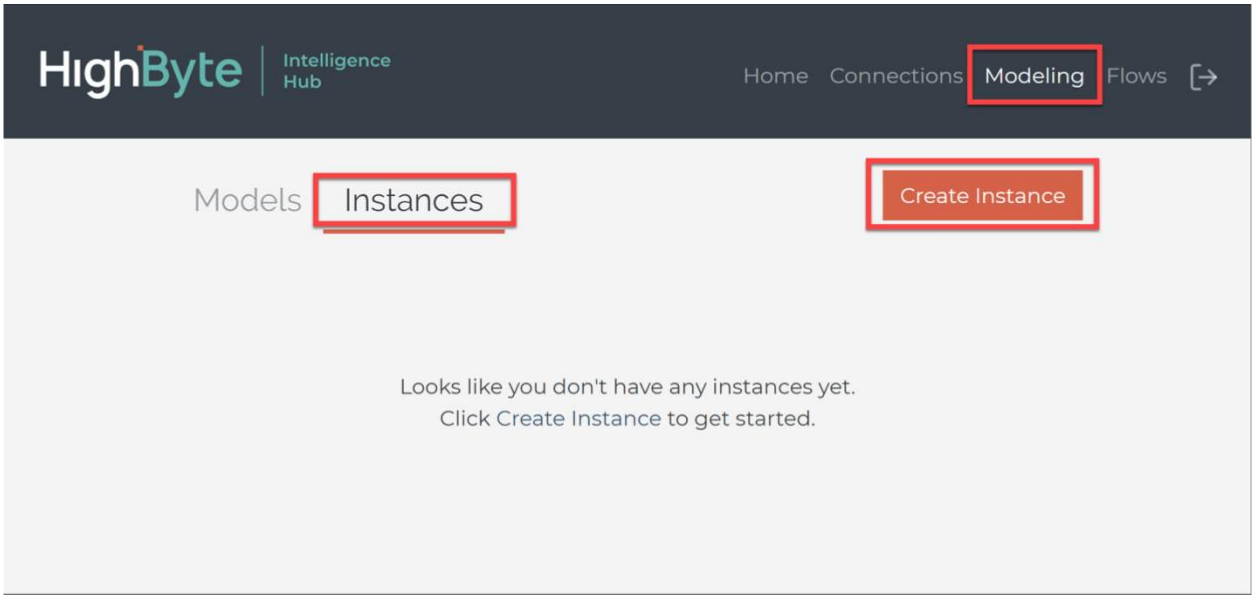
Figure 33

Attribute Setting	**Description**
Name	The name for the attribute to be included in the model.
Type	The type of value that will be assigned to the attribute. The type can be a primitive type, a reference to another model or the Any type. Primitive types include Boolean, String, Int8, Int16, Int32, Int64, Real32 and Real64 (where 8, 16, 32 and 64 indicate the number of bits in the Integer or Real value). Reference a model within a model to develop complex models and reuse previous work. When referencing a model select an existing model or create a new model inline. Specifying Any will take on the type of the value assigned to the attribute, including modeled or complex data.
Array	Specifies if the attribute is array of values of the specified type.
Required	Specifies if an instance of this model must populate the attribute.

4.2.6 Create a Modeled Instance

Whereas a *Model* specifies the standardization of an asset, process, product, system or role, a model *Instance* represents a real-time representation of one of these items. For example, a *Model* may be created to represent how a Quality Manager's view of a manufacturing line will be standardized. If there are 10 manufacturing lines, 10 modeled instances would be created and populated with data to represent each one.

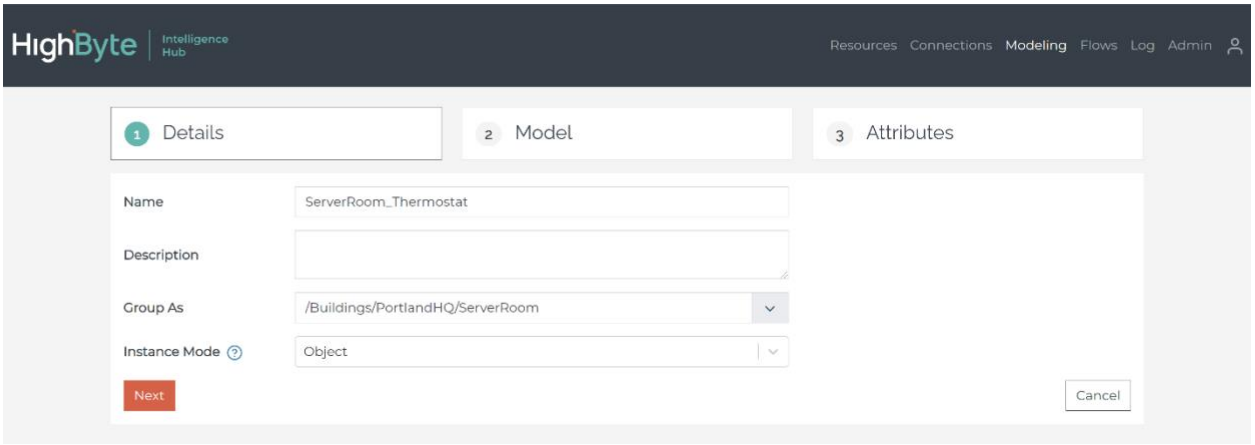
A. Click *Modeling* in the configuration's Main Menu, click the *Instances* tab, and then click the *Create Instance* button. See Figure 34.



Looks like you don't have any instances yet.
Click Create Instance to get started.

Figure 34

B. Enter a *Name* to represent the instance. Optionally enter in a *Description* and a *Group As* folder to categorize the instance. Set the model *Instance Mode*. When in *Array* mode the instance expands an array input, producing an array of transformed outputs. The is commonly used when transforming SQL rows through a model. Each row is applied to the instance and the output is an array of the transformed outputs. When in *Object* mode, only a single instance is output Click *Next* to continue. See Figure 35.



Name: ServerRoom_Thermostat

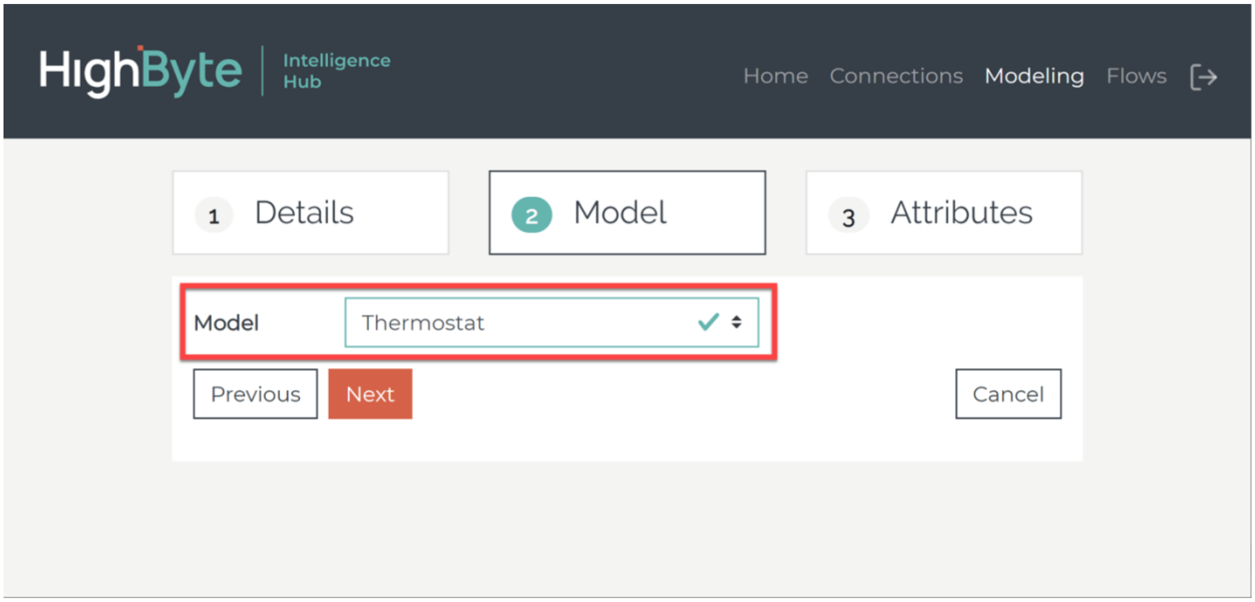
Description:

Group As: /Buildings/PortlandHQ/ServerRoom

Instance Mode: Object

Figure 35

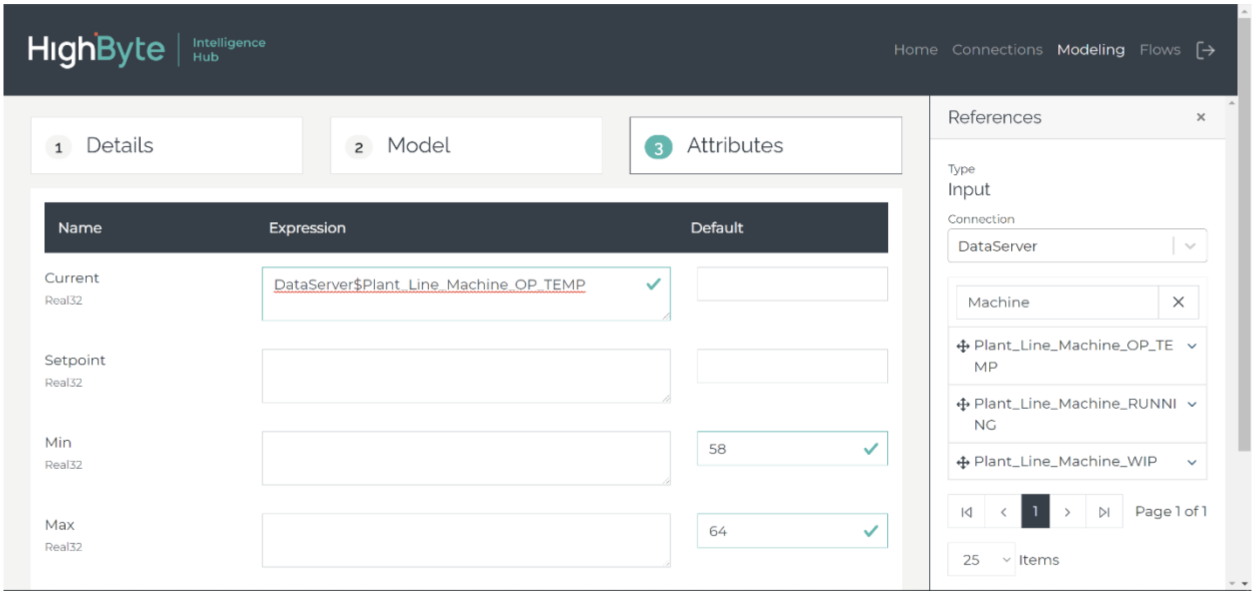
C. Select the *Model*. This will automatically pre-populate the attributes to be assigned to this instance. Click *Next* to continue. See Figure 36.



Model: Thermostat

Figure 36

D. Set an *expression* and/or *default* for each attribute. Use the *References* panel to find and select *Inputs* of interest and drag them into any expression. Click the *Submit* button when finished. See Figure 37.



Name	Expression	Default
Current Real32	DataServer$Plant_Line_Machine_OP_TEMP	
Setpoint Real32		
Min Real32		58
Max Real32		64

References

Type: Input

Connection: DataServer

Machine

- Plant_Line_Machine_OP_TEMP
- Plant_Line_Machine_RUNNING
- Plant_Line_Machine_WIP

25 Items

Figure 37

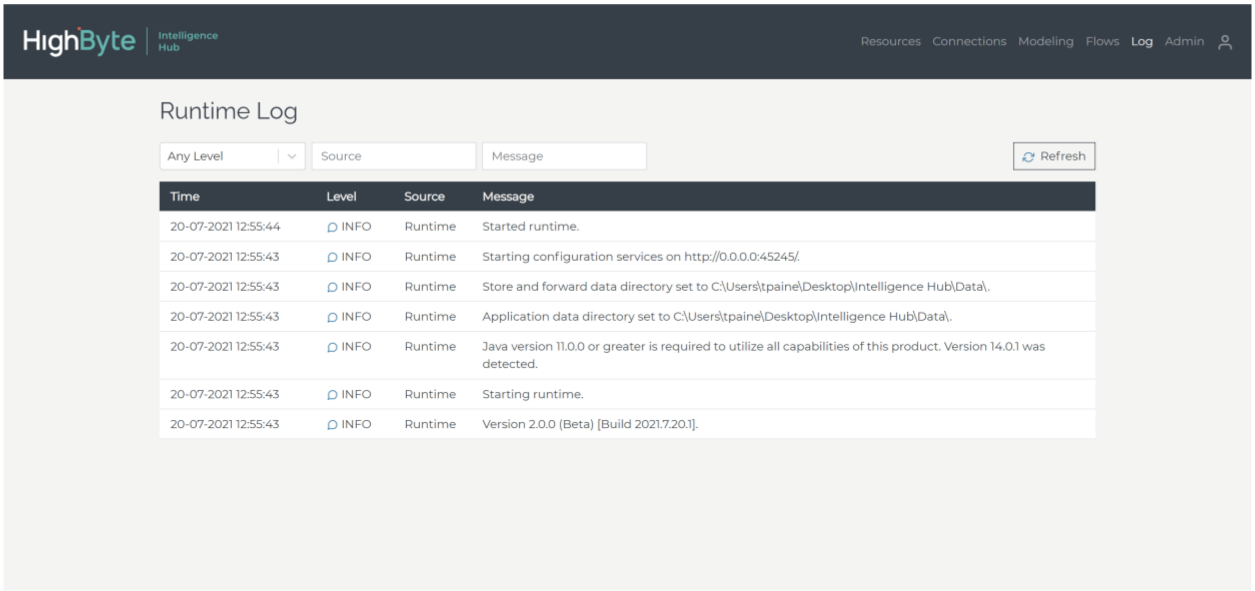
See *Appendix B* for more information on *Expressions and Default Values.*

4.2.7 Create a Flow with a Modeled Instance

1. Follow the same steps as in *Create a Flow for a Simple Input / Output*, only add one or more instances as a source.

4.3 Event Log and Troubleshooting

The Event Log is available via the main menu under *Log* and contains the runtime and audit log messages generated by the selected hub. The log is primarily used for troubleshooting connection and flow errors. See Figure 38.



Runtime Log

Time	Level	Source	Message
20-07-2021 12:55:44	INFO	Runtime	Started runtime.
20-07-2021 12:55:43	INFO	Runtime	Starting configuration services on http://0.0.0.0:45245/.
20-07-2021 12:55:43	INFO	Runtime	Store and forward data directory set to C:\Users\tpaine\Desktop\Intelligence Hub\Data\.
20-07-2021 12:55:43	INFO	Runtime	Application data directory set to C:\Users\tpaine\Desktop\Intelligence Hub\Data\.
20-07-2021 12:55:43	INFO	Runtime	Java version 11.0.0 or greater is required to utilize all capabilities of this product. Version 14.0.1 was detected.
20-07-2021 12:55:43	INFO	Runtime	Starting runtime.
20-07-2021 12:55:43	INFO	Runtime	Version 2.0.0 (Beta) [Build 2021.7.20.1].

Figure 38

Each log message includes the time (localized), the event level (e.g., info, warning, error, fatal, or audit), the source (the Runtime or a Connector), and the message that can be used for troubleshooting.

The log can easily be searched by the Level, Source, or a general filter that looks at the message body.

5.0 Central Configuration

The Intelligence Hub supports the ability for any hub to act as a central hub, allowing for the central hub to act as a single interface for configuring multiple remote hubs. This is useful when a single facility has many Intelligence Hub instances running closer to the machines or in cases where it is easier to manage many Intelligence Hubs centrally, either by application or facility.

5.1 Enable Central Configuration

To enable central configuration mode for a specific Intelligence Hub instance, edit the ‘intelligencehub-settings.json’ file in the runtime directory, and set the centralConfig property to true.

```
{
 "settings": {
  "configuration": {
   "scheme": "http",
   "port": 45245,
   "centralConfig": true
  }
 }
}
```

Restart the Intelligence Hub runtime for this change to take effect. Once restarted, refresh the browser and connect to the Intelligence Hub instance that is configured for central management. A new Network tab is now available. See Figure 39.

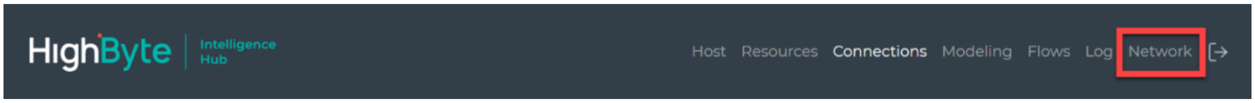
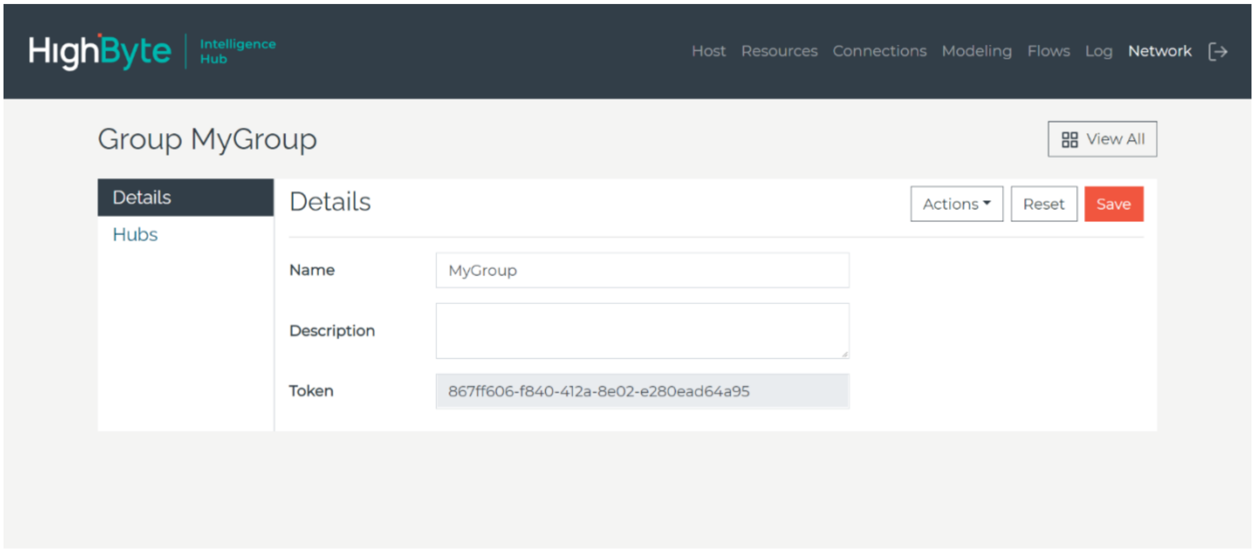


Figure 39

5.2 Create a Network Group

A network group represents a group of remote hubs and contains the credentials a remote hub needs to connect to the central hub. To create a network group, perform the following steps. See Figure 40.

A. *Click Network* and select the *Hubs* tab to list all *Network Groups* previously configured.
B. *Click* the *Create Network Group* button to create a new group.
C. *Enter* a Name for the group and an optional description.
D. *Click* the *Save* button for the change to take effect.
E. A *Token* will be automatically created for the *Network Group.* This *Token* will be required by a corresponding remote Intelligence Hub for authentication with this central hub.



Group MyGroup

Details

Name: MyGroup

Description:

Token: 867ff606-f840-412a-8e02-e280ead64a95

Figure 40

5.3 Connect a Remote Hub

Once a Network Group is created, perform the following steps to connect a remote hub to the central hub.

> The remote hub connects to the central hub's REST API (default port 45245) using websockets. The central hub needs to expose a websockets port to allow remote hubs to connect.

A. *Install* the Intelligence Hub *runtime component* on another machine. Only the runtime is required, but you may optionally install the configuration component to allow for direct configuration if desired. See section *Intelligence Hub Runtime Installation* above.

B. *Create* a new *intelligencehub-remoteconfig.json* file in the runtime directory. Define the file as follows. An *intelligencehub-remoteconfig.json.template* file is provided in the runtime directory as a starting point.

```
{
  "remoteUrl" : "ws://hostname:45245/websocket",
  "token": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

- *remoteUrl* is the URL to the central host's REST API. By default, this is port 45245 and uses web sockets (ws) over http.

  Note: HighByte recommends the use of HTTPS for accessing the REST API for secure websocket communications.

- *token* is the token automatically generated when creating a Network Group on the central hub and is used to authenticate the remote hub.

C. *Save* this file and *restart* the runtime for this remote hub installation. The remote hub will connect to the central host.

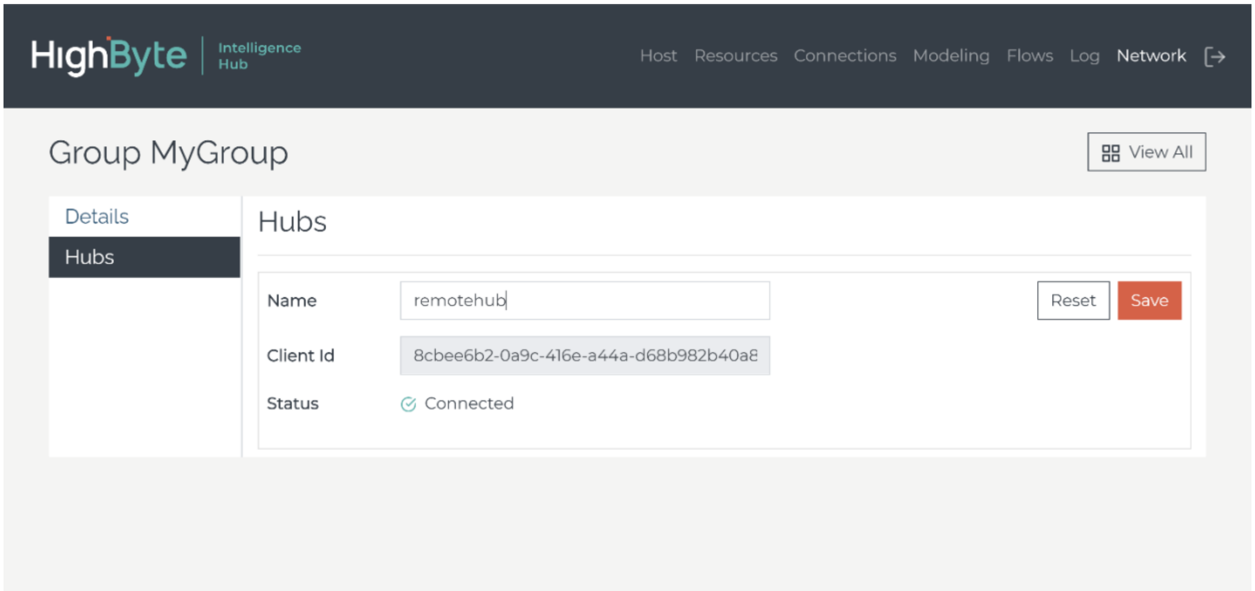
Upon successfully connecting, the following event message will be emitted: “Remote configuration connected to ws://hostname:45245/websocket”.

5.4 Rename the Remote Hub

By default, the remote hub connects to the central hub and uses its machine name as an identifier. This identifier can be changed by performing the following steps. See Figure 41.

A. Click *Network* and select the *Hubs* tab to list all Network Groups previously configured.
B. Select the corresponding *Network Group* in the list and *View* its *Details*.
C. Click *Hubs* to list all currently connected Hubs.
D. Rename the hub and click *Save.*

> The hub’s *Connected State* and its Client ID is also displayed. The Client ID is automatically assigned to the remote hub when it first connects. On future connections, the remote hub provides the Client ID to identify the hub.



Group MyGroup

Hubs

Name: remotehub

Client Id: 8cbee6b2-0a9c-416e-a44a-d68b982b40a8

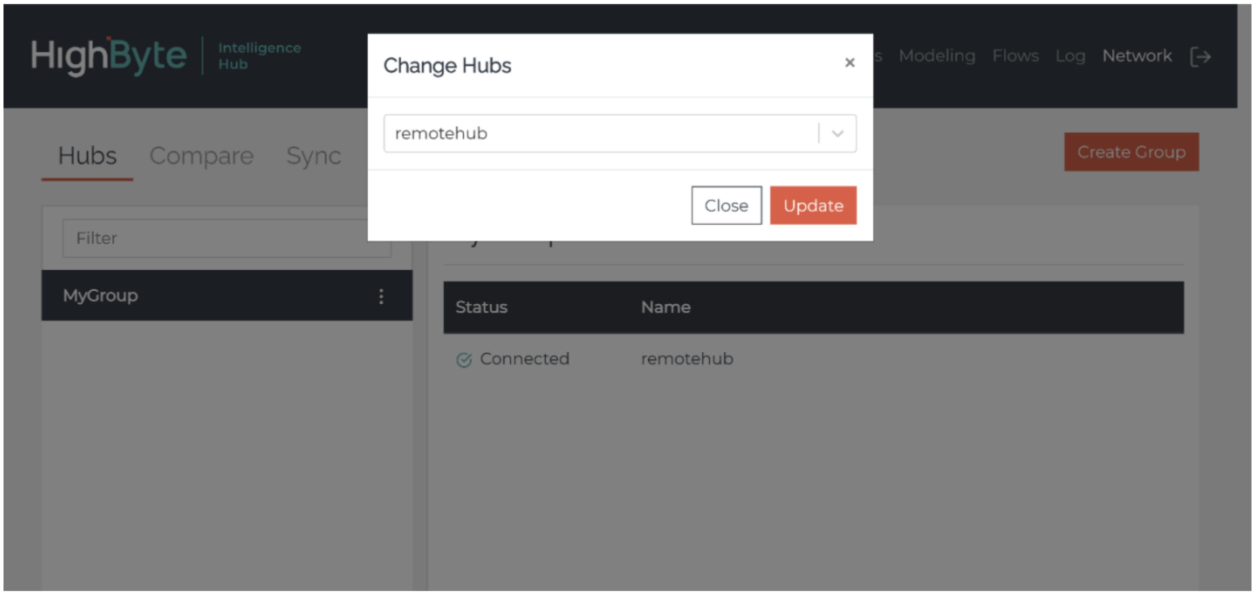
Status: ✓ Connected

Figure 41

5.5 Remotely Configure the Hub

To configure the connected remote hub, perform the following steps. See Figure 42.

A. Click *Host* from the main menu.
B. Select the remote hub to configure in the *Change Hubs* dialog and click *Update*.
C. All configuration changes will now be proxied to the remote hub. When finished, select another remote hub to configure or click *Host* to work with the central hub.



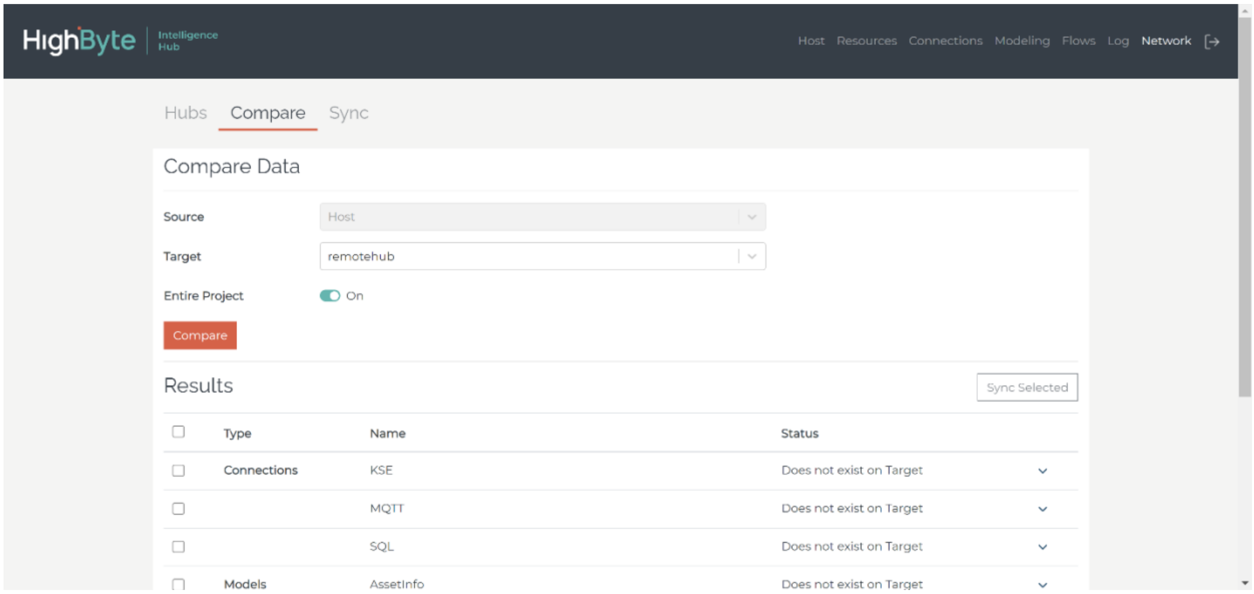
Status	Name
Connected	remotehub

Figure 42

5.6 Compare Two Hubs

Using the central host, compare the configuration of any two hubs by performing the following steps. See Figure 43.

A. Click *Network* and select the *Compare* tab.
B. The *Source* hub is the current hub being configured. Change the source by using the *Change Hubs* dialog.
C. The *Target* is the hub being compared to.
D. *Compare* the *Entire Project (On)* or *Specific Objects (Off)* between the two hubs. When comparing specific objects (e.g., Connections and Models), use the *References* panel to drag and drop which objects should be compared.
E. Click *Compare*.
F. The *Results* show the results of the comparison, including objects that are different or missing. Expand each element to see what is different between two objects.



Compare Data

Source: Host

Target: remotehub

Entire Project: On

Compare

Results

Sync Selected

☐	Type	Name	Status
☐	Connections	KSE	Does not exist on Target
☐		MQTT	Does not exist on Target
☐		SQL	Does not exist on Target
☐	Models	AssetInfo	Does not exist on Target

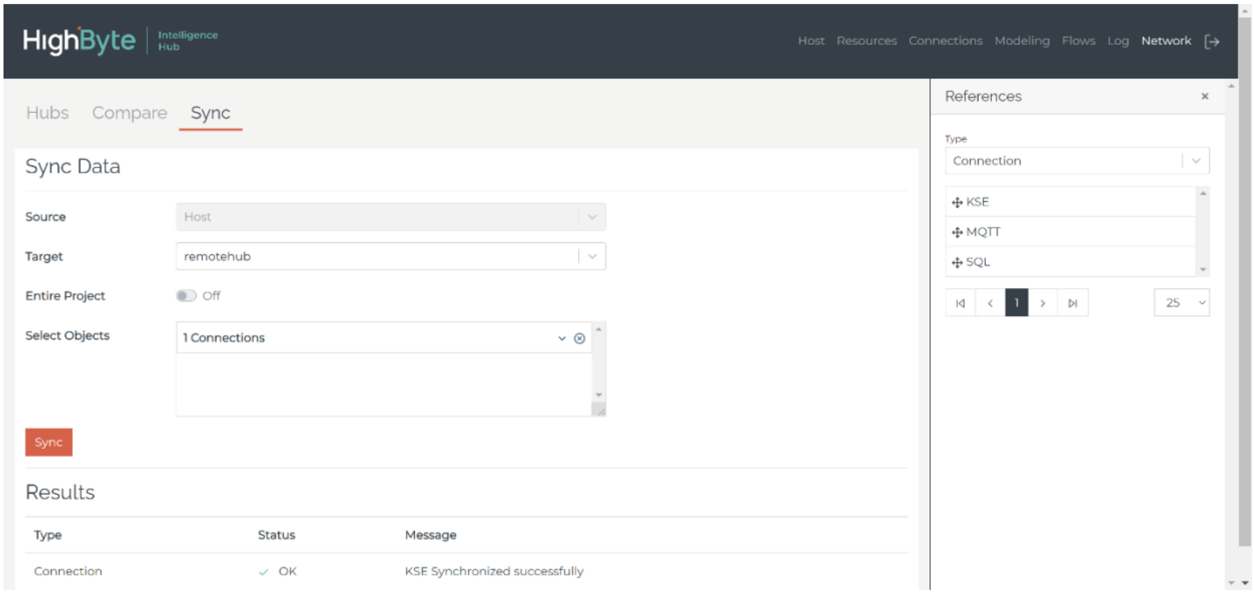
Figure 43

5.7 Sync Objects Between Hubs

Using the central host, synchronize connections, certificates, models, or entire projects between two hubs. To synchronize objects, perform the following steps. See Figure 44.

A. Click *Network* and select the *Sync* tab.
B. The *Source* hub is the current hub you are configuring. Change the source by using the *Change Hubs* dialog.
C. The *Target* is the hub being synchronized.
D. Use the *References* panel to drag and drop the *Connections* and *Models* to synchronize.
E. Click *Sync*.
F. The *Results* show whether synchronization succeeded for each object.

> Synchronization cannot be rolled back or reverted and partial success is possible if some objects fail to synchronize.


HighByte Intelligence Hub
Host Resources Connections Modeling Flows Log Network
Hubs Compare Sync
References ×
KSE
MQTT
SQL
1
25

Sync Data

Source: Host

Target: remotehub

Entire Project: Off

Select Objects: 1 Connections

Sync

Results

Type	Status	Message
Connection	✓ OK	KSE Synchronized successfully

References

Type: Connection

Figure 44

Appendix A – Security

Storing Secure Information

Passwords and other secure information are encrypted and stored in the intelligencehub-configuration.json file. In order to move this configuration between Intelligence Hub instances, the intelligencehub-certificatestore.pkcs12 and intelligencehub-configuration.json must both be moved to the new location.

AWS Token Security

AWS IoT SiteWise

The AWS IoT SiteWise connector uses the AWS SDK to communicate to IoT SiteWise, which is REST based. The Access Key and Secret Key must be acquired for an IAM user that is in a group with access to the AWS IoT SiteWise API. This permission in AWS is called 'AWSIoTSiteWiseFullAccess'. This role is used to publish models, assets, and data to IoT SiteWise.

Amazon Kinesis Data Streams

The Amazon Kinesis Data Streams connector uses the AWS SDK to communicate to Amazon Kinesis Data Streams, which is REST based. The Access Key and Secret Key must be acquired for an IAM user that is in a group with access to the Amazon Kinesis Data Streams API. This permission in AWS is called 'AmazonKinesisFullAccess'. This role is used to publish data to Amazon Kinesis Data Streams.

Amazon Kinesis Data Firehose

The Amazon Kinesis Firehose connector uses the AWS SDK to communicate to Amazon Kinesis Data Firehose, which is REST based. The Access Key and Secret Key must be acquired for an IAM user that is in a group with access to the Amazon Kinesis Data Firehose API. This permission in AWS is called 'AmazonKinesisFirehoseFullAccess'. This role is used to publish data to Amazon Kinesis Data Firehose.

IAM Best Practices

Please see AWS documentation on IAM best practices. HighByte strongly recommends following the policy of least privilege when granting the IAM role for the connection.

- https://docs.aws.amazon.com/iot-sitewise/latest/userguide/security-iam.html
- https://docs.aws.amazon.com/iot/latest/developerguide/security-best-practices.html

It is also recommended that users occasionally rotate new IAM credentials and manually update the Intelligence Hub's configuration with the new credentials.

Azure IoT Hub & Azure Event Hubs

Connection String

Azure IoT Hub and Azure Event Hubs support security via their connection string. The communications are secured by MQTT/TLS, HTTPS, or AMQP for the IoT Hub and AMQP for the Event Hubs.

Microsoft SQL Server, MySQL, PostgreSQL

Username / Password

The Intelligence Hub supports username and password security for all SQL connections. The password is encrypted and sent to the database via the JDBC driver implementation. The username and password can be included as part of the Additional Properties of the JDBC connection string, but it is strongly recommended that the user use the specific username and password fields as these are encrypted by the Intelligence Hub before storing them in the project configuration file.

MQTT Security

SSL

The Intelligence Hub supports the ability to protect the contents of MQTT messages communicated with an MQTT broker by leveraging the transport level security mechanisms supported by TCP/IP and adopted by the MQTT specification. Consult your MQTT broker documentation on how to require SSL for secure connectivity.

When using SSL, specifically with AWS IoT Core, AWS IoT Greengrass, or any broker that has a self-signed certificate, you will need to import the certificate into the HighByte Intelligence Hub certificate store. See the "Managing Certificates" section on how to import the certificates. You will need to import the self-signed CA certificate. You may also need to add the Client Certificate (as a certificate) and Client Key (as a key) if the broker is using certificate-based authentication.

Note: HighByte recommends the use of SSL for MQTT communications.

Username/Password

The Intelligence Hub supports the ability to set a username and password to be passed to the MQTT broker as part of its authentication process. Consult your MQTT broker documentation on how to require username/password authentication.

Note: Per the MQTT specification, the username and password are only protected when sent over the wire if used in conjunction with SSL. Caution should be exercised when using a username and password with SSL disabled.

OPC UA Security

Security Policy: Basic256Sha256 Sign | SignEncrypt

The Intelligence Hub supports the Basic256Sha256-Sign and Basic256Sha256-SignEncrypt security policies defined by the OPC UA specification.

To simplify the configuration and use of these security policies, the Intelligence Hub behaves as follows:

1. Automatically creates an application instance certificate the first time it is run.
2. Connects to servers using asymmetric security algorithms encryption to securely exchange public keys and create a secure connection.
3. Signs or Signs and Encrypts all requests before sending it to the server.
4. Decrypts (if applicable) and verifies the signature of all responses coming from the server.

> The first time the Intelligence Hub makes a secure connection to any OPC UA server, you will need to Trust the Intelligence Hub application certificate where the server is installed. Consult your OPC UA server documentation for instructions.

The Intelligence Hub will continually try to establish a secure connection until its certificate is trusted or it is shutdown. There is no need to restart the Intelligence Hub for these changes to take effect.

Security Policy: None

The Intelligence Hub supports the ability to turn off security as allowed by the OPC UA specification. In this mode, communications will not be encrypted nor signed, and as such will not be private and not be verified as coming from a trusted source.

Authentication: Username/Password

The Intelligence Hub allows connections to the OPC UA server with specific username and password credentials. Passwords are always encrypted using Basic256 encryption defined by the OPC UA specification. Consult your OPC UA server documentation for instructions on how to setup username/password authentication.

Authentication: Anonymous

The Intelligence Hub allows connections to the OPC UA server as an anonymous user. Consult your OPC UA server documentation for instructions on how to allow anonymous users.

Note: HighByte recommends using a minimum of Basic256Sha256–Sign along with username/password authentication for all OPC UA communications.

REST Client

Username/Password

The REST Client supports username and password Basic Authentication. Note if the endpoint URL is HTTPS the credentials are encrypted, but with HTTP they will be sent as plain text.

Token

The REST Client also supports token-based authentication, where the token is added to the Input/Output URL as a params (i.e., /endpoint?token=123) or the token is added to the Header (i.e., content-type=json&token=123).

External Identity Provider Setup

The Intelligence Hub supports the ability for users to utilize an external identity provider for authorization and authentication of the Intelligence Hub. The setup of the external identity provider is handled in the intelligencehub-settings.json file.

Note: The Intelligence Hub only supports authorization and authentication with external identity providers when self-hosting the Intelligence Hub. If you are using third-party hosting to host the configuration, you will not be able to login using an external identity provider. With third-party hosting, only the internal identity provider is supported.

SAML 2.0 Settings

To setup SAML 2.0, create a saml2 object inside the authentication object in the settings file. Once all SAML 2.0 settings are specified, add “saml2” to the providers array in authentication.

Settings	Description
sp.nameidformat	Specifies the name identifier used to represent the requested subject. Supported NameIdFormats: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress, urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName, urn:oasis:names:tc:SAML:1.1:nameidformat:WindowsDomainQualifiedName, urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos, urn:oasis:names:tc:SAML:2.0:nameid-format:entity, urn:oasis:names:tc:SAML:2.0:nameid-format:transient, urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted
sp.entityid	Specifies the identifier of the Intelligence Hub entity.
sp.assertion_consumer_service.url	Specifies where the SAML assertion gets sent to. Depending on where the Intelligence Hub is hosted, the URL for this will look like: http(s)://host:port/config/login. **Note: When self-hosting, ensure the http(s)://host:port in the configuration/config/settings.json URL matches the http(s)://host:port in the sp.assertion_consumer_service.url.**
sp.privatekey	The name of the private key used to sign SAML messages. The private key should also be stored under this name in the Intelligence Hub KeyStore. This field should only be used when the AuthnRequest is encrypted and signed. Note: Refer to section 4.2.5 for more information.
sp.x509cert	The name of the public key used to encrypt the SAML response. The certificate should also be stored under this name in the Intelligence Hub KeyStore. This field should only be used when the AuthnRequest is encrypted and signed. Note: Refer to section 4.2.5 for more information.
idp.entityid	Identifier of the IdP entity.
idp.single_sign_on_service.url	Single sign on endpoint of the IdP. The URL where the AuthnRequest is sent.
idp.single_sign_on_service.binding	The SAML protocol binding used to deliver the AuthnRequest message. Note: The Intelligence Hub currently only supports urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
idp.x509cert	The name of public x509 certificate of the IdP. The certificate should also be stored under this name in the Intelligence Hub KeyStore. Note: Refer to section 4.2.5 for more information.
security.signature_algorithm	Algorithm that the toolkit will use on the signing process. Supported algorithms:

	http://www.w3.org/2000/09/xmldsig#rsa-sha1, http://www.w3.org/2000/09/xmldsig#dsa-sha1, http://www.w3.org/2001/04/xmldsig-more#rsa-sha256, http://www.w3.org/2001/04/xmldsig-more#rsa-sha384, http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
security.digest_algorithm	Algorithm that the toolkit will use on the digest process. Supported algorithms: http://www.w3.org/2000/09/xmldsig#sha1, http://www.w3.org/2001/04/xmlenc#sha256, http://www.w3.org/2001/04/xmldsig-more#sha384, http://www.w3.org/2001/04/xmlenc#sha512
security.requested_authncontext	The authentication context. If this field is not included no AuthContext will be sent in the AuthnRequest. Supported AuthContext: urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified, urn:oasis:names:tc:SAML:2.0:ac:classes:Password, urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport, urn:oasis:names:tc:SAML:2.0:ac:classes:X509, urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI, urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos, urn:federation:authentication:windows, urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient, urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken
security.want_assertions_signed	Set to true if the assertion received by the Intelligence Hub are signed. Set to false if it is not signed.
security.authnrequest_signed	Set to true if the AuthnRequest sent by the Intelligence Hub is signed. Set to false if it is not signed.
security.want_assertions_encrypted	Set to true if the assertions received by the Intelligence Hub are encrypted. Set to false if it is not.
parsing.trim_name_ids	Set to true if the name IDs received by the Intelligence Hub need to be trimmed. Set to false if it is not.
Parsing.trim_attribute_values	Set to true if the attribute values received by the Intelligence Hub need to be trimmed. Set to false if it is not.

Retrieving Roles with SAML 2.0

When logging in using SAML 2.0, the Intelligence Hub depends on the identity provider to send over the corresponding role names for that user. SAML 2.0 supports custom attributes, and the Intelligence Hub looks for a custom attribute named 'roles'. The custom attribute, roles, contains the names of roles that are defined in the Intelligence Hub. If the SAML 2.0 assertion contains the name of a role that does not exist in the Intelligence Hub, that role is ignored.

Here is an example SAML 2.0 implementation in the settings file:

```
"authentication": {

    "providers": ["saml2"],

    "saml2": {

      "sp.nameidformat": "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress",

      "sp.entityid": "Intelligence Hub entity id",

      "sp.assertion_consumer_service.url": "http(s)://host:port/config/login/idp",

      "idp.entityid": "IDP entity id",

      "idp.single_sign_on_service.url": "IDP login in URL",

      "idp.single_sign_on_service.binding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect",
      "idp.x509cert": "IDPCertificate",
      "security.signature_algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
      "security.digest_algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
      "security.requested_authncontext": "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport",
      "security.want_assertions_signed": false,
      "security.authnrequest_signed": false,
      "security.want_assertions_encrypted": false
    }
}
```

Appendix B – Expressions and Default Values

Expressions

An expression is simply the right-hand side of an assignment to the attribute it is attached to and can be as simple as an assignment of an *Input*, *Modeled Instance, Complex Data Element*, or as complex as a computation on one or more *Inputs, Modeled Instance Attributes* or *Complex Data Elements*. The expression forms the basis for conditioning data. The syntax for an expression is basic JavaScript (widely used and documented online).

Some examples include:

Expression	**Description**
@OPC_Server$Room2B_TSTAT_CV	A simple assignment of the raw value associated with input *Room2B_TSTAT_CV* on connection *OPC_Server*.
@SQL$QueryAssetInfo.AssetID	A complex data element assignment. Some connectors (e.g., REST, Sparkplug, SQL) return complex data values that contain one or more element values. In this example, a SQL connection named *SQL* has an input called *QueryAssetInfo* that returns a result set for which the element *AssetID* value is assigned.
@OPC_Server$LineAPartsProduced + @OPC_Server$LineBPartsProduced + @OPC_Server$LineCPartsProduced	Assigns the summation of three inputs named *LineAPartsProduced*, *LineBPartsProduced* and *LineCPartsProduced* on connection *OPC_Server*.
@OPC_Server$ProductID.split (\"-\") [1]"	Assuming the raw value of ProductID is a string value that is always formatted as follows: AA-BBBBB-C-DDD. Assigns the substring BBBBB from the raw value associated with input *ProductID* on connection *OPC_Server.*
Math.sin (@OPC_Server$Device.Signal)	Calculates the sine of the raw value associated with input *Signal* on connection *OPC_Server* and assigns the result.
@ServerRoom_Thermostat	A simple assignment of a modeled instance within a parent modeled instance definition.
@ServerRoom_Thermostat.Current > @ServerRoom_Thermostat.Max	A logical expression that tests if the value associated with attribute *Current* of instance *ServerRoom_Thermostat* is greater than the value associated with attribute Max of the same named instance.
var data = [1,2,opc$tag] Java.to(data, "int[]")	Build an array from a primitive tag.
Java.to(Java.from(opc$arraytag).slice(0,2), "int[]")	Return a part of an array.

var value = {
    employees": {
        "name": "steve",
         "value": opc$tag
  }
}
value
	Build a complex value (JSON) from primitives.

var retRow;
Java.from(csv$data).forEach(function (item, index) {
  if(parseInt(item["07A7"]) === 916){
     retRow = item;
   }
});
retRow;
	Iterate an array of complex data (CSV in this example) and return the row that meets a given condition.

Default Values

A default value is a static value that forms the basis for adding context or metadata when used without an expression or setting any initial value when used with an expression that may not be able to be evaluated under all conditions.

Some examples include:

1. A default could be set to *F* to indicate the units of temperature for the modeled instance, where units are not accessible from any other source and no expression can be set.
2. A default could be set to *Undefined* to indicate that until real-time data can be set by an expression, the state of the model *Instance*'s attribute is undefined.

Appendix C – REST Output Templates

The REST output supports templates, which allow control over the REST payload. The template engine uses Apache FreeMarker. HighByte Intelligence Hub exposes the following API to Apache FreeMarker:

- ${value}
  - The value being written out. On it's on, this evaluates to the hub's default JSON output
- ${values}
  - Used in cases where the output is an array of values versus a single object. Like ${value} this by default evaluates to the default JSON output
- ${value.attribute_name}
  - Index into a model to get the value for 'attribute_name'. As an example, if a model pressMachine has a cycleCount attribute, this would be ${value.cycleCount} to access that value
- ${value.name}
  - Returns the name of the model instance or alias
- ${value.quality}
  - Returns the quality as a string, either Good or Bad
- ${value.time}
  - Returns the timestamp as an epoch (UTC)
- ${value.type}
  - Returns the model name
- ${value.elements}
  - Returns a list of key value pairs, where key is the attribute name and value is the attribute value
- ${value.array}
  - Returns an array, to use FreeMarker array operations

Note the syntax can be chained together. So if "press" has a child model "motor" with an attribute "amps", you can access amps using ${value.motor.amps}.

With this API, here are some example use cases. Please see the FreeMarker documentation online for FreeMarker specific syntax.

```
<#-- Decorate payload (add around payload) -->

{

"index": "myindex",

"source": "mysource",

"payload": [

        <#list values as value>

                ${value}<#if value_has_next>,</#if>
```

```
    </#list>
    ]
}

<#-- Rename a field -->
{
"payload": [
    <#list values as value>
        ${value?replace("_name", "name")?replace("_model",
"model")?replace("_timestamp", "timestamp")}<#if value_has_next>,</#if>
    </#list>
    ]
}

<#-- XML -->
<root>
    <values>
    <#list values as value>
     <value>
      <#list value.elements as name, v>
            <#if name != "motor">
            <${name}>${v.string}</${name}>
            </#if>
            </#list>
            <motor>
            <#list value.motor.elements as name, v>
                <${name}>${v.string}</${name}>
            </#list>
            </motor>
            </value>
        </#list>
        </values>
</root>

<#-- Values in OPC format -->
{
"timestamp": ${.now?long},
"values": [
        <#list values as value>
                {
                "id": "${value.name}",
                "v": ${value},
                "q": ${value.quality},
                "t": ${value.time}
              }<#if value_has_next>,</#if>
        </#list>
        ]
}

<#-- Values in influxdb format -->
${value.type},name=${value.name} <#list value.elements as name, v>${name}=${v}<#if name_has_next>,</#if></#list> ${value.time}

<#-- Transform time -->
{
"values": [
        <#list values as value>
        {
        "id": "${value.name}",
        "v": ${value},
        "q": ${value.quality},
        "t": "${value.time?number_to_datetime?iso_utc}"
     }<#if value_has_next>,</#if>
        </#list>
        ]
}
```

Appendix D – OSIsoft PI AF SDK Connector

The OSIsoft PI AF SDK Connector uses the OSIsoft AF SDK to communicate with the PI and AF Server. The agent is a .NET application and is delivered separately from the core HighByte Intelligence Hub installation. The requirements for the agent are as follows.

1. Windows Operating System
2. The OSIsoft AF SDK/AF Client installed on the system running the agent
3. Microsoft .NET 4.8 or greater

The agent runs as a standalone application, by running the *intelligencehub-osisoft.exe* OR as a service. When running standalone, it's recommended to run the executable with admin level permissions. To install as a service, run the following command. InstallUtil is found in the .NET installation directory (%windir%\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe).

```
installutil intelligencehub-osisoft.exe
```

The agent connects to the AF SDK/AF Client and opens a port to receive requests from the HighByte Intelligence Hub instance. By default this communication occurs using secure websockets. On the first run the agent creates a self-signed certificate called intelligencehub.pfx as well as a settings.json file. The file contains the following.

```
{

"port":45290,

"token":"randomtoken"

}
```

The port controls the port that the agent listens on. This port must be open and available to the Intelligence Hub connecting to the agent. The token is a random security token created by the agent. This token must be applied to the Intelligence Hub OSIsoft PI AF SDK connection settings to authenticate the hub. Any changes to these settings require restarting the agent.

In the Intelligence Hub OSIsoft PI AF SDK connection settings, enter the IP/port of the agent, as well as the token. The agent also has a local intelligencehub.log file for debug information.

Note the agent connects to the default PI Archive and Asset Framework system configured by the AF Client using the PI System Explorer.

Appendix E – HTTP Server

The Intelligence Hub configuration can be hosted using a 3rd party web server. To do this, follow the instructions below.

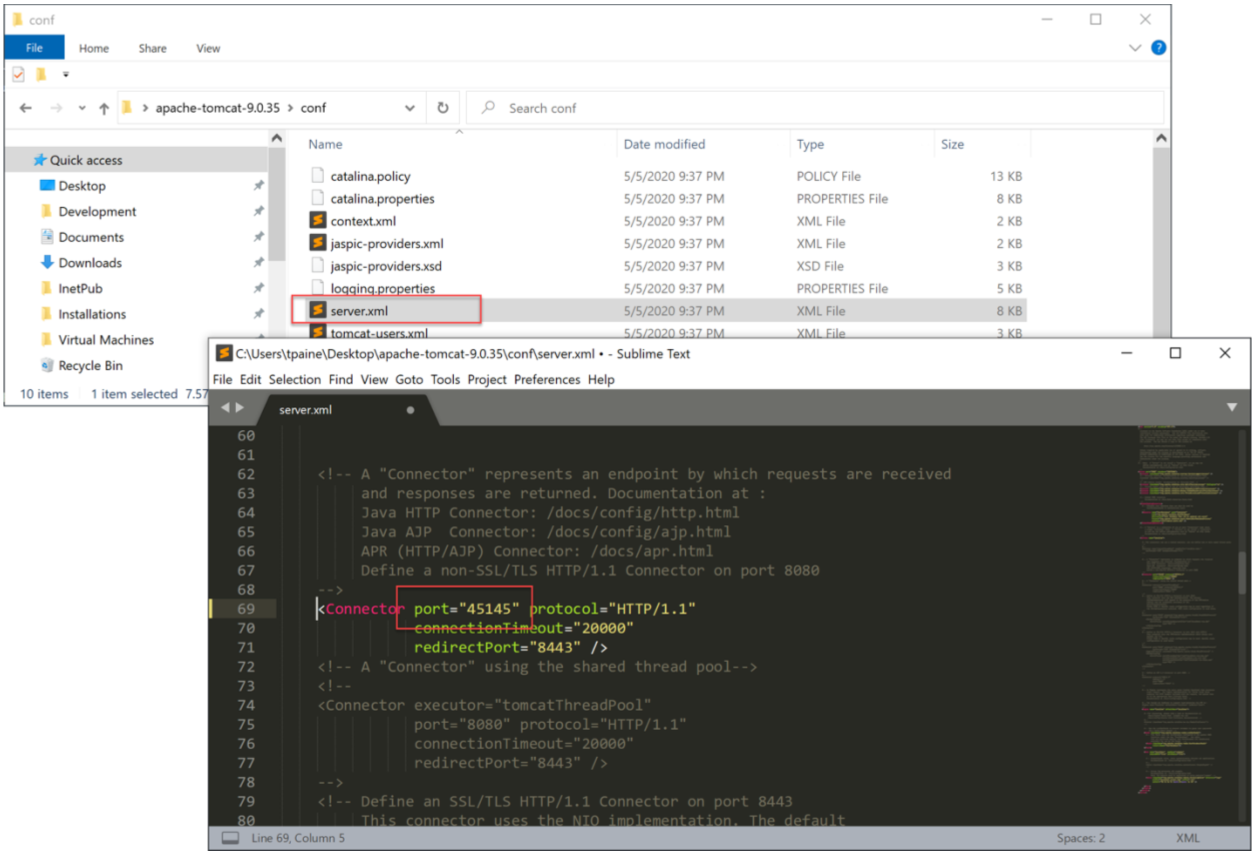
Check out the HighByte YouTube Channel for video tutorials on how to install the Intelligence Hub and any required dependencies on Windows 10 and Ubuntu Linux.

Step 1. Install Tomcat

You may also choose to host the configuration from any web server. Though you should be able to use any HTTP server, basic guidance on getting up and running with Tomcat 9 is provided.

Step 1. Install and Setup Tomcat

A. Download Tomcat 9 from the web. Currently available at Apache Tomcat Downloads. Select the binary distribution that is relevant to your OS (one of the compressed .zip or tar.gz should work).
B. Extract the compressed file to a location where you have read/write/execute privileges to setup and run TomCat 9.
C. Open the *server.xml* to modify the HTTP port setting. This file is located in the *./conf subdirectory* where you extracted the compressed file (e.g., *apache-tomcat-9.0.35/conf/server.xml*).
D. Within the *server.*xml file, find the string *<Connector port="8080" protocol="HTTP/1.1”*. To change the port, replace *8080* with the port you would like to use (e.g., *45145* will be used throughout this document). See Figure 45.
E. Consult your IT department and the TomCat 9 documentation to ensure that your HTTP server setup conforms to your organization’s IT policies.



Name	Date modified	Type	Size
catalina.policy	5/5/2020 9:37 PM	POLICY File	13 KB
catalina.properties	5/5/2020 9:37 PM	PROPERTIES File	8 KB
context.xml	5/5/2020 9:37 PM	XML File	2 KB
jaspic-providers.xml	5/5/2020 9:37 PM	XML File	2 KB
jaspic-providers.xsd	5/5/2020 9:37 PM	XSD File	3 KB
logging.properties	5/5/2020 9:37 PM	PROPERTIES File	5 KB
server.xml	5/5/2020 9:37 PM	XML File	8 KB
tomcat-users.xml	5/5/2020 9:37 PM	XML File	3 KB

```


    <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="45145" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    -->
    <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
         This connector uses the NIO implementation. The default
```

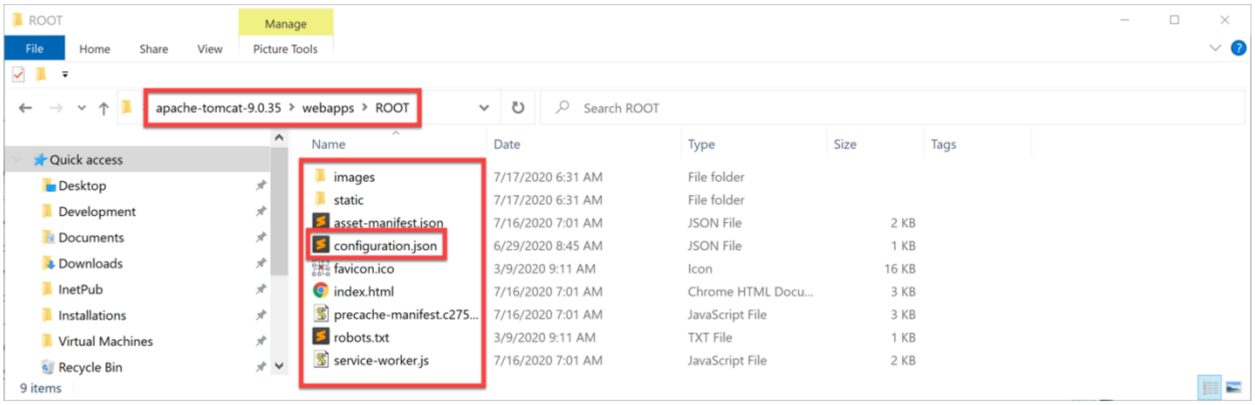
Figure 45

A. Extract *HighByte-Intelligence-Hub-2.x* (provided with this documentation) to a location where you have read/write privileges.

B. Copy the extracted contents of *HighByte-Intelligence-Hub-2.x/configuration* to a location that is recognizable to your HTTP server.

   For TomCat 9, this will be installed in the ROOT for simplicity. Go to the directory called *ROOT* under the *webapps* subdirectory of where you installed the HTTP server (e.g., *apache-tomcat-9.0.35/webapps/ROOT*) and delete any files installed by default. Next, copy the contents of *HighByte-Intelligence-Hub-1.x/configuration* to this *ROOT* directory. See Figure 46.

Step 2. Install and Start Configuration



Name	Date	Type	Size	Tags
images	7/17/2020 6:31 AM	File folder		
static	7/17/2020 6:31 AM	File folder		
asset-manifest.json	7/16/2020 7:01 AM	JSON File	2 KB	
configuration.json	6/29/2020 8:45 AM	JSON File	1 KB	
favicon.ico	3/9/2020 9:11 AM	Icon	16 KB	
index.html	7/16/2020 7:01 AM	Chrome HTML Docu...	3 KB	
precache-manifest.c275...	7/16/2020 7:01 AM	JavaScript File	3 KB	
robots.txt	3/9/2020 9:11 AM	TXT File	1 KB	
service-worker.js	7/16/2020 7:01 AM	JavaScript File	2 KB	

Figure 46

C. Edit the *configuration.json* file to set the *apiBaseUrl* to the *runtime configuration API endpoint*. If the file does not exist, copy or rename *configuration.json.template* to *configuration-settings.json*.

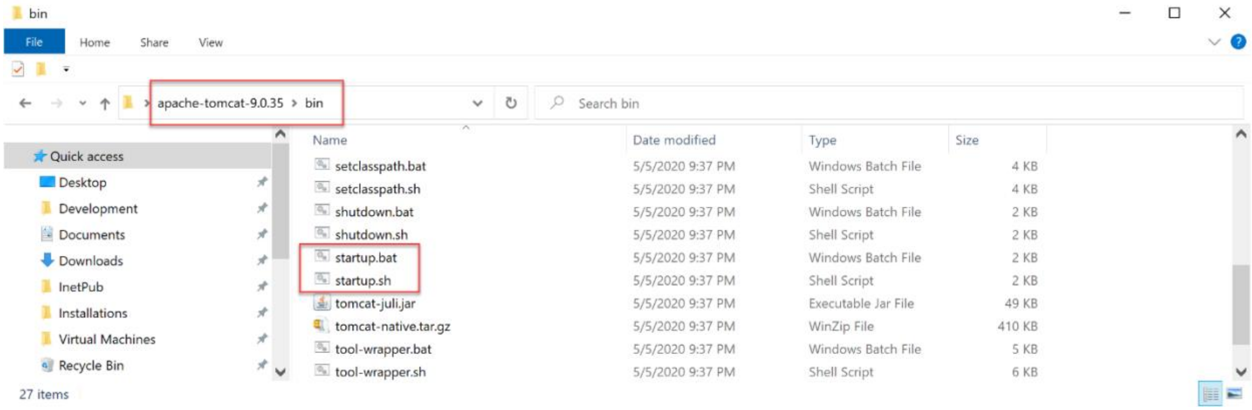
By *default*, the endpoint is *http://localhost:45245*, which constrains configuration access to web browsers running on the same machine as the runtime.

To configure the product from a remote machine, change *localhost* to the hostname or IP address of the machine where the runtime is executing (e.g., http://mymachine:45245, http://192.168.1.100:45245).

Save the file when finished. You must restart the HTTP server any time changes are made to this file.

D. Start the HTTP server.

For TomCat 9, go to the *bin* subdirectory of where you installed the HTTP server (e.g., *apache-tomcat-9.0.35/bin*) and *execute* the *startup* file associated with your OS (e.g., startup.bat, startup.sh). See Figure 47.



Name	Date modified	Type	Size
setclasspath.bat	5/5/2020 9:37 PM	Windows Batch File	4 KB
setclasspath.sh	5/5/2020 9:37 PM	Shell Script	4 KB
shutdown.bat	5/5/2020 9:37 PM	Windows Batch File	2 KB
shutdown.sh	5/5/2020 9:37 PM	Shell Script	2 KB
startup.bat	5/5/2020 9:37 PM	Windows Batch File	2 KB
startup.sh	5/5/2020 9:37 PM	Shell Script	2 KB
tomcat-juli.jar	5/5/2020 9:37 PM	Executable Jar File	49 KB
tomcat-native.tar.gz	5/5/2020 9:37 PM	WinZip File	410 KB
tool-wrapper.bat	5/5/2020 9:37 PM	Windows Batch File	5 KB
tool-wrapper.sh	5/5/2020 9:37 PM	Shell Script	6 KB

Figure 47

E. Display the configuration by launching a Web Browser and connecting to *http://localhost:45245* (assuming you set the HTTP port to *45245* under *Required Dependencies* | *HTTP Server* above). See Figure 48.

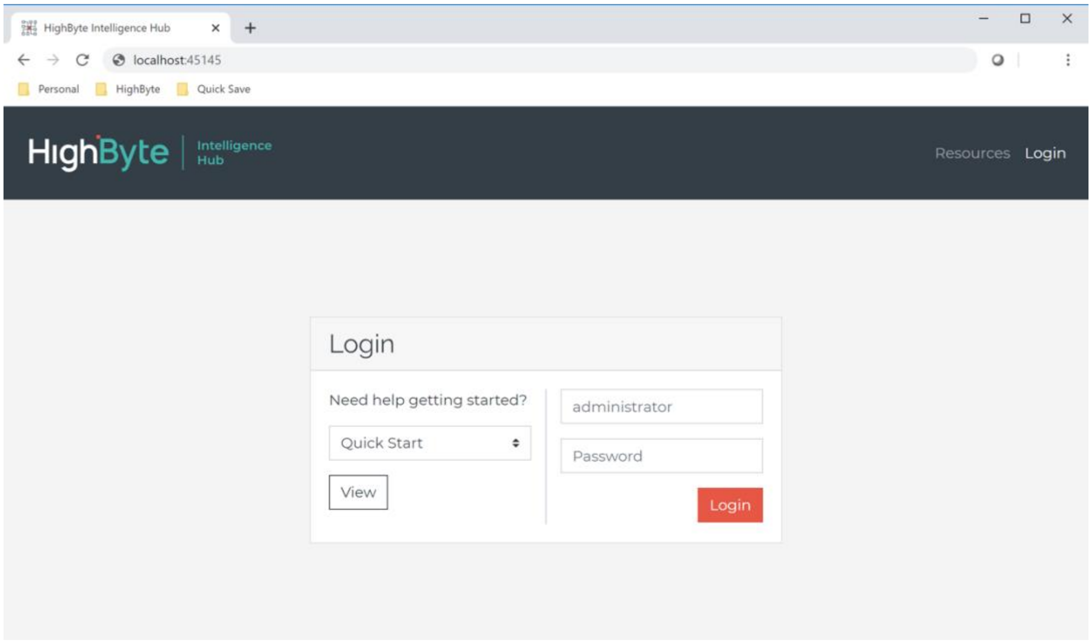


Figure 48