MQTT Connector User Guide

Carrier ©2025 · Catalog No. 11-808-930-01 · 8/27/2025

 Verify that you have the most current version of this document from **www.hvacpartners.com**, the **Carrier Partner Community** website, or your local Carrier office.

Important changes are listed in **Document revision history** at the end of this document.

Carrier© 2025. All rights reserved.

The content of this guide is furnished for informational use only and is subject to change without notice. Carrier assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Contents

What is the MQTT add-on?

The MQTT connector add-on enables the i-Vu® system to act as an MQTT client to publish data from the controllers in the building automation system to an MQTT broker.

Licensing

The MQTT connector add-on uses a non-expiring single purchase license. This license is valid for single i-Vu® serial number only, and is purchasable on the Carrier® e-store. A valid license is required to establish a persistent connection to 3rd party MQTT brokers. However, configuration, test connections to brokers, and connections to Carrier's Abound™ platform can function without a license.

Basic MQTT concepts

Brokers, clients, publication, and subscription

MQTT is a popular transport protocol for internet of things (IoT) applications. It is a "server-mediated" protocol, meaning that a server (**broker**) is at the center of all communication between devices and applications (**clients**). MQTT clients can "publish" data to the broker and "subscribe" to data published by other clients.

Topics

Clients publish their data to "topics". Topics allow data to be organized inside the broker hierarchically. This system of organization is similar to a file system where folders may contain other folders. MQTT messages are published to topics in the same way that files are saved to folders.

Clients can subscribe to topics and will receive automatic notifications sent by the broker when data is updated under the topics they are subscribed to.

In MQTT, the organization of topics is arbitrary and typically coordinated by parties using the data. For example, a zone temperature at a university campus might be published to any of the following topics depending on how the university's IT staff chose to organize their topics:

- bldg1/floor1/room1
- hvac/car/router/device/equipment
- bldg1/zonetemps

In contrast, the MQTT connector add-on conforms to the Sparkplug B™ standard for MQTT communication. This means that the add-on uses a strict topic structure that cannot be modified. The MQTT add-on will always publish a zone temperature to the topic "**spBv1.0/group_Id/edge_node_Id/device_Id"** where group_id, edge_node_id, and device_id are defined by the Sparkplug B™ specification. See *Unique Identifiers and add-on data* for details.

©2025 Carrier.
All rights reserved.

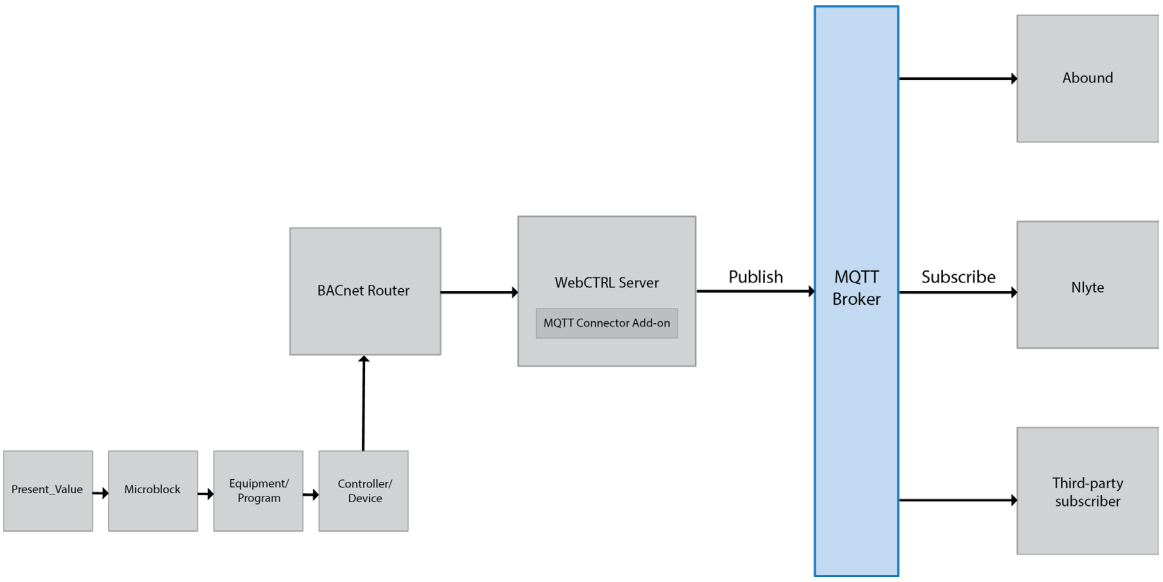

Figure 1

i-Vu® Data

Unlike traditional point-to-point integration, data is published from the MQTT add-on by defining the types of data to be shared. This is accomplished using a feature called "publication scopes" to determine which microblocks should have their **present_value** property (and metadata, such as units and semantic tags) published to one or more MQTT brokers. See *Publication scopes* for more information.

Microblocks

Once the MQTT connector add-on has identified which microblocks to publish to the MQTT broker, it publishes the **present_value** property (and the associated metadata) of that microblock. i-Vu®'s microblocks can contain other properties, but the MQTT connector is only capable of publishing the **present_value** property. As a result, there is additional information within microblocks that cannot be published by this add-on.

Some microblocks do not have a **present_value** property so they are unable to have their data published by the MQTT add-on. Some examples of microblocks without a present_value property are non-BACnet setpoints, logical comparators (and, or, greater than, etc.), and math microblocks (add, multiply, etc.). BACnet Setpoint microblocks can have their data published. See the following paragraph for more information.

There are some complex microblocks that contain sub-microblocks, which each have their own present_value property. Sub-microblocks can be viewed by expanding the parent microblock in the i-Vu® **Geographic** tree. See Figure 2. Sub-microblocks can be tagged by creating rules in i-Vu®'s semantic tagging engine. The semantic tagging engine references the parent microblock and sub-microblock separated by a forward slash. For example, a rule of the type "Reference Name Matches" with the name **setpt/occupied_cool_setpoint** tags the occupied cooling setpoint within all setpoint microblocks with the reference name "setpt". This allows for a general reference of multiple microblocks across a system without tagging each one individually. See the *i-Vu® Help* for more information about semantic tagging.

©2025 Carrier.
All rights reserved.

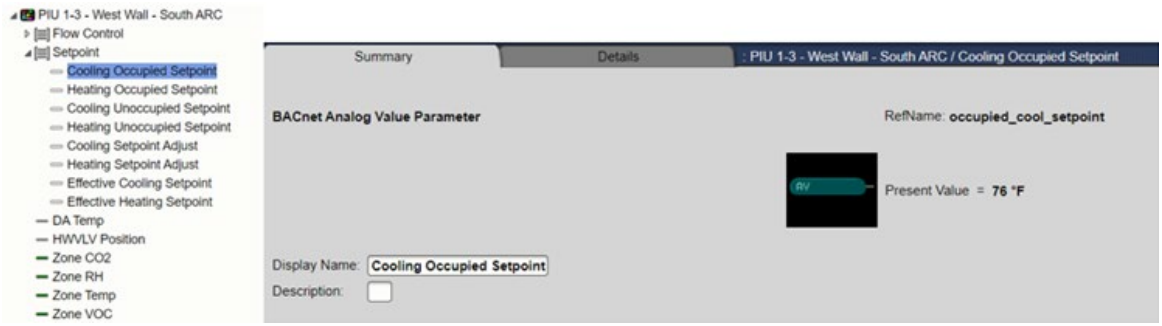

Figure 2

Data collection

The MQTT connector add-on collects the **present_value** of microblocks by polling controllers periodically (every 30 seconds by default, adjustable from the add-on **Settings** page) and comparing the current value to the last value read. If the current value is different than the last value, the current value is published to the appropriate topic on the MQTT broker.

Requirements

i-Vu® and add-on

- i-Vu® Pro v8.0 cumulative patch 5 (20221114-0) or greater and the standalone "IV80_System_Service_API" update.
- A valid license if connecting to an MQTT broker other than Abound™
- A i-Vu® operator account with the Engineer System privilege (to edit configuration; All users can see the add-on and view configurations)

NOTE This add-on is designed to run without a license. Configuration, test connections to brokers, and full connections to Carrier's Abound™ platform will all work without a license. Only persistent connections to non-Abound™ MQTT brokers requires a license.

Infrastructure

Identify a target MQTT cloud broker or local server running MQTT broker software supporting MQTT v3.1.1 which will be published to a server running MQTT broker software supporting MQTT v3.1.1.

NOTE The broker is often provided by the customer and is included with Abound™. In these cases, the information required to create a connection to the broker is provided by a third party.

  - Connection information including address and port
  - Authentication information. See Authentication and certificates for more information.

- Connectivity from the i-Vu® server to the MQTT broker. This may require customer IT support to allow traffic from specific ports and protocols.

©2025 Carrier.
All rights reserved.

Before you begin

All microblocks that will have their data published must be tagged through i-Vu®'s semantic tagging engine. See the *What is the MQTT add-on?*, *i-Vu® data*, and *Publication scope*s sections of this document.

©2025 Carrier.
All rights reserved.

Certificates

There are two types of certificates used by the MQTT connector add-on: Client certificates and trusted certificates. Client certificates are used for client certificate authentication and trusted certificates are used to validate the identity of servers with untrusted certificates.

NOTE If neither of these features are required, you can safely skip this section.

This add-on allows the following types of certificates to be added during the configuration of a broker. For instructions, see the Add a broker section of this document.

- **Client certificates** - Used by the broker to authenticate the MQTT connector add-on when client certificate authentication is in use.
- **Trusted certificates** - Used by the MQTT connector add-on to validate that the broker is a trusted server. A TLS connection cannot be established unless the broker's certificate is trusted. If the issuer of the broker's certificate is not in the add-on's list of trusted certificate authorities, such as if the broker is using a self-signed certificate, the broker's certificate or certificate of the issuing authority may be added as a trusted certificate using the instructions in the following section.

©2025 Carrier.
All rights reserved.

Brokers

The MQTT connector add-on publishes i-Vu® data to a server known as the MQTT broker. For more information about MQTT brokers see *Basic MQTT concepts* page 1.

NOTE This add-on features an automatic reconnect function which will attempt to reconnect the broker if an unexpected disconnection occurs.

Add a broker

1. Click on the **Brokers** tab in the left pane of the add-on.
2. Select **Add New** to create a new broker to begin configuration.
3. Enter a **Broker Name**.
4. Enter an **Address**. This is the hostname, or IP address, of the MQTT broker you are connecting to without the port or protocol specified.

   NOTE If using TLS, this address must match the CN of the trusted certificate.
5. The **Use WebSocket** option enables WebSocket for the connection. The default port number used when WebSocket is enabled is 443.
6. The **Use TLS** option is used to enable or disable TLS encryption.

   NOTE Leaving the TLS checkbox unchecked, allows information to be sent to the broker unencrypted. This is an insecure and potentially dangerous configuration.
7. If using a custom **Port** number for the connection, check the **Custom Port** box and enter the port number.
8. Use **Refresh Connection**. If required, upload a **Trusted Certificate**. This certificate is added to the add-on's trust store. Trusted certificates are required if using TLS and if the broker's certificate is not issued by a publicly trusted certificate authority. The certificate file must be in PEM format.
9. Select the authentication method. Most systems use standard authentication. JWT is generally used for Google IoT brokers.
10. If required by the broker, enter a **Username** and **Password**. If the broker does not use username and password authentication, leave these fields blank. JWT instead requires an audience claim, generally provided by your broker hosting service.
11. If the broker requires client certificate authentication, click **Browse** to upload your certificate and private key for the TLS handshake. These files are optional, but some brokers require a specific client certificate for authentication. The certificate file should only contain the client certificate. The certificate and key files must both be in PEM format. See the *Certificates* section of this document for more information.

    NOTES

    - Each of the two files must be in PEM format. If they are not in PEM format, follow the instructions in *Extracting a key file and certificate from a keystore (such as .pfx or .p12)* section of this document.
    - The following options relate to unique identifiers within the add-on. For more information, please see the *Unique identifiers* sections of this document.

©2025 Carrier.
All rights reserved.

12 Use the **Client ID** option to choose the System Generated UUID, the i-Vu® Serial number, or a custom entry to be used for Sparkplug B™ client id. Leave this option set to *System Generated* unless you are sure it should be set otherwise.

13 Select the protocol specification to use:

- **Sparkplug B** – Publish data conforming to Sparkplug B specification
- **Sparkplug B w/ Virtual-EON** – Conforms to Sparkplug B specification with the additional option of utilizing software-based Edge of Network (EON) nodes. This can be used to organize devices in topic structures.
- **Telemetry Only** – Publish telemetry only payloads which is formatted based on the custom payload configuration file upload. No Sparkplug B specifications are used in this configuration.

NOTE "`car`" is the **Group ID** expected by Abound™.

14 The **Use CoV** option is used to enable or disable Change of Value (CoV). When CoV is enabled, data will only be published to the MQTT broker when the value has changed since the last publish. Leaving this option enabled reduces the load on the BAS server and MQTT broker.

15 **Use Heartbeat** publishes all at the user set interval. This setting works simultaneously with the data publishing at "change of value" (COV) intervals. **Use COV** must be enabled when **Use Heartbeat** is enabled.

16 If you have been asked to use a specific Sparkplug B™ group ID, enter it in the **Group ID** field. If not, leave this field blank to use "`car`" as the **Group ID**.

NOTE "`car`" is the **Group ID** expected by Abound™.

17 **Equipment Reference Type** sets the type of identifier to use in payloads. Select **UUID, Equipment Display Name**, or **Equipment Reference Name**. UUID is universally unique but display names and even reference names may not be. If a duplicate exists, data publishes to the same topic, so ensure display names and reference names of equipment set to publish are unique.

18 Use the **Edge of Network/Node ID** option to choose between a System Generated UUID, the WebCTRL serial number, or a custom entry to be used with **Sparkplug B Edge of Network ID.** Leave this option set to *System Generated* unless you are sure it should be set otherwise.

If **Sparkplug B + Virtual-EON** is selected, this option changes to “Node ID”. In this configuration, there could potentially be many edge-of-network nodes, so this option specifies the node which will handle certain lifecycle messages, including NBirth’s, receiving NDeath’s from the broker, and subscribing to NCMD.

NOTE The default System Generated UUID for the Edge of Network option will be the same of that of the default system generated client UUID.

19 Use the **Test Connect** button to verify that the configuration allows connectivity to the broker.

20 Click **Add** to save this configuration and add this broker to the list of configured brokers.

NOTE After a broker is added, it can be edited by selecting that broker on the **Broker** tab.

Broker details page

Select a broker from the main broker menu to navigate to the broker details page, which displays information about the broker settings and connection. The following connection information is available on the **Broker Details** page.

©2025 Carrier.
All rights reserved.

- Broker URL
- WebSocket status
- Encryption status
- Change of Value status
- Heartbeat settings
- Authentication status
- Certificate expiration date
- Client ID in use

Sparkplug B™ specific information, such as the NCMD topic, group ID, and Edge of Network ID is also located on this page. The broker can be connected or disconnected and the **Edit** menu may be accessed from this page.

NOTE The **Edit** menu is not accessible while the broker is in a connected state.

©2025 Carrier.
All rights reserved.

Publication scopes

To publish data to an MQTT broker, the add-on uses the i-Vu® semantic tagging engine and its ability to apply tags to microblocks and equipment. The semantic tagging engine applies meaning to locations in the system through tagging. These tags are used to create "publication scopes", which define what types microblocks have their present_value published to the MQTT broker. For this reason, i-Vu® semantic tagging engine must be used to tag every microblock to be published before a publication scope can be used to successfully search for them. For more information on applying semantic tags in i-Vu®, see the "Semantic Tagging" section of the *i-Vu® help*.

NOTE This add-on will operate more efficiently if fewer publication scopes are present. Fewer complex publication scopes should be favored over creating many simple publication scopes.

Creating a publication scope

1. Click the **Scopes** tab in the left pane of the add-on.
2. Click **Add New** to create a new publication scope.
3. Enter a **Publication Scope Name**.

   NOTE This will be used for display and description purposes.
4. Check the box next to the location(s) to have their data published. (The **Location** tab should be displayed first) Microblocks at, or below, the selected locations are eligible for publication.

   NOTE Leaving no locations selected will effectively disable the publication scope.
5. Click the **Point Types** tab. Point definitions are collections of tags that define a type of point in the BMS. A publication scope may contain more than one point definition. Each point definition is listed as row on the **Point Types** tab.

   Tags in a **Point Definition** have an 'and' relationship and point definitions (rows) have an 'or' relationship with each other. A point from the BMS will be published if ALL the tags in at least one point definition (row) are assigned to that point in the BMS. If no tags are selected, no points are published.

   a) The first row in the definitions list is selected by default.

   b) Add tags to the selected definition row by clicking on them in the **Tags** column of the table at the right.

   NOTES

   - If the tags you want to add are present in an i-Vu® semantic rule, all tags may be added at once by navigating to the **Rules** tab and clicking the "+" button next to the rule.
   - The list of tags or rules may be filtered by typing a tag name in the **Search** field above the Tags or Rules table.

   c) A tag may be removed from the selected definition row by clicking the "X" icon on the tag.

   d) Add a new definition row by clicking the **Add** button above the definition rows.

   e) Switch to editing a different definition row by clicking on it.

   f) Delete definition rows by clicking the **Delete** button above the definition rows.

©2025 Carrier.
All rights reserved.

6 Click the **Equipment Types** tab.

Tags in an **Equipment Definition** have an 'and' relationship and equipment definitions (rows) have an 'or' relationship with each other. Points in an equipment in the BMS are eligible for publishing if ALL the tags in at least on equipment definition (row) are assigned to that equipment in the BMS. If no tags are selected, points are filtered by their equipment.

a) The first row in the definitions list is selected by default.

b) Add tags to the selected definition row by clicking on them in the Tags column of the table at the right.

NOTES

- If the tags you want to add are present in an i-Vu® semantic rule, all tags may be added at once by navigating to the **Rules** tab and clicking the "+" button next to the rule.
- The list of tags or rules may be filtered by typing a tag name in the **Search** field above the *Tags or Rules table*.

c) A tag may be removed from the selected definition row by clicking the "X" icon on the tag.

d) Add a new definition row by clicking the **Add** button above the definition rows.

e) Switch to editing a different definition row by clicking on it.

f) Delete definition rows by clicking the **Delete** button above the definition rows.

7 At any time during the configuration process, click **Add** to save the configuration. After saving, the **Add** button is replaced with a **Save** button, but functions identically. Clicking **Cancel** at any time discards any unsaved changes and navigate back to the **Publication Scopes** page.

NOTE Saving or editing publication scopes will not automatically cause new points to be published, or newly out-of-scope points to stop publishing. To update what is published, use the **Apply All Scopes** button, which refreshes all publication scopes. Note that publication stops during this process. Additionally, scopes will be refreshed upon start-up of the add-on.

©2025 Carrier.
All rights reserved.

Scan

Shows system items to be published based on the publication scope selections, which displays in a tree format, similar to the geographic tree in i-Vu®. The areas in the tree display the number of equipment found associated with a scope, the number of offline equipment (unable to be published) in parenthesis, and the number of points across each equipment that are set to publish. Areas may be expanded to view the itemized information for each equipment node underneath. Areas and equipment that do not have any points associated with a publication scope are gray and unable to be selected.

If Sparkplug B is used for publication, the DBIRTH message associated with each equipment is available to view by clicking the "eye" icon that appears in line with the equipment. In a telemetry only configuration, a separate option will be present to display the payload example and download a .csv report of the scan results for the telemetry only format.

©2025 Carrier.
All rights reserved.

Settings

This tab contains add-on system settings.

Setting	Description
Daily Alarm Time	This field sets time of day that alarms will be generated. Valid range of 0-23 (24-hour clock).
Certificate Warning Days	This field sets the number of days before a certificate expiration alarm warning is generated. The alarm is sent once daily, but only for certificates associated with an active broker connection. The default setting is 30 days.
Swagger UI	A link to the Swagger user interface RESTful API is provided. See MQTT Connector API section for more details.
Custom Payload	Upload a custom payload. A custom payload will be used when a telemetry only client type is selected in a broker configuration. The custom payload file must be in json format. See the *Appendix* (page 22) below for further configuration information.
Sample Rate	The time, in seconds, between each poll for data points being published. The default setting is 30 seconds. See Data Collection for more information.
Initial Read Delay	This is the delay, in seconds, between when data is cached from i-Vu® during publication scope initialization (typically on add-on startup) and when the data is published to the broker. The default setting is 1 second

©2025 Carrier.
All rights reserved.

Alarms

An alarming function is built in to the MQTT connector add-on, which alerts the user of certain conditions in the add-on. The alarms appear as "System Info" alarm in the i-Vu® alarms page interface. There are two types of alarm conditions:

Alarm	Description
Certificate Expiration	An alarm is generated when a certificate is 30 days or less from its expiration date. After the initial alarm, subsequent alarms are generated daily until either the certificate expires, or a new certificate is uploaded.
Broker Disconnect/Reconnect	An alarm is generated if a broker is connected and becomes disconnected unexpectedly. An alarm is also generated when the automatic reconnect attempt is successful. the automatic reconnect attempt is successful, an alarm is generated. An alarm is not generated when a user attempts to connect a broker from the user interface.

TIP Consider setting up email alarm actions for system info alarms to reduce the chance of missing license and certificate expiration alarms.

©2025 Carrier.
All rights reserved.

Virtual Edge of Network (VEON)

Users have the option to implement virtual edge of network nodes (veon) if required. Virtual edge of network nodes act as normal edge of network nodes, so sparkplug B standards such as lifecycle messages apply. In this configuration, there could potentially be many virtual edge of network nodes, so the user must specify which node will handle certain lifecycle messages, including NBirth's, receiving NDeath's from the broker, and subscribing to NCMD messages.

To implement virtual edge of network nodes, semantic tagging must be properly applied. Devices assigned as a virtual edge of network *must* specifically have the "veon" value tag assigned through custom semantic rules. The value for this tag can then be set at the equipment in i-Vu® to whatever is required by the consumer of the data. For example, the tag might be applied to a UPS in a datacenter, so the value tag may be "veon = UPS".

As with other changes to tags and identifiers in i-Vu® if a veon tag is changed or added after a broker and scope has already been configured, the scope must be re-applied to scan in the changes.

If Sparkplug B w/ VEON is selected as the protocol, and no veon tag is set, then the "null" value will be used instead.

©2025 Carrier.
All rights reserved.

Custom Payload (Telemetry Only)

The MQTT add-on can be configured with a publication specification alternative to Sparkplug B. This is a format which follows a json template standard (see Appendix) but is customizable within that template. Sparkplug B specifications such as lifecycle messages are not generated, and payloads are sent at QOS 0 in this mode.

©2025 Carrier.
All rights reserved.

Protocol details

This add-on supports the following protocols:

- MQTT 3.3.1
- Sparkplug B™ 1.0

©2025 Carrier.
All rights reserved.

Unique identifiers and add-on data

Unique Identifiers

The MQTT connector add-on uses four types of unique identifiers that differ from those used in i-Vu®:

Unique Identifiers	Description
client_id	Client ID is a unique identifier defined by MQTT. This is the identifier for a given client (in this case, the add-on) and is used by the broker for access control and subscription mapping. For this reason, it is important to keep the same client ID once it has been established. **NOTE** MQTT 3 specified 23-character client IDs. MQTT 3.1.1 increased the limit to 64k characters, but some brokers only support client IDs up to 23 characters. Character set may also be limited. This add-on does not limit its client ID to the version 3 specifications, so you may need to manually define a client ID if the broker does not support the default value generated by the add-on. **NOTE** Duplicate client IDs can cause broker connection issues. If using a manually set client ID, be careful not create a duplicate ID on a broker.
device_id	Device ID is a Sparkplug B™ unique identifier and is used to uniquely identify devices (in the case of this add-on, devices are i-Vu® equipment nodes). This add-on generates a universal unique identifier (UUID) for each equipment node that has microblocks published and then maps them to equipment nodes within i-Vu®'s core database. These UUIDs are used as the Sparkplug B™ device ID.
edge_node_id	An edge node ID is a Sparkplug B™ unique identifier and is used to uniquely identify a gateway that is publishing information on behalf of one or more devices (called an edge-of-network node). The broker connection configuration provides an option to use an automatically generated UUID (default) or serial number (this will use the serial number of i-Vu® license).
group_id	A group ID is a Sparkplug B™ unique identifier for grouping edge-of-network nodes. This is often used by consuming applications (such as Abound) to group edge-of-network nodes with similar payload formats and identify which "data adapter" to use when interpreting that data. **NOTE** Abound currently requires that i-Vu® use a group ID of "`car`" which is the default value for this add-on.

Add-on data

This add-on takes care to preserve unique identifiers. All unique ID's are stored in the add-on's data directories. If all data is deleted from the add-on, these IDs will be lost. For example, this occurs if the "Remove Add-on and Data" action on the i-Vu® Add-on page is used.

©2025 Carrier.
All rights reserved.

Loss of these unique identifiers can cause issues for broker access and can cause the applications consuming the data published by this add-on to interpret that data as coming from a different source. For these reasons, care should be taken in maintaining the add-on data across upgrades and re-installs.

TIP After configuring the add-on, use the "Save Data" option on the i-Vu® Add-on page to create a backup of the add-on configuration data. Store this for safe- keeping in the event of unexpected server data loss.

©2025 Carrier.
All rights reserved.

Tips and best practices

Extracting a key file and certificate from a keystore (such as .pfx or .p12)

To use client certificate authentication the MQTT connector add-on needs a certificate file and an associated private key. Each of the two files must be in PEM format.

Other file formats are sometimes used to transmit a client certificate and its associated private key. Some of these formats even combine the two into a single file. Fortunately, these files can be separated and converted into the correct PEM format by a free application called KeyStore Explorer. Instructions to extract the appropriate files from the common PKCS #12 keystore format (which can have the extensions .pfx or .p12) follow.

1. Install and open KeyStore Explorer.
2. Click **File**, and then **Open**.
3. Navigate to the keystore file and select it.
4. Click **Open**.
5. If the file is password protected, enter the password. Otherwise, leave the password field blank.
6. Click **OK**.
7. There should only be one row displayed in KeyStore Explorer. The first column of that row should contain an icon depicting two keys. This means that it is a keypair entry. Right-click this row and choose **Export > Export Private Key**.
8. If the keypair entry is locked, KeyStore Explorer will provide a prompt for a password.
9. Enter the password and click **OK**.
10. Choose **OpenSSL**.
11. Uncheck the **Encrypt** checkbox.

    NOTE It is not best practice to keep unencrypted private keys saved. Once this file is imported into the add-on, remove from disk immediately.
12. Leave the **PEM** checkbox checked.
13. Click **Browse** to select the location and filename for the private key file. Click Choose.
14. Click **Export**.
15. Right-click on the keypair entry row in KeyStore Explorer again. Now choose **Export > Export Certificate Chain**.
16. Leave the default settings (this should only be Head Only, X.509, and PEM).
17. Click **Browse** to select the location and filename for the certificate file. Click Choose.
18. Click **Export**.

©2025 Carrier.
All rights reserved.

MQTT Connector API

Some functions of the MQTT connector add-on are made available via an API called Swagger. A link to this API is provided in the settings page of the add-on. You can also access this from a web browser by navigating to `i-Vu® server address/mqtt-connector/dist/index.html` on the local i-Vu® server. From this page, you can explore the API and generate test calls and view the responses.

Test connect succeeds, but nothing is published to the broker

This behavior can occur when the MQTT broker is configured to restrict publication and subscription using an access control list (ACL). If the add-on's client ID is allowed to connect, but does not have permission to publish anything, the test connect will succeed, but the broker connection will fail and an EOF exception will be written to the logs.

In MQTT 3.1.1 (the version supported by this add-on), a broker will notify the client if the client is not allowed to connect, but will not notify the client if the client is not allowed to publish. This was fixed in MQTT 5, but that version has not been widely adopted.

NOTE The add-on must have permission to subscribe to its own topics. If it does not, it fails and writes an EOF exception in the logs.

Resending birth messages

There are cases when it is desirable to republish Sparkplug B NBIRTH and DBIRTH messages. These cases include troubleshooting and if a subscriber was not present for the issuance of the original birth messages. The add-on provides four ways to republish birth messages:

On Restart	The MQTT connector add-on always publishes its birth messages on all previously connected brokers when starting up.
On Connect	The MQTT connector add-on always publishes its birth messages when connecting to a broker.
Apply All Scopes Button	The **Apply All Scopes** button on the **Publication Scopes** page scans the i-Vu® system again for in-scope equipment and points then republishes the birth messages to all brokers.

©2025 Carrier.
All rights reserved.

Node Control/Rebirth Metric	In accordance with the Sparkplug B™ standard, the MQTT connector listens to the spBv1.0/<group_id>/NCMD/<eon_id> topic (where <group_id> and <eon_id> are the group ID and edge of network ID of the add-on for a given connection) and publishes a metric during its NBIRTH called Node Control/Rebirth with its values set to false.
	
	A subscriber may write to the spBv1.0/<group_id>/NCMD/<eon_id> topic and publish the Node Control/Rebirth metric with its value set to "true" to request the birth messages be sent again from the add-on. Upon receipt of this payload, the MQTT connector will stop sending data and republish its birth messages before resuming data publication.
	
	This method allows subscribers to initiate birth messages without any human interaction. The JSON interpretation of an example NCMD payload for requesting rebirth follows:
	
	`{`
	`  "timestamp": 1486144502122,`
	`  "metrics": [`
	`  {`
	`  "name": "Node Control/Rebirth",`
	`  "timestamp": 1486144502122,`
	`  "dataType": "Boolean",`
	`  "value": true`
	`  }`
	`  ]`
	`}`

©2025 Carrier.
All rights reserved.

Appendix

Sparkplug B Example Payloads

DBirth:

Topic: spBv1.0/[GROUP]/DBIRTH/[EON UUID]/[Device UUID]


```
{
  "timestamp": "1750950206337",
  "metrics": [
  {
    "name": "Identity/Program",
    "timestamp": 1750950206337,
    "dataType": "String",
    "isHistorical": false,
    "value": "variable_air_volume_terminal_unit"
  },
  {
    "name": "Identity/ReferenceNamePath",
    "timestamp": 1742926796831,
    "dataType": "String",
    "isHistorical": false,
    "value": "#bldg_1/#first_floor/#zone_1a/#zone_a1/#vav_1"
  },
  {
    "name": "Identity/DisplayNamePath",
    "timestamp": 1742926796831,
    "dataType": "String",
    "isHistorical": false,
    "value": "Bldg 1 / Floor 1 / Zone 1A / Zone A1 / VAV 1"
  },
  {
    "name": "Status/zone_temp",
    "timestamp": 1742926796831,
```


©2025 Carrier.
All rights reserved.

```
  "dataType": "Float",
  "isHistorical": false,
  "properties": {
    "DisplayName": {
      "type": "String",
      "value": "Zone Temp"
    },
    "SemanticTags": {
      "type": "PropertySet",
      "value": {
        "hvac": {
          "type": "String",
          "value": null
        },
        "temp": {
          "type": "String",
          "value": null
        },
        "zone": {
          "type": "String",
          "value": null
        },
        "air": {
          "type": "String",
          "value": null
        }
      }
    },
    "Units": {
      "type": "String",
      "value": "°F"
    }
  }
},
 {
   "name": "Status/isHeartbeat",
   "timestamp": "1750946830931",
```


©2025 Carrier.
All rights reserved.

```
      "datatype": 11,
      "isHistorical": false,
      "isNull": false,
      "properties": {},
      "booleanValue": false
    }
  ],
  "seq": 90
}
```


DData:

Topic: spBv1.0/[group ID]/DDATA/[EON UUID]/[device UUID]


```
{
  "timestamp": 1702396520356,
  "metricsList": [
    {
      "name": "Status/da_temp",
      "timestamp": 1702396520356,
      "datatype": 9,
      "isHistorical": false,
      "isNull": false,
      "properties": {
        "keysList": [],
        "valuesList": []
      },
      "floatValue": 70.5,
      "bytesValue": ""
    },
    {
      "name": "Status/flow",
      "timestamp": 1702396520356,
      "datatype": 9,
      "isHistorical": false,
      "isNull": false,
      "properties": {
        "keysList": [],
```


©2025 Carrier.
All rights reserved.

```
        "valuesList": []
      },
      "floatValue": 148.39405822753906,
      "bytesValue": ""
    },
   {
     "name": "Status/isHeartbeat",
     "timestamp": "1750946830931",
     "datatype": 11,
     "isHistorical": false,
     "isNull": false,
     "properties": {},
     "booleanValue": false
   }
  ],
  "seq": 90
}
```


Telemetry Only JSON Template

The custom payload format is defined as:


```
{
  "topic": "TOPIC",
  "data": {
    "tag": {
      "resolve": "TAG",
      "pattern": "TAG_PATTERN"
    }...
  }
}
```


©2025 Carrier.
All rights reserved.

An example of a custom payload is:

NOTE This is the input payload. This is what the add-on uses to find and format data from the BMS:

```
{
  "topic": "companyA/?{tag}/?{betterTag}/?{equipment}",
  "data": {
    "tag": {
      "resolve": "TAG",
      "pattern": "@zone.reference-id"
    },
    "betterTag": {
      "resolve": "TAG",
      "pattern": "@yearBuilt.value"
    },
    "bestTagOfAll": {
      "resolve": "TAG",
      "pattern": "evaporative"
    },
    "equipment": {
      "resolve": "GQL",
      "pattern": "~equipment.display-name"
    },
    "staticExample": {
      "resolve": "STATIC",
      "pattern": "I'm hard coded"
    },
    "SuperTag": {
      "resolve": "TAG",
      "pattern": "entering.display-name"
    }
  }
}
```


©2025 Carrier.
All rights reserved.

Example of a payload which would be published:


```
{
	"staticExample": "Hard coded",
	"bestTagOfAll": "null",
	"SuperTag": "null",
	"betterTag": "1947",
	"equipment": "Zone 1",
	"tag": "hvac",
	"deviceID": "722d8efd-5f0c-4ca0-a2e8-cc628fccca84",
	"tags": {
		"ABSPATH:1:#vav_zone1/hvac/zone_temp": {
			"value": "72.0",
			"displayName": "Zone Temp",
			"unit": "°F"
		}
	},
	"timestamp": "2025-04-21T18:30:24Z"
}
```


Open source components

Information about open source components used in this application can be found at:

www.controlj.com/opensourceindex.html

©2025 Carrier.
All rights reserved.

Document revision history

Important changes to this document are listed below. Minor changes such as typographical or formatting errors are not listed.

Date	Topic	Change description	Code*
8/27/25	Requirements	Clarified Infrastructure	X-PM-MS-J-MS
	Broker	Added steps to Add a Broker	
	Scan	Updated description	
	Settings	Added Swagger UI and Custom Payload	
	Virtual Edge of Network (VEON)	New topic	
	Custom Payload (Telemetry Only)	New topic	
	MQTT Connector API	Swagger details added	
	Appendix	New topic	

* For internal use only

©2025 Carrier.
All rights reserved.

Carrier ©2025 · Catalog No. 11-808-930-01 · 8/27/2025