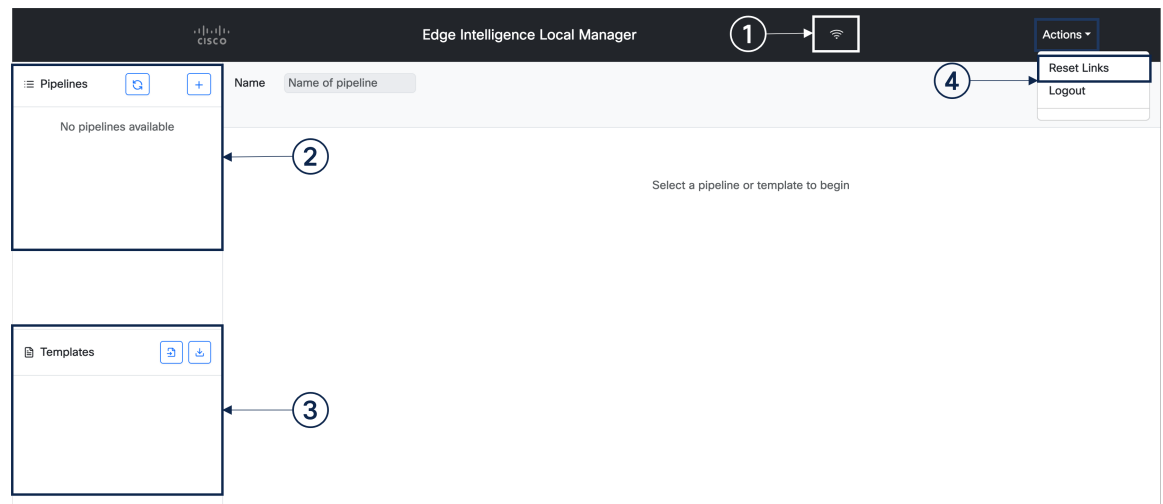




Cisco Edge Intelligence Local Manager

In Cisco Edge Intelligence Local Manager dashboard, we create pipelines to define the data progression. Along with pipeline creation, the following features are available in the dashboard.

Figure 1: Cisco Edge Intelligence Local Manager dashboard



1. Check Cisco Edge Intelligence connection status:

Hover over the network connection icon (Wi-Fi icon) in the top banner to view whether Cisco Edge Intelligence is online or offline, and for agent details such as version and ID.

2. Create and view deployed pipelines:

- The pipelines area provides a quick view of the deployed pipelines and their respective statuses.
- Click an existing pipeline to view its health status details, or to edit the pipeline's configurations.
- To create a new pipeline, click the plus (+) icon.

3. View, import, or export templates:

The templates area displays all the templates that are available in the Cisco Edge Intelligence Local Manager.

- Click the import icon to upload template files from your local system.

- Click the download icon to download one or all the templates to your local system, in JSON format. You can then import the templates into other agents for deployment.

4. Delete all pipelines:

To delete all the pipelines in your Cisco Edge Intelligence Local Manager, choose **Actions > Reset Links** from the top banner.



Caution Once deleted, a pipeline cannot be retrieved.

Templates are browser-specific and access is restricted to your user credentials. However, deployed pipelines do not have similar restrictions. Multiple users can check the health status of a pipeline and edit any existing pipeline configurations.

- [How to create pipelines, on page 2](#)
- [Data sources, on page 3](#)
- [Add data destinations, on page 24](#)
- [About Data Policies, on page 35](#)
- [Deploy or undeploy pipelines, on page 43](#)
- [View health status, on page 44](#)

How to create pipelines

Creating a Cisco Edge Intelligence pipeline involves defining:

1. One/multiple source asset type
2. One data destination
3. A data transformation method, in the form of a data rule or a data logic

To create a pipeline in the Cisco Edge Intelligence Local Manager GUI, in the **Pipelines** area of the left pane, click the plus (+) sign button. At the top of the page, enter a name for the pipeline. Ensure that each pipeline name for the Cisco Edge Intelligence must be unique, in compliance with the following recommendations.

- Do not use special characters for a pipeline name, as special characters are removed during internal processing. We recommend using the CamelCase naming convention.

For example, use `WaterSensorSalinityJ2345` instead of `Water Sensor-Salinity_J2345`.

- Avoid ending a pipeline name with the letter `s` to indicate a plural form, as this name can cause internal naming conflicts.

For example, use `ColdStoragePlc` instead of `ColdStoragePlcs`.

Edge Intelligence Local Manager

Actions ▾

Pipelines

Name Save As Template Deploy Undeploy Cancel

serial_to_mqtt **Error**

ntcip1202_mqtt1 **Error**

Templates

Source Destination Data Policy

Expand All Collapse All + Add Asset 1/20

AssetName : input

Connection Type *
 Choose ▾

Serial No *

Custom Attributes + Add

| # | Name * | Data Type | Value * | Action |
|---|--------|-----------|---------|--------|
|---|--------|-----------|---------|--------|

© 2025 Cisco Systems, Inc.

Data sources

Define assets or data source types based on the communication protocols they use. Each protocol then allows further configurations to define the data sources.

In the **Source** tab, these fields are the required for all type of connection types.

| Field | Description |
|-----------------|--|
| Asset Name | Click the pencil icon to enter a name for the asset. This name gives a significant value to the data logic. |
| Connection Type | From the drop-down list, choose one of the following protocols: <ul style="list-style-type: none"> • MQTT • Modbus – Serial • Modbus – TCP/IP • OPC-UA • Serial • RSU • NTCIP1202 • NTCIP1203 • NTCIP1204 |
| Serial Number | Enter a serial number for the chosen source type. |

| Field | Description |
|--------------------------------|---|
| Custom Attribute Configuration | <p>You can add custom attributes along with each asset type-specific attribute. Add the following details to configure the custom attribute:</p> <ol style="list-style-type: none"> 1. Name: Enter a name for the custom attribute. 2. Data Type: From the drop-down list, select a data type, string, double, encrypted string, or file. 3. Value: Enter a custom attribute value. <p>If the Data Type is selected as File:</p> <ul style="list-style-type: none"> • There can be only one attribute type File for a given asset type. • The max size of the uploaded file should be 12 KB. • The file can be of any type - ASCII or binary. • If the asset is part of a data rule policy and the destination is configured to send the custom attribute to the northbound destination, the base64 equivalent of the file contents will be sent. • If the asset is part of a data logic policy, a custom attribute value is available in the data logic script as a byte array (UInt8Array), and it can be converted to the original format for access. • For example, if the custom attribute "reference_data" was of File type and the uploaded file was as ASCII file, the following code shows how to convert the value available as UInt8Array into ASCII text. <pre>function on_update() { ... var file_contents = new TextDecoder().decode(input.reference_data); // Converts UInt8Array to ASCII ... }</pre> <p>Click Add to add more custom attributes to any asset configuration</p> |

**Note**

Cisco Edge Intelligence Local Manager have the provision to configure the multiple assets under a single pipeline. You can configure upto 20-assets type details and run the deployment.

If you want to duplicate an existing asset details with a different serial number, click copy-paste icon next to the asset name.

Configure MQTT asset type

The MQTT asset type enables and configures the Cisco Edge Intelligence MQTT Server. You can publish data to the EI Agent from an MQTT client on the following ports after deploying this to an EI Agent:

- Port 8883 for TLS configurations
- Port 1883 for non-TLS configurations

In the **Source** tab, from the **Connection Type** drop-down list, choose **MQTT**. Then, fill out the following fields.

The screenshot shows the MQTT configuration form. At the top, there is a 'Connection Type' dropdown set to 'MQTT' and a 'Serial No' field. Below this is the 'Configuration Details' section, which includes a checked 'Enable TLS' checkbox, a 'Username' field, a 'Password' field, a 'Certificate' field with a 'Choose File' button, and a 'Private Key' field with a 'Choose File' button. The 'Advanced Settings' section contains a 'Client Id' field. At the bottom is the 'Attribute Definitions' table, which has columns for '#', 'Name', 'Label', 'Data Type', 'Topic', and 'Action'. The table contains one row with a 'String' data type.

| Field | Description |
|------------------------------|---|
| Configuration Details | |
| Enable TLS | Choose this checkbox to enable TLS. When you choose to enable TLS, additional fields are displayed to allow the upload of certificate and private key files. The MQTT server that is deployed on the EI agent, uses these certificates and private keys to authenticate the MQTT clients connecting to it. |
| Username | Enter a username. The connecting MQTT Clients (sensors) will use this username for authentication at the MQTT Server which will be spawned on the EI Agent. |

| Field | Description |
|------------------------------|--|
| Password | Enter the Password. The connecting MQTT Clients (sensors) will use this password for authentication at the MQTT Server which will be spawned on the EI Agent. |
| Advanced Settings | |
| Client ID | <p>Add a client ID to publish data. (Topic-based device or sensor identity detection is not supported).</p> <p>The client ID, specified in the MQTT client connection, differentiates various MQTT connections to Cisco Edge Intelligence.</p> |
| Attribute Definitions | <p>The data model explains how data is represented in the asset, and what MQTT topics the asset should be updated from. To define a data model, add the following details:</p> <ol style="list-style-type: none"> Name: Enter a name for the data model Label: Enter a label for the data model Data Type: From the drop-down menu, select a measurement entity (string, integer, float, long, or boolean) for the MQTT topic. Topic: Enter the MQTT topic over which the measurement is sent by the transmitting MQTT client. For example, <code>sensors/tempXY/temp</code>. <p>Click Add to add more data model attributes to the configuration</p> |



Note The MQTT topic used to publish sensor data and the data format should match the **data model** JSON file.

MQTT data model example:

```
{
  "apiVersion": 1.0,
  "connectionType": "MQTT",
  "fields": {
    "temperature": {
      "category": "TELEMETRY",
      "label": "Temperature",
      "description": "Outside temperature sensor XY | Temperature",
      "datatype": "Float",
      "topic": "sensors/tempXY/temp"
    },
    "humidity": {
      "category": "TELEMETRY",
      "label": "Humidity",
      "description": "Outside temperature sensor XY | Humidity",

```

```
        "datatype": "Float",
        "topic": "sensors/tempXY/hum"
    },
    "attr1": {
        "category": "ATTRIBUTE",
        "label": "Attribute 1",
        "datatype": "Float",
        "description": "My Attribute 1",
        "required": true,
        "defaultValue": 12.9
    },
    "attr2": {
        "category": "ATTRIBUTE",
        "label": "Attribute 2",
        "datatype": "String",
        "description": "My Attribute 2",
        "required": false,
        "defaultValue": null
    },
    "encrypted_attr3": {
        "category": "ATTRIBUTE",
        "label": "New Attribute",
        "datatype": "EncryptedString",
        "description": "My Attribute 3",
        "required": true,
        "defaultValue": null
    }
  }
}
```

MQTT topic and sensor data used by MQTT Client example:

MQTT Client Topic: sensors/tempXY/hum
MQTT Client Data: 50.0

Configure Modbus-Serial asset type

From the **Device Type** drop-down list, choose **MODBUS-Serial**. Then enter the required details in the following fields.

Configure Modbus-Serial asset type

Connection Type *
Modbus - Serial

Serial No *

Configuration Details

Transport *
RTU

Baud Rate *
9600

Stop Bits
☒ 1 ☐ 2

Slave ID *
1

Serial Port * ⓘ

Parity
None

Data Bits
8

Advanced Settings

Zero On Failed Poll
☐ True ☒ False

Contiguous Batch Request Only
☐ True ☒ False

Timeout (in milliseconds) *
500

Max Read Bit Count
2000

Max Write Register Count
120

Use Batch Polling
☒ True ☐ False

Use Multiple Write Commands
As Appropriate

Retries
2

Max Read Register Count
125

Attribute Definitions + Add

| # | Name * | Label * | Data Type * | Polling Interval (ms) * | Type * | Offset * | RawType | Access Mode | Action |
|---|--------|---------|-------------|-------------------------|--------|----------|---------|-------------|--------|
| 1 | | | String | | COIL | | UINT16 | Read Only | |

| Field | Description |
|------------------------------|--|
| Configuration Details | |
| Transport | This field is not editable. It has a default value set for RTU. |
| Serial Port | Enter a port number. For example, <code>/dev/ttyS0</code> . |
| Baud Rate | Enter a baud rate. |
| Parity | From the drop-down list, choose None , Odd , or Even . |
| Stop Bits | This value can be either 1 or 2. Choose the required radio button. |
| Data Bits | From the drop-down list, choose a value 5–9. |
| Slave ID | This refers to the unique identifier assigned to a Modbus secondary device (such as a sensor or PLC) on a serial network. Modbus protocol requires each secondary device to have a unique ID (1–247) to distinguish it on a shared bus. This ID ensures that the Modbus primary (such as the Cisco EI agent) sends requests to the correct device and processes its responses. |
| Advanced Settings | |
| Zero or Failed Poll | The value if there is no response from the asset. This field is not editable. |
| Use Batch Polling | To request batch responses from the asset. This field is not editable. |

| Field | Description |
|--------------------------------------|--|
| Contiguous Batch Request Only | To request contiguous batch responses from the asset. This field is not editable. |
| Use Multiple Write Commands | This field is not editable. |
| Timeout | The time in which to receive the data before it is reset. The default value is 500. |
| Retries | The number of times the server requests for retransmission of data. This field is not editable. |
| Max Read Bit Count | The maximum number of bits that the server reads in one read request. This field is not editable. |
| Max Read Register Count | The maximum number of registers that the server reads in one read request. This field is not editable. |
| Max Write Register Count | The maximum number of registers that the server writes in one write request. This field is not editable. |
| Attribute Definitions | <p>A data model explains how data is represented in the asset. You can create a data model using a JSON file. Select one of the following:</p> <ol style="list-style-type: none"> 1. Name: Enter a name for the data model. 2. Label: Enter a label value. 3. Data Type: From the drop-down list, select a measurement entity (string, integer, float, long, or boolean) for the MODBUS serial data. 4. Polling Interval (ms): Enter a value to define the fastest rate at which the server must test and debug. 5. Type: From the drop-down list, choose a data type for industrial control of factory devices. The available choices are COIL, HOLDING, DISCRETE, and INPUT. 6. Offset: Enter a value to reference a specific register within the function. 7. RawType: From the drop-down list, choose the value type for storing binary data or byte strings. 8. Access Mode: From the drop-down list, choose an access mode. The available choices are Read Only, Read & Write, and Write Only. <p>Click Add to add more data model attributes to the configuration</p> |

Modbus_serial data model example:

```

{
  "apiVersion": 1,
  "connectionType": "MODBUS_SERIAL",
  "fields": {
    "pressureInPascal": {
      "label": "MetrLabeModb1",
      "pollingInterval": 50,
      "offset": 12,
      "type": "HOLDING",
      "datatype": "Float",
      "rawType": "VARCHARSTRING",
      "description": "",
      "access": "Write"
    },
    "TemperatureInDegrees": {
      "label": "Temperature",
      "pollingInterval": 41,
      "offset": 56,
      "type": "DISCRETE",
      "datatype": "String",
      "rawType": "INT16",
      "description": "",
      "access": "ReadWrite"
    },
    "HumidityInDegrees": {
      "label": "Humidity",
      "pollingInterval": 20,
      "offset": 45,
      "type": "COIL",
      "datatype": "Int",
      "rawType": "FLOAT64",
      "description": "Humidity Value ",
      "access": "Read"
    }
  }
}

```

Adding Asset Types for Modbus TCP Connection Type

After you select the MODBUS-TCP/IP, complete the following additional fields by adding the details on it.

Connection Type *
Modbus - TCP/IP

Serial No *

Configuration Details

IP Address or Hostname

Port *

502

Slave ID *

1

Advanced Settings

Zero On Failed Poll
☐ True ☒ False

Contiguous Batch Request Only
☐ True ☒ False

Timeout (in milliseconds) *
500

Max Read Bit Count
2000

Max Write Register Count
120

Use Batch Polling
☒ True ☐ False

Use Multiple Write Commands
As Appropriate

Retries
2

Max Read Register Count
125

Attribute Definitions

| # | Name * | Label * | Data Type | Polling Interval (ms) * | Type | Offset * | RawType | Access Mode | Action |
|---|--------|---------|-----------|-------------------------|------|----------|---------|-------------|--------|
| 1 | | | String | | COIL | | UINT16 | Read Only | |

Add

| Field | Description |
|--------------------------------------|--|
| Configuration Details | |
| IP Address or Host Name | Enter an IP address or hostname. |
| Port | Enter a port number. |
| Slave ID | This refers to the unique identifier assigned to a Modbus secondary device (such as a sensor or PLC) on a serial network. Modbus protocol requires each secondary device to have a unique ID (1–247) to distinguish it on a shared bus. This ID ensures that the Modbus primary (such as the Cisco EI agent) sends requests to the correct device and processes its responses. |
| Advanced Settings | |
| Zero on Failed Poll | The value if there is no response from the asset. This field is non-editable. |
| Use Batch Polling | To request batch responses from the asset. This field is non-editable. |
| Contiguous Batch Request Only | To request contiguous batch responses from the asset. This field is non-editable. |
| Use Multiple Write Commands | This field is non-editable. |
| Timeout | The time set to receive the data before it is reset. The default value is 500. |
| Retries | The number of times the server requests for retransmission of data. This field is non-editable. |
| Max Read Bit Count | The maximum number of bits that the server reads in one read request. This field is non-editable. |
| Max Read Register Count | The maximum number of registers that the server reads in one read request. This field is non-editable. |
| Max Write Register Count | The maximum number of registers that the server writes in one write request. This field is non-editable. |

| Field | Description |
|------------------------------|---|
| Attribute Definitions | <p>A data model explains how data is represented in the asset. You can create a data model using a JSON file. Enter the following details:</p> <ol style="list-style-type: none"> 1. Name: Enter a name for the data model. 2. Label: Enter a label value. 3. Data Type: From the drop-down menu, select a measurement entity (string, integer, float, long, or boolean) for the MODBUS-TCP/IP data. 4. Polling Interval: Enter a value to define the fastest rate at which the server should test and debug. 5. Type: From the drop-down menu, choose a data type for industrial control of factory devices. The available choices are COIL, HOLDING, DISCRETE, and INPUT. 6. Offset: Enter a value to reference a specific register within the function. 7. RawType: From the drop-down menu, choose the value type for storing binary data or byte strings. 8. Access Mode: From the drop-down list, choose an access mode. The available choices are Read Only, Read & Write, and Write Only. <p>Click Add to add more data model attributes to the configuration</p> |

The following is an example of a data model for Modbus-TCP/IP source type.

```
{
  "apiVersion": 1,
  "connectionType": "MODBUS_TCP",
  "fields": {
    "desired_temp": {
      "label": "Desired Temperature",
      "datatype": "Int",
      "description": "WO",
      "rawType": "UINT16",
      "type": "HOLDING",
      "pollingInterval": 5000,
      "offset": 5,
      "category": "TELEMETRY",
      "access": "Write"
    },
    "temp_to_display": {
      "label": "Temperature to be displayed",
      "datatype": "Int",
      "description": "RW",
      "rawType": "UINT16",

```

```

    "type": "HOLDING",
    "pollingInterval": 5000,
    "offset": 100,
    "category": "TELEMETRY",
    "access": "ReadWrite"
  },
  "temp": {
    "label": "Current Temperature",
    "datatype": "Int",
    "description": "RO",
    "rawType": "UINT16",
    "type": "HOLDING",
    "pollingInterval": 5000,
    "offset": 1,
    "category": "TELEMETRY",
    "access": "Read"
  }
}

```

Configure OPC-UA asset type

After you select the asset type OPC-UA, enter the required details in the following fields:

Connection Type *
OPC-UA

Serial No *

Configuration Details

IP Address or Hostname *

Port *

Publishing Interval (in milliseconds) * ⓘ

Advanced Settings

Authentication
☒ Anonymous ☐ Username & Password
Security Mode
☐ None

Attribute Definitions

| # | Name * | Label * | Data Type * | OPC-UA Type | NameSpace URI * | NameSpace Index * | Identifier * | Sampling Interval * | Type |
|---|--------|---------|-------------|-------------|-----------------|-------------------|--------------|---------------------|------|
| 1 | | | String | URI | | | | 1000 | Nu |

Field

Description

Configuration Details

IP Address or Host Name

Enter an IP address or hostname.

Port

Enter a port number.

| Field | Description |
|----------------------------|---|
| Publishing Interval | <p>This is the requested publishing frequency from the OPC-UA Server. The interval must be greater than or equal to 1000ms.</p> <p>Note The OPC-UA server publishing frequency is independent of the metric-specific sample interval. In case the sampling interval (in the following attribute table) of an individual metric is smaller than the publishing interval, the OPC-UA server queues up and send all the sampled values for a metric between the last publish and the current publish.</p> |
| Advanced Settings | |
| Authentication | <p>Choose an authentication type:</p> <ul style="list-style-type: none"> • Anonymous: The OPC-UA client inside the EI Agent does not authenticate at the OPC-UA server. Use this authentication type if your OPC-UA server does not have authentication that is enabled for connecting clients. • Username & Password: Enter the username and password that the EI Agent must use to authenticate at the OPC-UA server. |
| Security Mode | This field is non-editable. The default value is None . |

| Field | Description |
|------------------------------|---|
| Attribute Definitions | <p>A data model explains how data is represented in the asset.</p> <ol style="list-style-type: none"> Name: Enter a name for the data model. Label: Enter a label for the data model. Data Type: From the drop-down menu, select a measurement entity (string, integer, float, long, or boolean) for the OPC-UA data. OPC-UA Type: From the drop-down menu, choose URI or INDEX. Namespace URI: Enter a value to identify the naming authority that defines the identifiers of Node IDs. <p>Note This field is editable if you select URI under OPC-UA Type.</p> Namespace Index: Enter a value to identify the naming authority that defines the identifiers of Node IDs. <p>Note This field is editable if you select INDEX under OPC-UA Type.</p> Identifier: Enter a value that is unique across different naming authorities. Sampling Interval: Enter a value to indicate the fastest rate at which the server should sample its underlying source for data changes. Type: From the drop-down menu, choose the data type for the identifier. The available choices are Numeric and String. <p>Click Add to add more data model attributes to the configuration</p> |

The following is an example of a data model for OPC-UA source type.

```
{
  "apiVersion": 1,
  "connectionType": "OPC-UA",
  "fields": {
    "temperature": {
      "label": "Temperature",
      "description": "",
      "datatype": "Float",
      "nodeId": {
```

```
        "namespaceUri": "2",
        "identifier": "2",
        "type": "numeric"
    },
    "samplingInterval": 1000,
    "category": "TELEMETRY"
}
}
```

Configure Serial asset type

After you select the asset type **Serial**, enter the required details in the following fields.

Connection Type *

Serial

Serial No *

Configuration Details

Serial port *

Parity

None

Data Bits

7

8

Baud Rate *

9600

Stop Bits

1

2

Attribute Definitions

#

1

Name *

Label *

Data Type *

String

Access

READ

Start Code

End Code

Message Size

Tin

+ Add

| Field | Description |
|-----------------------|--|
| Configuration Details | |
| Serial Port | Enter a port number. For example, /dev/ttyS0 |
| Baud Rate | From the drop-down menu, choose 9600, 19200 or you can add a new custom value on it. |
| Parity | From the drop-down menu, choose None , Odd , or Even |
| Stop Bits | Choose the radio button for 1 or 2 . |
| Data Bits | From the drop-down menu, choose 7 or 8 . |

16

Cisco Edge Intelligence Local Manager

| Field | Description |
|-----------------------|---|
| Attribute Definitions | <p>A data model explains how data is represented in an asset.</p> <ol style="list-style-type: none"> Name: Enter a name for the data model. Label: Enter a label for the data model. Data Type: From the drop-down menu, choose String or Binary. If you select Binary, the data is delivered in binary form 1. <ol style="list-style-type: none"> For a data logic policy, the data is delivered as a binary buffer in the data logic script in the <i>on_update()</i> function. For a data rule policy, the data is sent to the northbound destination in the base64 format. Access: From the drop-down menu, choose read, write, or read and write. Start Code: Enter the marker that indicates the start of a stream of bytes. End Code: Enter the marker that indicates the end of a stream of bytes. Message Size: This is the size, in bytes, between the start code and the end code. Timeout: Enter a time, in milliseconds, within which to receive data, before it is reset. This is enabled only for READ and READWRITE access modes. This field is mandatory only if the message size is configured. |

Common errors and troubleshooting

Errors can easily occur when configuring a serial port. For example:

- The wiring must be accurate. For example, see the [Cisco IR829 installation guide](#).
- The serial relay service should be configured correctly for the Guest OS. For example, see the [Cisco Catalyst IR1101 documentation](#) and [Cisco Catalyst IR1800 documentation](#).
- The physical serial port must be correctly exposed to IOx through the Local Manager.

To troubleshoot a serial interface:

- Make sure that serial port is configured in propagate mode at the IOS level. A current workaround is to use just the 0x prefix as the StartCode to specify an empty StartCode.
- For testing interface options only:
 - Use a data model with a fixed message size of 1 byte and no start code.

- Verify that there is some data that is coming in to ensure that the connection is working.
- Once this is done, the actual data model can be defined.

Serial Connector data model

A serial connector asset type has a reduced data type.

The following combinations are allowed:

- One read attribute
- One read plus one write attribute
- One read-write attribute

Read attribute allows the following configuration combinations:

- Message Size and Timeout
- StartCode and Message Size and an optional Timeout
- StartCode and EndCode and an optional Timeout

Start-/End-Code prefix handling:

- Prefix 0x allows to specify hex encoded binary data. For example, 0x1310 -> CR+LF)
- **Prefix** allows to specify **as-is**

Serial data model example:

```
{
  "apiVersion": 1,
  "connectionType": "SERIAL",
  "fields": {
    "data_string1": {
      "label": "My Data String",
      "datatype": "String",
      "description": "serial read attribute",
      "access": "READ",
      "startCode": "$",
      "endCode": "0x0a",
      "messageSize": ""
    }
  }
}
```

RSU asset type settings

The RSU asset type supports a set of static attributes in addition to the regular configurable attributes. Static attributes are always available and not required to be configured.

- The static attributes have a JSON string content and reflect incoming DSRC messages, except storeAndRepeatMessage and broadcastImmediately static attributes.
- The storeAndRepeatMessage must be set as an array of message objects. All previous messages will be overwritten by the new array.

- The current array of messages can be obtained by reading the attribute.
- The basic configuration for this asset type includes the host, port, and SNMP version.
- Advanced settings like community or authentication data must be set depending on the SNMP version.

In the **Source** tab, from the **Device Type** drop-down list, choose **RSU**. Then, fill the following fields.

Connection Type *
RSU

Serial No *

Configuration Details

IP Address or Host Name *

Port
4444

SNMP Version *
3

Advanced Settings

Security Name *

Security Level *
AuthPriv

Authentication password *

Authentication Protocol *
SHA

Privacy password *

Privacy Protocol *
AES

Default Attributes

Attribute Definitions

| # | Name * | Label * | Data Type | Object ID(OID) * | OID Data Type | Polling Interval (ms) | Access Mode | Action |
|---|--------|---------|-----------|------------------|---------------|-----------------------|-------------|--------|
| 1 | 0 | 0 | Choose | 0 | Choose | 0 | Choose | |

| Field | Description |
|--------------------------------|---|
| Configuration Details | |
| IP Address or Host Name | Enter the IP address or hostname. |
| Port | Enter the port number. |
| SNMP Version | Select a version from the drop-down list from 1, 2c and 3. 3 is the most secure version. |
| Advanced Settings | Complete these fields based on the selected SNMP Version. |
| Default Attributes | It shows the default attributes that are specific to RSU. Even if you will not add any additional attributes, it runs for the agents. |

| Field | Description |
|------------------------------|--|
| Attribute Definitions | <p>A data model explains how data is represented in an asset.</p> <ol style="list-style-type: none"> Name: Enter a name for the data model. Label: Enter a label for the data model. Data Type: From the drop-down list, choose String, Integer, Float, Long, or Boolean. Object ID (OID): OID address is used to uniquely identify managed devices and their statuses. OID Data Type: OID is the data type for the object. String or Integer. Polling Interval (ms): This indicates the fastest rate at which the Server should test and debug. Select a unit from the up-down menu. Access Mode: From the drop-down list, choose read, write, or read and write. <p>Click Add to add more attributes to the asset configuration.</p> |

NTCIP1202, NTCIP1203, NTCIP1204 asset type settings

Cisco Edge Intelligence Local Manager supports three NTCIP devices. Use the Asset Type for the correct connection type.

- NTCIP 1202—Actuate Signal Controller
- NTCIP 1203—Dynamic Message Sign
- NTCIP 1204—Road Weather Information System

The basic configuration for all these three asset types include the host, port, and SNMP version.

Advanced settings like community or authentication data must be set depending on the SNMP version.

Each NTCIP asset type supports a set of static attributes in addition to the regular configurable attributes.

Connection Type *
NTCIP1202

Serial No *

Configuration Details

IP Address or Host Name *
SNMP Version *
☒ Enable Streaming
Standard or Asset Manufacturer *
SAE J2735 Standard
Intersection Name

Port
161
Trap Port
162
Intersection ID
0

Advanced Settings

Security Name *
Authentication password *
Privacy password *
OID Count per Request
32

Security Level *
AuthPriv
Authentication Protocol *
SHA
Privacy Protocol *
AES

Attribute Definitions

| # | Name * | Label * | NTCIP Type * | Data Type | Object ID * | Trap OID * | OID Data Type * | Polling Interval (ms) * | Access Mode | Service | Action |
|---|--------|---------|--------------|-----------|-------------|------------|-----------------|-------------------------|-------------|--------------|--------|
| 1 | | | Default | String | RawSpat | | Integer | | Read Only | TRAP_RECEIVE | |

Field**Description****Configuration Details**

IP Address or Host Name

Provide the IP address or Host Name.

SNMP Version

Select a version from the drop-down list from 1, 2c and 3.

3 is the most secure version.

Port

Provide a NTCIP connector port number.

Note

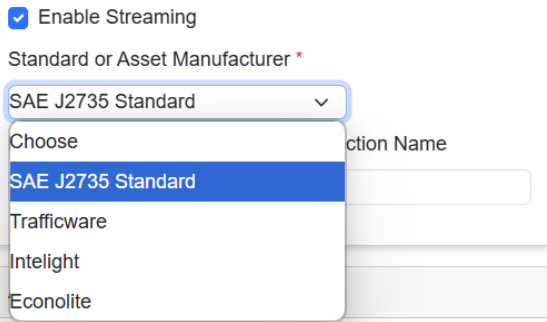
NTCIP1202 port number should never match Trap port number, and vice versa.

Trap Port (Applicable only for device type: NTCIP1202)

Provide Trap port number.

Note

Trap port number should never match NTCIP1202 port number, and vice versa.

| Field | Description |
|---|--|
| Enable Streaming (Specific to device type: NTCIP1202) | <p>Choose this checkbox to enable Streaming.</p> <p>When you choose to enable Streaming, an additional field Standard or Asset Manufacturer protocol is displayed. NTCIP 1202 allows the streaming of data using standard and Asset manufactured protocols. The different protocols are:</p> <p>Standard protocol:</p> <ul style="list-style-type: none"> • SAE J2735 Standard <p>Asset manufactured protocols</p> <ul style="list-style-type: none"> • Trafficware • Intelight • Econolite  |
| Intersection ID (Specific to device type: NTCIP1202) | Intersection ID is optional and depends on the user's specific use case. You can choose to configure this field if required for their particular scenario. For example, in traffic use cases, this information is mandatory. |
| Intersection Name (Specific to device type: NTCIP1202) | Intersection Name is optional and depends on the user's specific use case. You can choose to configure this field if required for their particular scenario. |
| Advanced Settings | The specific Advanced Setting details vary based on the SNMP Version. Provide the appropriate authentication information accordingly. |

| Field | Description |
|-----------------------|---|
| Attribute Definitions | <p>A data model explains how data is represented in an asset.</p> <ol style="list-style-type: none"> Name: Enter a name for the data model. Label: Enter a label for the data model. NTCIP Type: From the drop-down list, choose Default or trap. Data Type: From the drop-down list, choose String, Integer, Float, Long, or Boolean. Object ID: From the drop-down list, choose RawSpat, NTCIP-1211 SRM, and SAE J2735 SRM. Bottom to that add OID. OID address is used to uniquely identify managed devices and their statuses. <p>Note This field is applicable if you select Default under NTCIP Type.</p> Trap OID: Enter an ID for Trap NTCIP type. <p>Note This field is applicable if you select Trap under NTCIP Type.</p> OID Data Type: OID is the data type for the object. String or Integer. Polling Interval (ms): It indicates the fastest rate at which the Server tests and debug. Select a unit from the up-down menu. Access Mode: From the drop-down list, choose read, write, or read and write. <p>Note This field is applicable if you select Default under NTCIP Type.</p> Service: From the drop-down list, choose TRAP_RECEIVE. <p>Note This field is applicable if you select Trap under NTCIP Type.</p> |

Add data destinations

In the **Destination** tab, define where configured data policies must send data to.

The supported destinations are:

- MQTT servers
- AWS server
- Azure IoT Hub
- Splunk

From the **Type** drop-down list, choose the destination. Then, configure the connection settings for the chosen destination.

Before you configure a data destination in the pipeline, ensure that you have set up the servers or hubs. You must have the required identifying information ready to complete the destination configuration.

Table 1: Feature History Table

| Feature | Release Information | Feature Description |
|--------------------|---------------------|---|
| Splunk destination | Release 2.2.x | Splunk is introduced as a new data server destination in this release. Splunk integration serves as a new data server for configuring a pipeline in Cisco Edge Intelligence. |

Add a Microsoft Azure IoT Hub destination

Procedure

In the **Destination** tab, from the **Type** drop-down list, choose **AZURE IoT**, and enter the required details in the following fields.

Data Destination

Type *
Azure IoT

Azure IoT Connection Details

ID Scope *

CA Certificate *

Choose File
or drag and drop your file here

CA Certificate Key *

Choose File
or drag and drop your file here

Advanced Settings

Device Provisioning Endpoint *
global.azure-devices-provisioning.net

☐ Enable MQTT Over Websockets

Data Rule Classification *
Select an option

Message structure

Asset Attribute
☐ Include Asset Attributes
☐ Group asset attributes under property

Telemetry Data
☐ Group telemetry data under property
☐ Include Timestamps

①
attributeData

①
deviceData

Example:

```
{
  "telemetryData": 33.4,
  "telemetryData": 45.2
}
```

| Field | Description |
|-------------------------------------|--|
| Azure IoT Connection Details | |
| ID Scope | Enter the ID Scope that is displayed in the Azure Device Provisioning Service Overview page. |
| CA Certificate | <p>Upload the intermediate CA Certificate file that you have configured in your Azure Device Provisioning Service.</p> <p>Note The status of this CA certificate in the Azure Provisioning Service must be marked as trusted, or device creation (and therefore metric sending) is not allowed by Azure.</p> |
| CA Certificate Key | <p>Upload the unencrypted private certificate key file that belongs to the intermediate CA certificate that you uploaded. The private key must be in PKCS8 format, and must not include a passphrase.</p> <p>To convert an existing key, use the command: <code>openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in azure-iot-test-only.intermediate.key.pem -out azure-iot-test-only.intermediate.pkcs8.pem</code></p> |
| Advanced Settings | |
| Device Provisioning Endpoint | Enter the Global Device Endpoint value from the Azure IoT Hub Device Provisioning Service. |

| Field | Description |
|-----------------------------|---|
| Enable MQTT over WebSockets | Check this check box to enable the browser to leverage all MQTT features. |
| Data Rule Classification | From the drop-down list, choose DEVICE_PROPERTY or TELEMETRY . |
| Message Structure | <p>Choose the required options in this area to customize the structure of the device-to-cloud message. You can choose Include asset attributes, Group asset attributes under property in the Asset Attributes section or choose Group telemetry data under property, Include Timestamps in the Telemetry Data section.</p> <p>The data can be sent in a flat structure or can be grouped with a key. Customization does not apply if the chosen data policy is of the type Device Properties.</p> |

Add an MQTT Server destination

Procedure

In the **Destination** tab, from the **Type** drop-down list, choose **MQTT Server** and enter the required details in the following fields.

Source
Destination
Data Policy

MQTT Server

Connection Details

Broker *

Port *

Additional custom topics can be defined in your Data Logic(s). Refer the documentation for more details.

Topic *

☐ Enable TLS

Username

Password

Advanced Settings

QoS *

Client ID *

☒ Retain Messages

Message Structure

Asset Attribute

☒ Include Asset Attributes
☐ Group asset attributes under property

Telemetry Data

☒ Group telemetry data under property
☒ Include Timestamps

attributeData

deviceData

Example:

```
{
  "assetAttribute1": "Herkules",
  "assetAttribute2": 120,
  "deviceData": {
    "telemetryData1": {
      "v": 22.4,
      "ts": 1545730073
    },
    "telemetryData2": {
      "v": 45.2,
      "ts": 1511913359
    }
  }
}
```

| Field | Description |
|---------------------------|--|
| Connection Details | |
| Broker | Enter the URL or IP address of your MQTT broker. |
| Port | Enter the port number used by the broker. |

| Field | Description |
|------------|---|
| Topic | <p>Enter the topic to which device states and other data are published. For example, <code>cisco/edge-intelligence/telemetry/%deviceSerialNumber%%deviceSerialNumber%</code>. The example topic matches the device or asset instance serial number that is configured previously in the source tab.</p> <p>Note MQTT topic has a restricted number of characters that can be used in a topic name. For example, # or + cannot be part of a topic name.</p> <p>Topic names are URL-encoded to ensure that they do not violate MQTT specifications. URL-encoding also allows northbound applications to decode a topic easily to get to the original contents.</p> |
| Enable TLS | <p>Check the TLS check box to enable the protocol. When you choose TLS, the following fields are displayed:</p> <ul style="list-style-type: none"> • Verify Peer: Check this checkbox to allow peer verification. When you select this option, the Certificate field is displayed where you can upload a CA certificate. • Enable X.509: Check this check box to use X.509 certificates. When you select this option, two fields are displayed where you can upload a CA certificate and a private key. <p>Enabling the use of X.509 certificates allows you to turn on configure MQTT brokers (like Mosquitto) to require certificates for authentication. You can also use X.509 client certificates instead of usernames and passwords to ensure that only trusted assets are allowed to send data to a cloud MQTT broker.</p> <p>Note</p> <ul style="list-style-type: none"> • The private key must be PKCS8-compatible. • The generated certificate for each asset contains the CN - Serial Number of the asset. • X.509 certificates can be used with the username and password authentication method, or as the only authentication method. |
| Username | Enter the username to connect to the MQTT destination broker. |
| Password | Enter the password to connect to the MQTT destination broker. |

| Field | Description |
|--|---|
| Advanced Settings | |
| QoS | From the drop-down list, choose 0 , 1 , or 2 . |
| Client ID | <p>Enter a Client ID.</p> <p>The Client ID field is not URL-encoded because there are no restrictions in MQTT specifications about allowed characters.</p> |
| Retain Messages | Check this check-box to retain messages on the broker for new subscribers. |
| Message Structure | <p>Choose the required options in this area to customize the structure of the device-to-cloud message. You can choose Include asset attributes, Group asset attributes under property in Asset Attributes section or choose Group telemetry data under property, Include Timestamps in Telemetry Data section.</p> <p>The data can be sent in a flat structure or grouped with a key. Customization does not apply if the chosen data policy is of the type Device Properties.</p> |
| Cloud to Network Device | |
| Enable cloud to data logic commands | <p>Use the cloud to data logic commands to send a command (with payload) from a cloud app to a data logic on an edge device. The data logic script parses the command.</p> <p>This feature provides bi-directional communication between the cloud and edge, allowing the application to send a command and receive a response. For example, a cold storage unit connected to an edge device can send commands to:</p> <ul style="list-style-type: none"> • Set the temperature on the cold storage unit • Initiate a defrost action on the cold storage unit <p>To enable cloud to data logic commands, enter the following topic values:</p> <ol style="list-style-type: none"> Command Topic: Enter the syntax and variables in the format, <i>cisco/edge-intelligence/commands/variable</i>. Response Topic: The response topic must use the format <i>cisco/edge-intelligence/responses/variable</i>. <p>Responses are optional. If a script doesn't send a response, nothing is published to the topic.</p> |

Add an AWS Server destination

Procedure

From the **Destination** tab, from the **Type** drop-down list, choose **AWS** and enter the required details in the following fields.

Type *
AWS

Connection Details

Broker *
Port *

Additional custom topics can be defined in your Data Logic(s). Refer the documentation for more details.

Topic *
☐ Enable TLS
Username
Password

Advanced Settings

QoS *
1
Client ID *
☐ Retain Messages ⓘ

Message Structure

Asset Attribute
☐ Include Asset Attributes
☐ Group asset attributes under property
Telemetry Data
☐ Group telemetry data under property
☐ Include Timestamps

attributeData
deviceData

Example:

```
{
  "telemetryData1": 22.4,
  "telemetryData2": 45.2
}
```

| Field | Description |
|---------------------------|---|
| Connection Details | |
| Broker | Enter the URL or IP address of your AWS broker. This info can be found from the AWS IoT Settings page |
| Port | Enter 8883. |
| Topic | Enter a topic to which device states and other data are published. |

| Field | Description |
|--------------------------------|---|
| Enable TLS | For AWS destinations, you must enable the TLS protocol. When you choose TLS, the following fields are displayed: <ul style="list-style-type: none"> • Verify Peer: Do not choose this option. • Enable X.509: Check this check box to use X.509 certificates. When you select this option, two fields are displayed where you can upload a CA certificate and a private key. |
| Username | Do not enter any value in this field. |
| Password | Do not enter any value in this field. |
| Advanced Settings | |
| QoS | From the drop-down list, choose 1 . |
| Client ID | Enter a Client ID. |
| Retain Messages | For AWS destination type, do not choose the Retain Messages option. This option retains messages on the broker for new subscribers. |
| Message Structure | Choose the required options in this area to customize the structure of the device-to-cloud message. You can choose to include asset attributes, include timestamps, group asset attributes, or group telemetry data in the device-to-cloud messages. The data can be sent in a flat structure or can be grouped with a key. Customization does not apply if the chosen data policy is of the type Device Properties. |
| Cloud to Network Device | |

| Field | Description |
|-------------------------------------|--|
| Enable cloud to data logic commands | <p>Use the cloud to data logic commands to send a command (with payload) from a cloud app to a data logic on an edge device. The data logic script parses the command.</p> <p>This feature provides bi-directional communication between the cloud and edge, allowing the application to send a command and receive a response. For example, a cold storage unit connected to an edge device can send commands to:</p> <ul style="list-style-type: none"> • Set the temperature on the cold storage unit • Initiate a defrost action on the cold storage unit <p>To enable cloud to data logic commands, enter the following topic values:</p> <p>a. Command Topic: Enter the syntax and variables in the format, <i>cisco/edge-intelligence/commands/variable</i>.</p> <p>b. Response Topic: The response topic must use the format <i>cisco/edge-intelligence/responses/variable</i>.</p> <p>Responses are optional. If a script doesn't send a response, nothing is published to the topic.</p> |

Add a Splunk server destination

Splunk is a software platform designed to collect, analyze, and visualize machine-generated data in real time, delivering operational intelligence for IT operations, security, and business analytics. With the capability in Cisco Edge Intelligence to use Splunk as a destination, customers can seamlessly send their data to Splunk.



Note When you choose Splunk as the destination type, set the data policy type to Data Logic. Data Rule policy type is not supported for this destination.

This procedure explains the fields and settings for the Splunk server destination.

Before you begin

Make sure that you have an active Splunk account with the HTTP Event Collector enabled.

Procedure

In the **Destination** tab, select **Splunk** from the **Type** drop-down list. Enter the required details in each field.

Table 1 lists the required configuration fields and settings for the Splunk server destination.

| Field | Description |
|--|---|
| Connection Details | |
| HEC URL | Enter the complete URL for the HTTP Event Collector (HEC). |
| HEC token | Enter the HEC token. |
| For more information about configuring the HEC token and other configuration details, see the HTTP Event Collector Guide . | |
| Enable TLS | <p>For Splunk destinations, you must enable the TLS protocol. Enabling TLS displays these checkboxes.</p> <ul style="list-style-type: none"> • Verify Server Certificate: This option enables certificate validation for secure connections. • Enable Mutual TLS (mTLS): This option allows the use of client certificates for authentication. |
| Verify Server Certificate | Select the checkbox to upload a CA Certificate Bundle , which verifies the indexer. |
| Enable Mutual TLS (mTLS) | <p>Selecting this checkbox displays two fields for uploading a client certificate and client private key.</p> <ul style="list-style-type: none"> • Client Certificate: Provide a PEM file containing the client certificate. • Client Private key: Provide a PEM file containing the decrypted private key associated with your client certificate. |

How to configure single or batch payloads

Splunk data can be sent as a single payload or in batches. This approach is generally used for handling large volumes of data.

This sample snippet demonstrates how to send North Bound data to Splunk.

- Using a single payload

```
function init() {
  logger.info("Starting initialization")

  // SSL setup can be added here if required
}

var counter = 100

function on_update() {
  // Reserved for external update triggers
}

function on_time_trigger() {
```

```

    counter = counter + 1

    // Create a single event payload
    var payload = {
      event: {
        pressure: counter
      },
      host: "FCW22360076",
      source: "FCW22360076",
      sourcetype: "EI Agent"
      // Optional fields like index or timestamp can be added here
    }

    // Send the payload to output immediately
    publish("output", payload);
  }

```

• Using a batch payload

```

• function init() {
  logger.info("Starting initialization and setup")
  // SSL options can be added here if needed in future
}

// Message buffer to hold event data before sending
var messageBuffer = []

// Max number of events to buffer before sending
const maxBufferSize = 2

// Example counter for generating event values
var counter = 100

function on_update() {
  // Reserved for handling updates from external source
}

function on_time_trigger() {
  counter = counter + 1
  // non batch payload
  var payload = {
    "event": {
      "data": counter,
      "escaped_chars": "Line 1\\nLine 2\\tTabbed\\\\"Quoted\\\"",
    },
    "host": "FCW22360076",
    "source": "FCW22360076",
    "sourcetype": "EI Agent",
    //"index": "ei-hec-index",
    // "time": new Date(trigger.timestamp).getTime() / 1000
  };
  //publish("output", payload);

  // batch payload
  for (var i = 0; i < 3; ++i) {
    messageBuffer.push(payload);
  }
  publish("output", messageBuffer);
  messageBuffer = [];
}

```

About Data Policies

Data Policies define how data is sent from edge assets to a destination. There are two types of policies:

- **Data Logic:** Data is transformed before being sent to a destination. Data Logic scripts are developed using Microsoft VS Code and embedded UI editor.
- **Data Rule:** Data is sent from Assets to a destination without transformation.



Restriction Make sure to configure **Data Logic** when you have configured multiple Source type. Proceeding only with **Data Rule**, will pop-up an error while deployment.

Table 2: Feature History Table

| Feature | Release Information | Feature Description |
|---|---------------------|--|
| HTTP supports in data logic script | Release 2.2.x | HTTP and HTTPS are new enhancement features in the data logic script that allows you to run various methods to fetch data from the source. |
| Get a device's asset serial number from global device model | Release 2.2.x | <p>Cisco Edge Intelligence data logic scripts now offer enhanced flexibility for accessing a device-specific attribute. You can retrieve asset serial number directly from the global device mode using the <code>on_update()</code> function.</p> <pre>function on_update() { serial_number = input.asset_serial_number; serial_number = globalThis[trigger.device_name].asset_serial_number; }</pre> |

Create a Data Logic

Data logic is used to transform data from connected assets before it is delivered to a destination. Unlike Data Rules that send all the raw data for an Asset Type, Data Logic allows you to aggregate or average data, send only data that exceeds certain value, detect anomalies, and more.

Data Logic is developed and debugged using JavaScript in Microsoft Visual Studio (VS Code) and embedded UI editor. Scripts are synchronized to Cisco IoT where they can be deployed to EI Agents running on Edge Devices.

Procedure

Step 1 From the **Data Policy** tab, select **Data Logic** checkbox.

Data Logic Configuration

Data Logic Script File (.js) * [DL Editor](#)

[Choose File](#)

or drag and drop your file here

Data logic file is required

☐ Invoke Periodically (in ms) [i](#)

☐ Invoke on New Data

☐ Cloud to Device Command [i](#)

☐ Enable Raw Mode

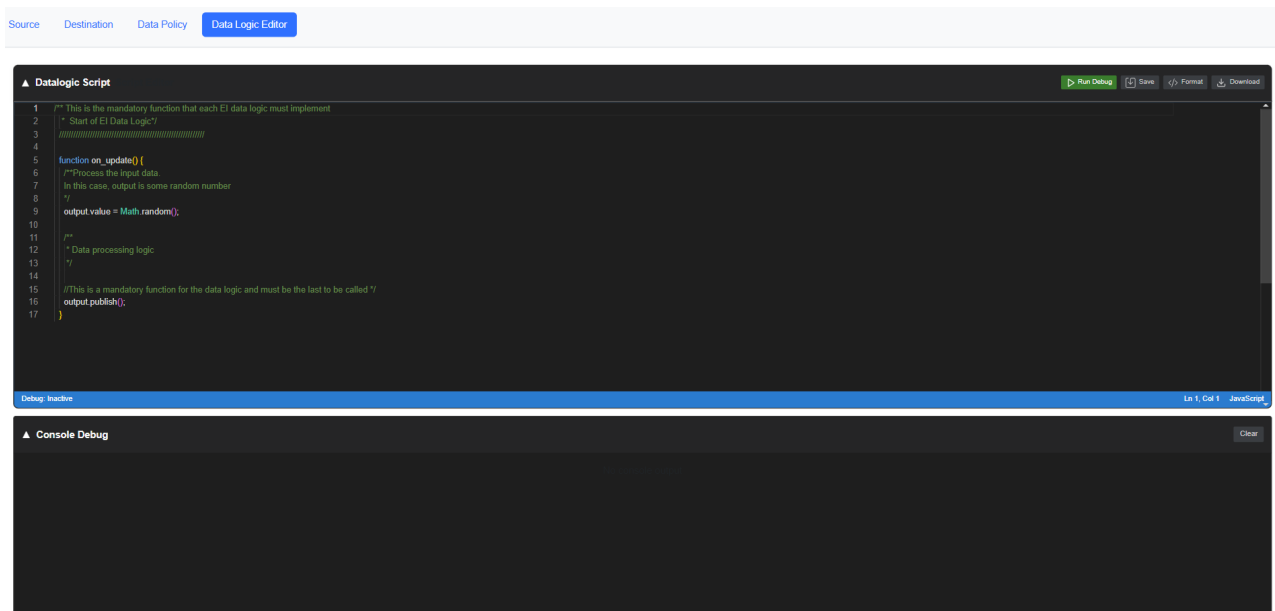
Note

Clicking upon **Data Logic** radio button the **Data Logic Editor** tab appears next to Data Policy.

Step 2 To add a data logic script, click **Choose File** to select precreated new script (example: **LogicExample.js**) from your local drive.

Step 3 To review or to create a new data logic script, click **DL Editor**.

- Data logic functionalities where user can write his own transformation code to help to transform data from source to destination.
- It helps to verify and validate the existing script file by clicking **Run Debug**.
- You can write a new script and validate it.

**Note**

HTTP functionality is introduced on data logic script. Data logic editor supports HTTP CRUD (create, retrieve, update and delete) operation. To know various supported HTTP features, refer to [HTTP and HTTPS protocols in data logic scripts, on page 38](#).

Step 4 After you upload a script file, the new download and delete icon appears. It helps to check/validate what uploaded from other agents. Another user can also check an existing pipeline and can download it.

Step 5 From the following run-time options, select when the script will be run.

- **Invoke Periodically (in ms):** when you tick the checkbox, it enables the ms interval. Enter the interval time, in ms. For example, if you enter 500, the Data Logic script will be called every 500 ms. Enable this option if needed.
- **Invoke on New Data:** The script is called when data changes.
- **Cloud to Device Command:** This function is called when you receive a command from the cloud.
- **Enable Raw Mode:** A default output JSON data model is automatically created when a Data Logic script is created.
- (Optional) In the **Output Logic Data Model**, you can modify the default format of the Output Data Logic Model script (in JSON) and specify the output model with custom names.

Valid categories: TELEMETRY, PROPERTY, ATTRIBUTE

Valid types: string, int, binary, boolean, double

Output Logic Data Model

```
[
  {
    "key": "value",
    "type": "DOUBLE",
    "category": "TELEMETRY"
  }
]
```

Valid categories: TELEMETRY, PROPERTY, ATTRIBUTE

Valid types: string, int, binary, boolean, double

HTTP and HTTPS protocols in data logic scripts

The data logic editor supports HTTP and HTTPS protocols. For these protocols, you can use the CRUD operations, configure SSL settings, timeout settings, and cancel requests.

Cisco devices use the following ports by default:

| protocol | Device port |
|----------|-------------|
| HTTP | 80 |
| HTTPS | 443 |

If a device uses the default ports for HTTP and HTTPS communications, the configurations in the data logic script are automatically applied to the port. If a device uses a different port for these protocols, additional configurations may be required.

Secure the HTTP and HTTPS communications by using one of these TLS configurations.

- Trusted Certificate Authority (CA)
- Self-signed certificates
- Self-signed CA
- Mutual TLS (mTLS) using client certificate and key
- Use the API (`setSslOptions`) to configure the root certificate, client certificate, and client key

| Method | Signature | Sample Snippet |
|-------------|--|--|
| GET | <pre>/** HTTP GET: Fetch all items Signature: http.get(url: string, headers: object, callback: function) Example: Get list of all items Expected: 200 OK with array of items */</pre> | <pre>function getAllItemsCallback(err, status, body, headers) { if (err) { logger.error("[GET ALL] Error:", err.message); } else { logger.info("[GET ALL] Success:", status, body); } } http.get("http://localhost:3000/a pi/data", { "Accept": "application/json" }, getAllItemsCallback);</pre> |
| POST | <pre>/** HTTP POST: Create new item* Signature: http.post(url: string, body: object string, headers: object, callback: function)* Example: Create new item with name "Item Three" * Expected: 201 Created */</pre> | <pre>function createItemCallback(err, status, body, headers) { if (err) { logger.error("[POST] Error:", err.message); } else { logger.info("[POST] Success:", status, body); } } http.post("http://localhost:3000/a pi/data", { name: "Item Three" }, { "Content-Type": "application/json" }, createItemCallback);</pre> |

| Method | Signature | Sample Snippet |
|--------------------|---|---|
| DELETE | <pre>/** * HTTP DELETE: Delete * item by ID * Signature: * http.delete(url: string, * callback: function) * Example: Delete item * with ID 2 * Expected: 200 OK or 404 * Not Found */</pre> | <pre>function deleteItemCallback(err, status, body, headers) { if (err) { logger.error("[DELETE] Error:", err.message); } else { logger.info("[DELETE] Success:", status, body); } } http.delete("http://local host:3000/api/data/2", deleteItemCallback);</pre> |
| PUT | <pre>/** * HTTP PUT: Update item * by ID * Signature: * http.put(url: string, * body: object string, * headers: object, callback: * function) * Example: Update item ID * 1 to have a new name * Expected: 200 OK or 404 * Not Found */</pre> | <pre>function updateItemCallback(err, status, body, headers) { if (err) { logger.error("[PUT] Error:", err.message); } else { logger.info("[PUT] Success:", status, body); } } http.put("http://localhost:3000/a pi/data/1", { name: "Updated Item One" }, { "Content-Type": "application/json" }, updateItemCallback);</pre> |
| SSL Options | <pre>/** * SSL Configuration * Example (optional) * Signature: * http.setSSLOptions(options * : object) * Example: Enable selfsigned * certificate support * Use this before calling * HTTPS endpoints */</pre> | <pre>http.setSSLOptions({ verify: true, verifyHostname: true, // allowSelfSigned: true, // });</pre> |

| Method | Signature | Sample Snippet |
|--------------------------------------|---|---|
| SSL options with Certificates | <p>Certificates and keys can also be provided via the <code>customAttribute</code> file options. These values can be integrated into the data logic. Before being passed to <code>setSslOptions</code>, the certificate files must be base64- encoded, as demonstrated in the provided example.</p> | <pre>function decodeFileContent(fileAt tribute) { // File attributes come as binary data, need to decode to string var decoder = new TextDecoder(); return decoder.decode(fileAttri bute); } var caCert = decodeFileContent(input. ca_certificate); var clientCert = decodeFileContent(input. client_certificate); var clientKey = decodeFileContent(input. client_key); var sslOptions = { verify: true, caFile: caCert, // root certificate verifyHostname: false, certFile: clientCert, // client certificate keyFile: clientKey // client / private key }; var sslConfigured = http.setSSLOptions(sslOp tions);</pre> |
| Timeout | <pre>/** Timeout Configuration Signature: http.setTimeout(timeoutMs: number) Example: Set timeout to 2000 ms */</pre> | <pre>http.setTimeout(2000); // Set timeout for all requests to 2 seconds MIN_TIMEOUT =1000; // 1 second MAX_TIMEOUT =300000; // 5 minutes DEFAULT_TIMEOUT = 8000; // 8 seconds</pre> |

| Method | Signature | Sample Snippet |
|-------------------------|---|---|
| Connection Reuse | <pre>/* http.setConnectionReuse(enabled : boolean): boolean Controls whether HTTP connections are reused (pooled) or closed after each request. } □ Parameters enabled (boolean): true: Enables connection reuse (default behavior) false: Disables connection reuse — HTTP connection will be closed after each request □ Returns true on success false if setting failed (e.g., unsupported in the current environment)*/</pre> | <pre>function init() { http.setConnectionReuse(true); // by default its true mentioned this . }</pre> |

Known Limitations

- Avoid using `http.get/post/delete/put` within the `init()` call to prevent unexpected delays during initialization.
- When making API calls that handle large payloads, particularly for CRUD operations, refrain from logging the full response due to the logging buffer's size limit.
- Configure SSL options during the `init` call to ensure secure communication is established from the start.
- Be aware that the maximum payload size is limited to 1MB.

Create a Data Rule

Data rules define the flow of data, from connected assets to data destinations, without transformation.

Procedure

-
- Step 1** From the **Data Policy** tab, select **Data Rule** checkbox.
- Step 2** To configure the Data Rule, select a data from **Data Sampling Interval (ms)** drop-down list.

Data Policy Type *

☐ Data Logic ☒ Data Rule

Data Rule Configuration

Data Sampling Interval (ms)

Disabled



Disabled

100

250

500

1000

2000

5000

Deploy or undeploy pipelines

When deployed, a pipeline runs on the Cisco network device where the EI agent is installed.

Procedure

Step 1

To deploy a pipeline:

- Create a pipeline.
- Click **Deploy**.

The deployed pipeline is listed in the **Pipelines** area of the Cisco Edge Intelligence page.

Step 2

To undeploy a pipeline, in the **Pipeline** section, select a pipeline and click **Undeploy**.

When you undeploy a pipeline, it is entirely removed from Cisco Edge Intelligence. If you wish to retain a copy of the configuration, save the pipeline configuration as a template before you undeploy the pipeline.

View health status

To track the health status of an EI agent, select a pipeline from the list of pipelines.

Procedure

- Step 1** From the left pane, click any **Pipeline** for which you want to view the details.
- Step 2** Click **Health Status** tab. A combined list log of pipeline overview, source status, and destination status appears for the EI agent.
- Step 3** Click on any status tab to preview each log report.

| Column Header | Description |
|--------------------|---|
| Health Status | Displays the overall health of the data pipeline, indicating errors in source or destination connections. |
| Pipeline Status | Shows the current status of the data pipeline. For example, Error and so on. |
| Source Status | Indicates the connection status (online or offline) of the source asset. |
| Destination Status | Reflects the status (online or offline) of the data destination. |