CHIPKIN

HTTP Trigger Driver FS-8705-111

Chipkin - Enabling Integration

salesgroup1@chipkin.com

Tel: +1 866 383 1657

Driver Version:
Document Revision: 4

TABLE OF CONTENTS

1 HTTP Trigger Description

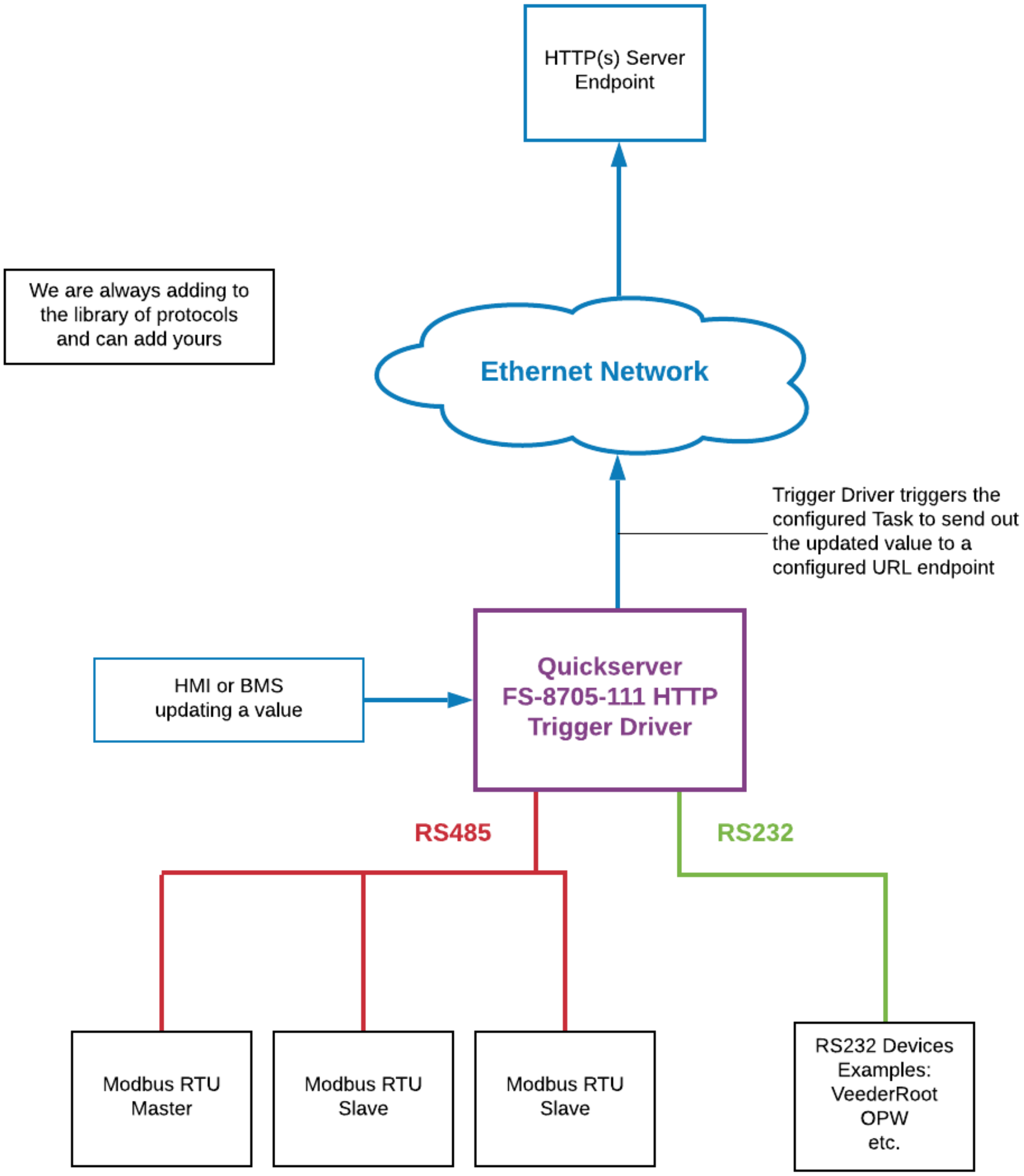
The HTTP Trigger Driver allows the FieldServer to push updated values to any any accessible HTTP(s) API over Ethernet, using RESTful HTTP(s) request methods. The HTTP Trigger Driver supports pushing values to multiple HTTP(s) server endpoints.

The HTTP Trigger Driver is a client driver that pushing data to a configured URL endpoint. The FieldServer stores values to be mapped to other protocols or simply to be viewed. When configured, these change of values in the data arrays trigger a configured Task to send this changed value to a corresponding configured Node URL endpoint.

The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

2 Connection Description

This block diagram shows data being served using other protocols like Modbus® RTU/TCP, and BACnet®. The FieldServer can use the HTTP Trigger Driver to push these change of values out to various HTTP(s) API endpoints.


HTTP(s) Server Endpoint
We are always adding to the library of protocols and can add yours
Ethernet Network
Trigger Driver triggers the configured Task to send out the updated value to a configured URL endpoint
HMI or BMS updating a value
Quickserver FS-8705-111 HTTP Trigger Driver
RS485
RS232
Modbus RTU Master
Modbus RTU Slave
Modbus RTU Slave
RS232 Devices Examples: VeederRoot OPW etc.

3 HTTP Trigger Driver Configuration

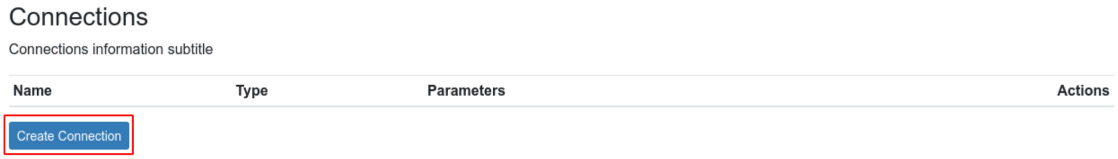
To configure the HTTP Trigger Driver, from the home page, visit the following link:
http://{IP_ADDRESS}/chipkin/ui/#/HTTPTriggerDriver.

To configure the FieldServer, follow the instructions below to add a Connection (physical port), Nodes (HTTP(s) Endpoint Urls and Ports to push data to), and finally Tasks (the data to to listen for changes to).

3.1 Create Connection

To set up the FieldServer HTTP Trigger Driver, first create a connection. The connection contains information about the physical port to use.

1. Click on the “Create Connection” button to open the Create Connection form.



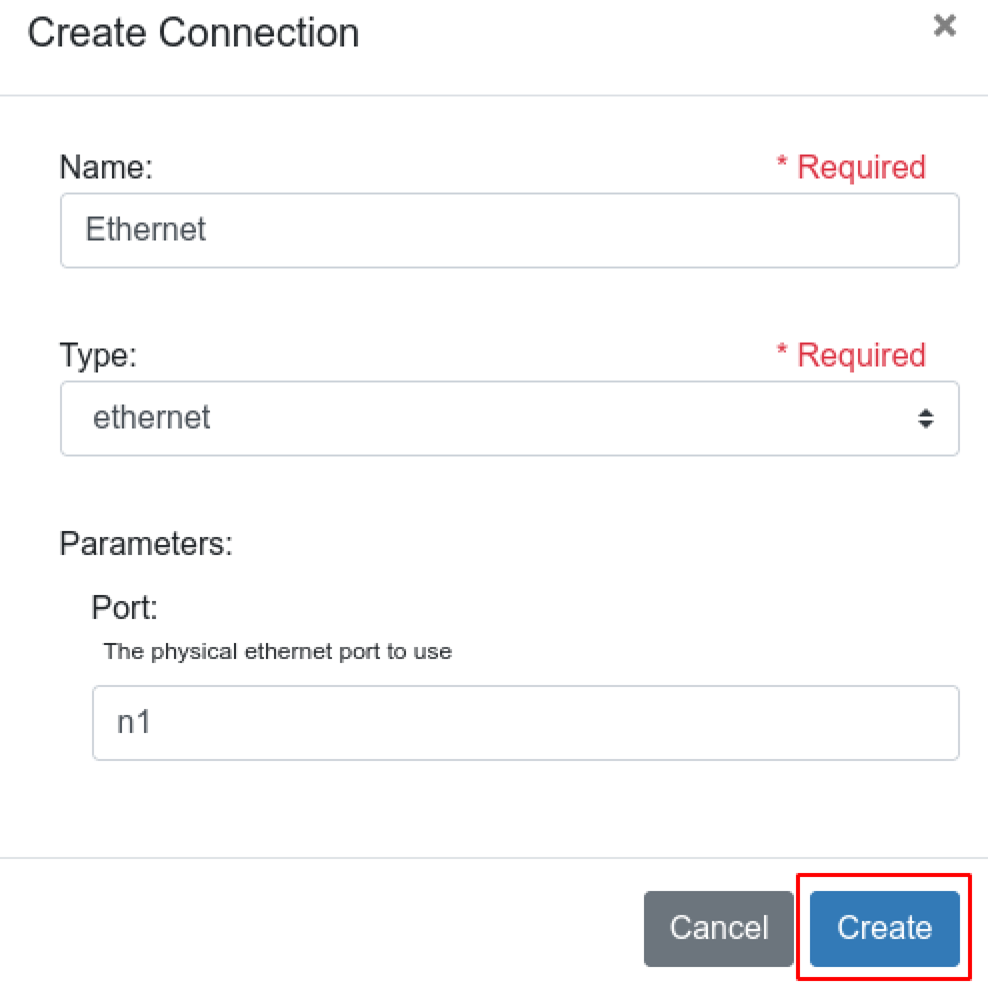
Create Connection

2. The fields are as follows:

COLUMN TITLE	FUNCTION	LEGAL VALUES
Name	Name of the connection, used internally as an identifier for Nodes.	Text, must be unique
Type	The type of connection this is. Currently, only ethernet is supported.	**ethernet**
Parameters: Port	The physical port on the FieldServer to use.	**n1**

* Bolded values are defaults

3. Click the “Save” button to add the connection.



Create Connection

Name: * Required

Ethernet

Type: * Required

ethernet

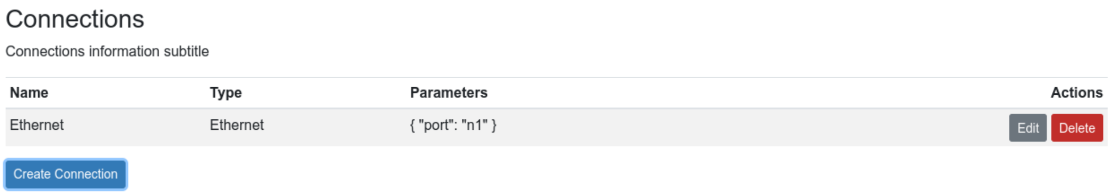
Parameters:

Port:

The physical ethernet port to use

n1

4. If successful, the new entry will be populated in the Connections table:



Connections

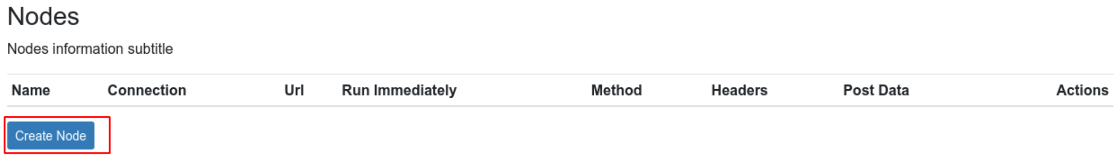
Connections information subtitle

Name	Type	Parameters	Actions
Ethernet	Ethernet	{ "port": "n1" }	Edit Delete

3.2 Create Node

Follow the instructions below to configure the device to send data to various HTTP(s) API endpoints.

5. Click on the Create Node button to open the Create Node form.



Nodes

Nodes information subtitle

Name	Connection	Url	Run Immediately	Method	Headers	Post Data	Actions

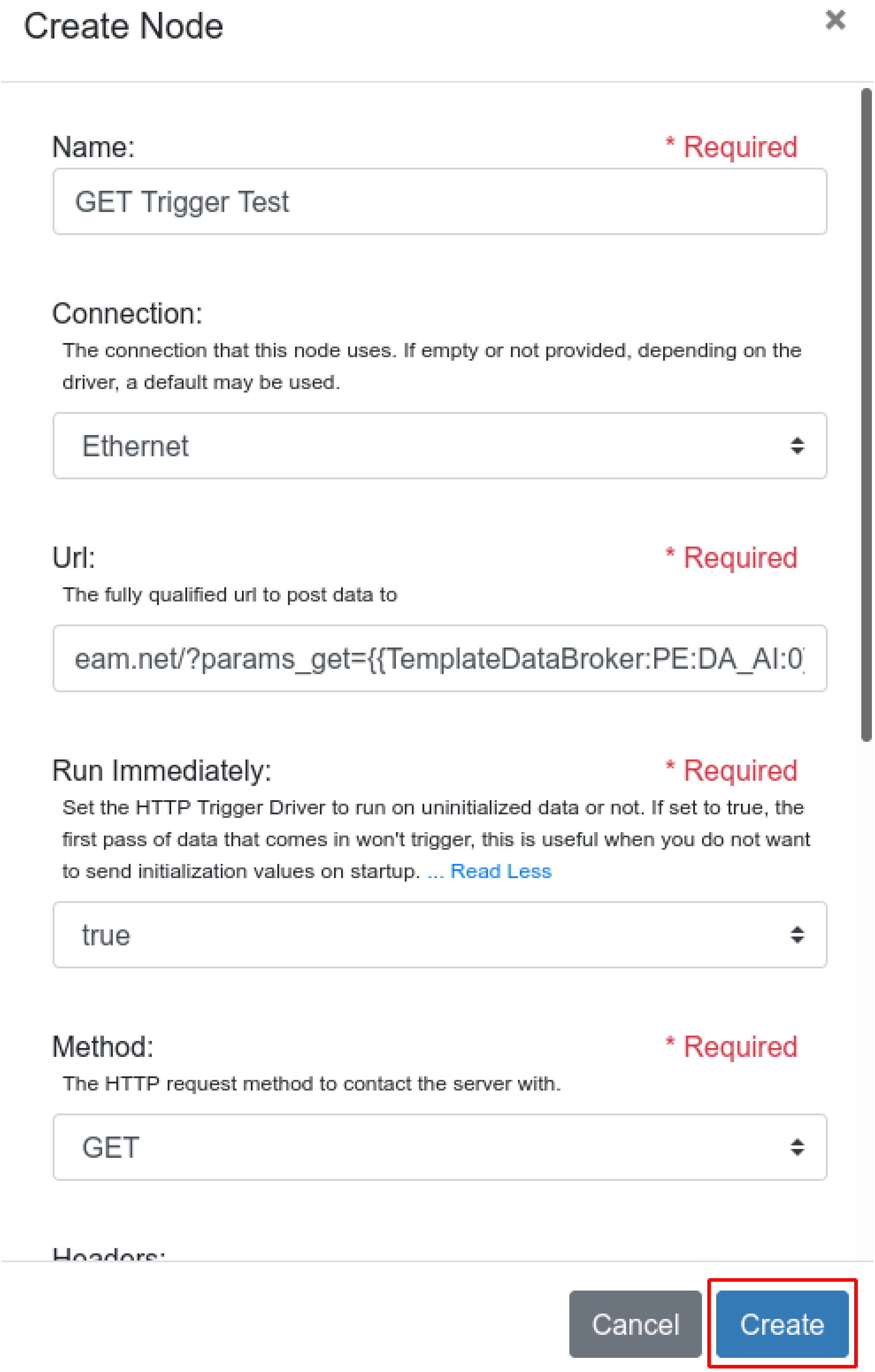
6. Fill out the fields in the form. The fields are as follows:

COLUMN TITLE	FUNCTION	LEGAL VALUES
Name	A name given to this Node. A Task will reference this Node by this Name. Must be unique.	Text (string)
Connection	The name of the FieldServer's physical port, linked via the Connection.	Text (Use the name of the Connection created in the previous section)
Run Immediately	Whether or not the HTTP Trigger Driver should trigger on initialization data. As an example, when the FieldServer starts up, all values are initialized as 0, and when data comes in for the first time, the Field only watch for subsequent data change events. Defaults to `false`	True - Listens for data event changes immediately after initializing the data arrays. **False** - Skips listening for data within the initialization process, and only triggers once data has populated a data array offset for the first time.
URL	A fully qualified HTTP(s) URL to send data to.	Text (string) - `https://example/server/api/location`The HTTP Trigger Driver supports internal templating in order to add values to the request. See Examples {Section ##} for full examples of POST Triggers and templating.
Method	The HTTP REST request method the server accepts. Only GET and POST are currently supported. Consult the external API for which method to use.	**GET** - `https://example/api/request?param=1` See Examples {Section ##} for full examples of all GET Triggers and template styles.POST - `{"param": 1}` See Examples {Section ##} for full examples of POST Triggers and templating.
Headers	HTTP(s) headers to be sent with the request	Text (object) - Headers as a JSON object.Example: `{"Content-Type":`

		`"application/json}`
Post Data	HTTP(s) POST data to be sent with the request. Only available when POST is selected as the Method. Ignored on GET requests	Text (string) - Must conform to the servers Content-Type. See Examples {Section ##} for full examples of POST templating.

Examples using Templating System (See Section 5 for more details, and Section 8 for a working example)

GET Example Using the Template Data Broker (See Section 5, and Section 8 for more details)



Create Node

Name: * Required

GET Trigger Test

Connection:

The connection that this node uses. If empty or not provided, depending on the driver, a default may be used.

Ethernet

Url: * Required

The fully qualified url to post data to

eam.net/?params_get={{TemplateDataBroker:PE:DA_AI:0

Run Immediately: * Required

Set the HTTP Trigger Driver to run on uninitialized data or not. If set to true, the first pass of data that comes in won't trigger, this is useful when you do not want to send initialization values on startup. ... Read Less

true

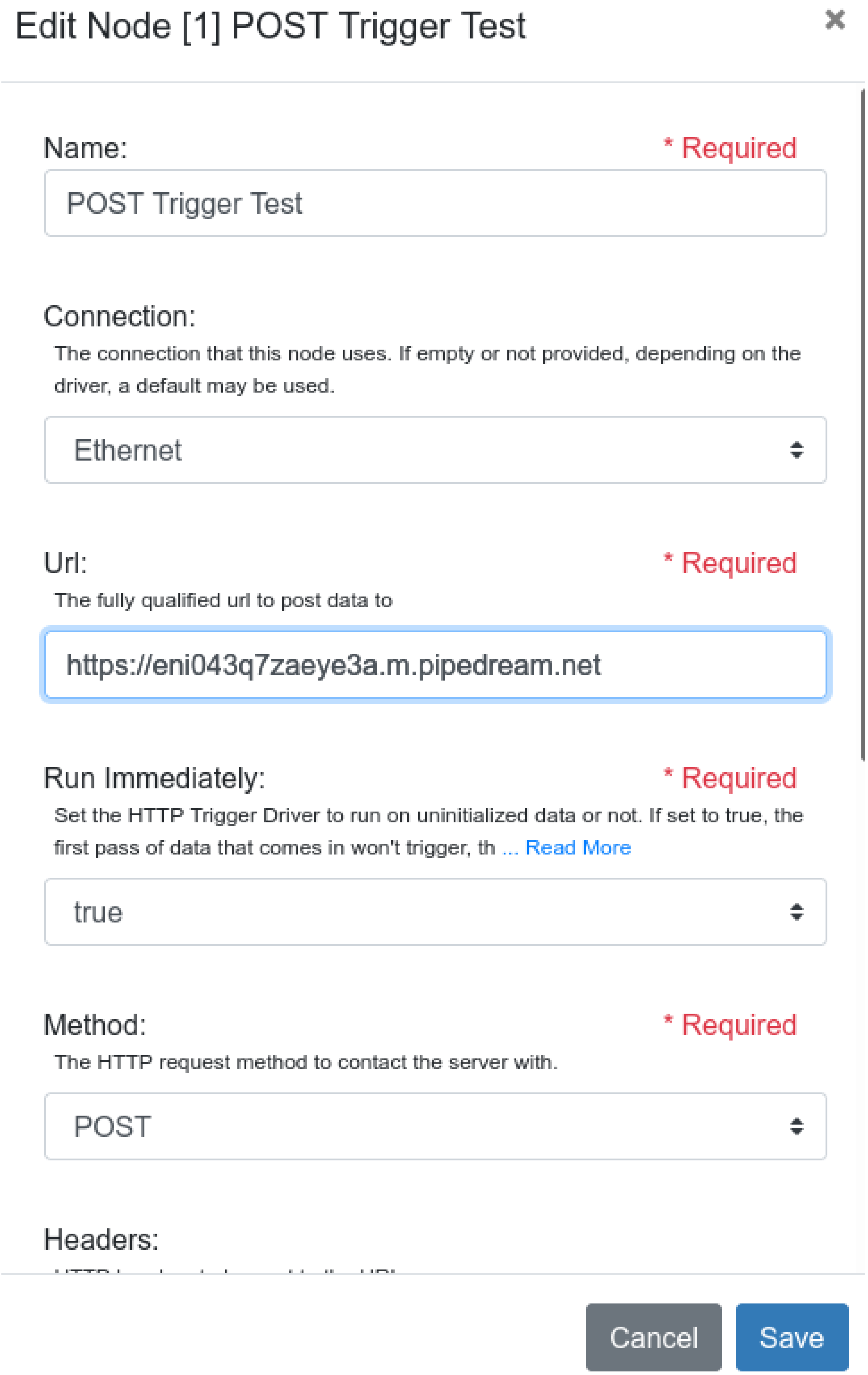
Method: * Required

The HTTP request method to contact the server with.

GET

Create Node Form Filled

POST Example Using Template Data Broker (See Section 5, and Section 8 for more details)



Edit Node [1] POST Trigger Test

Name: * Required

POST Trigger Test

Connection:

The connection that this node uses. If empty or not provided, depending on the driver, a default may be used.

Ethernet

Url: * Required

The fully qualified url to post data to

https://eni043q7zaeye3a.m.pipedream.net

Run Immediately: * Required

Set the HTTP Trigger Driver to run on uninitialized data or not. If set to true, the first pass of data that comes in won't trigger, th ... Read More

true

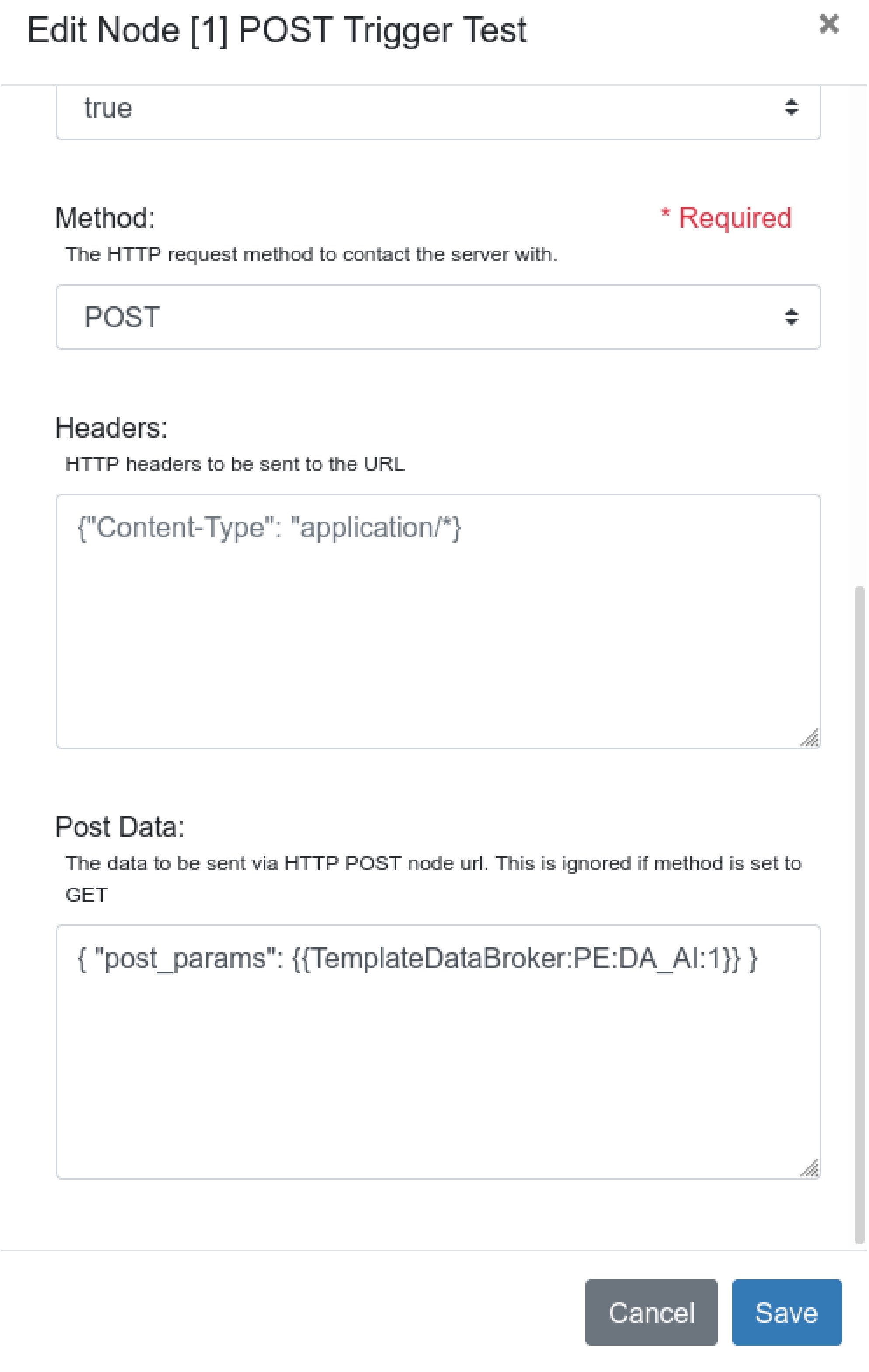
Method: * Required

The HTTP request method to contact the server with.

POST

Headers:

Create Node Form Filled 1



Edit Node [1] POST Trigger Test

true

Method: * Required

The HTTP request method to contact the server with.

POST

Headers:

HTTP headers to be sent to the URL

{"Content-Type": "application/*}

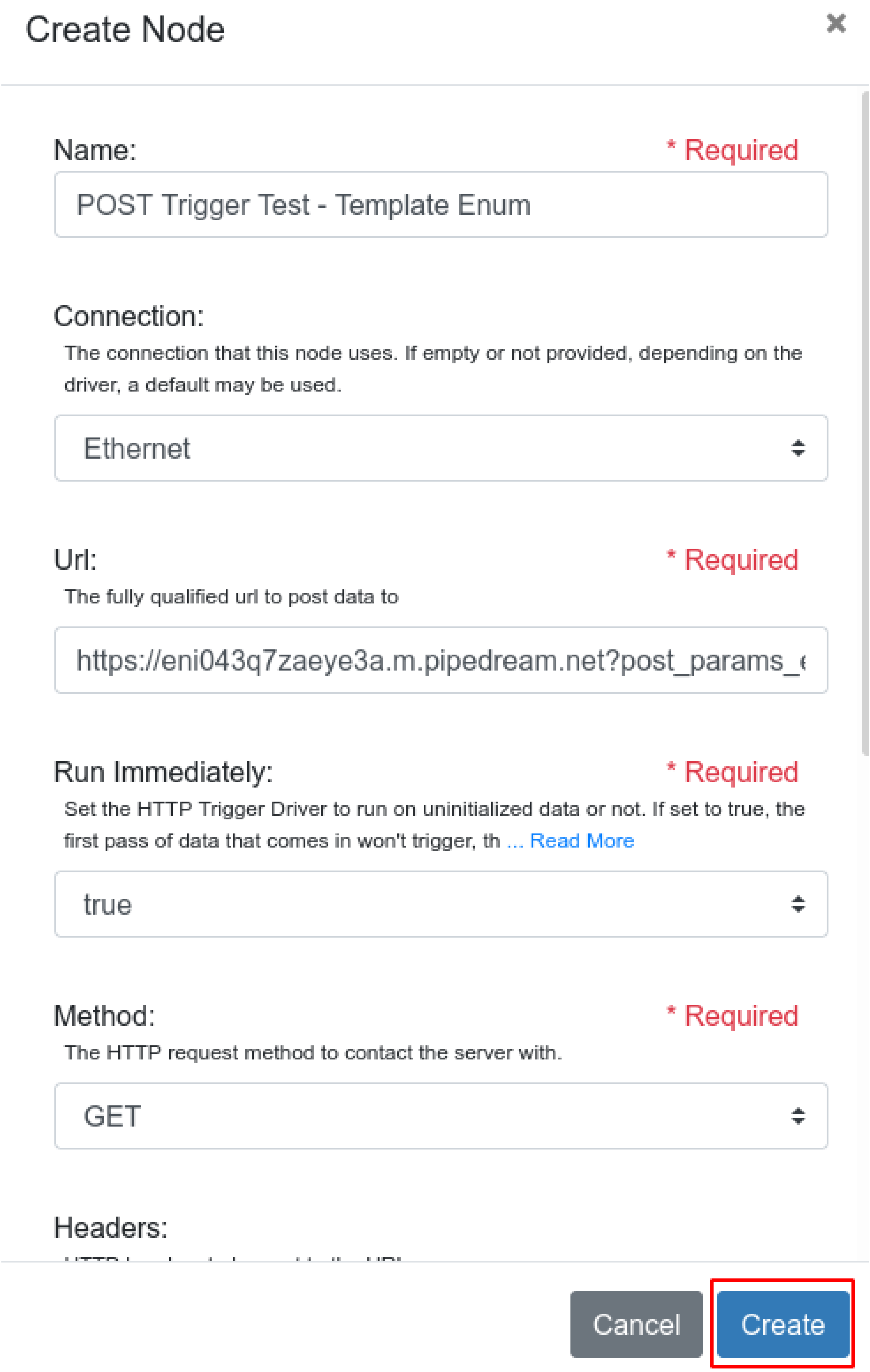
Post Data:

The data to be sent via HTTP POST node url. This is ignored if method is set to GET

{ "post_params": {{TemplateDataBroker:PE:DA_AI:1}} }

Create Node Form Filled 2

GET Example Using Template Enumerations (See Section 5, and Section 8 for more details)



Create Node

Name: * Required

POST Trigger Test - Template Enum

Connection:

The connection that this node uses. If empty or not provided, depending on the driver, a default may be used.

Ethernet

Url: * Required

The fully qualified url to post data to

https://eni043q7zaeye3a.m.pipedream.net?post_params_e

Run Immediately: * Required

Set the HTTP Trigger Driver to run on uninitialized data or not. If set to true, the first pass of data that comes in won't trigger, th ... Read More

true

Method: * Required

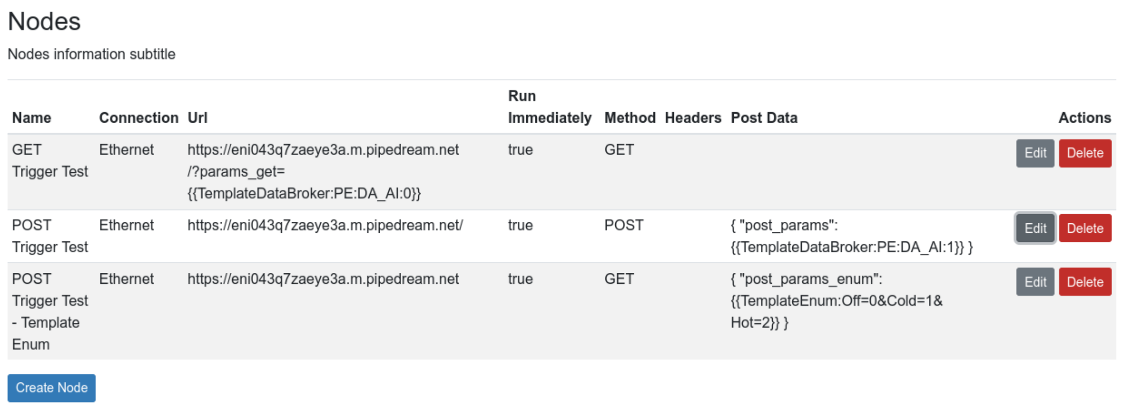
The HTTP request method to contact the server with.

GET

Headers:

Create Node Form Filled 3

7. If successful, the new entry will be populated in the Nodes table



Nodes

Nodes information subtitle

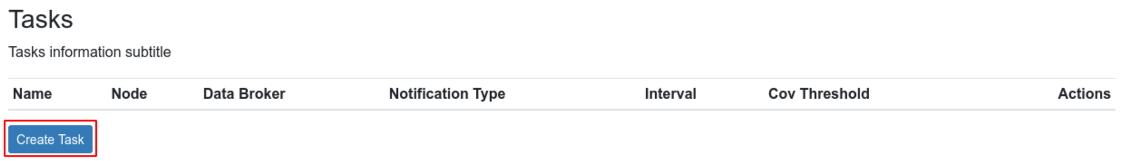
Name	Connection	Url	Run Immediately	Method	Headers	Post Data	Actions
GET Trigger Test	Ethernet	https://eni043q7zaeye3a.m.pipedream.net /?params_get= {{TemplateDataBroker:PE:DA_AI:0}}	true	GET			Edit Delete
POST Trigger Test	Ethernet	https://eni043q7zaeye3a.m.pipedream.net/	true	POST		{ "post_params": {{TemplateDataBroker:PE:DA_AI:1}} }	Edit Delete
POST Trigger Test - Template Enum	Ethernet	https://eni043q7zaeye3a.m.pipedream.net	true	GET		{ "post_params_enum": {{TemplateEnum:Off=0&Cold=1& Hot=2}} }	Edit Delete

Create Nodes Filled

3.3 Create Task

Create tasks in order to send updated values to any HTTP(s) endpoint, or REST API.

1. Click on the "Create Task" button to open the Create Task form.



Tasks

Tasks information subtitle

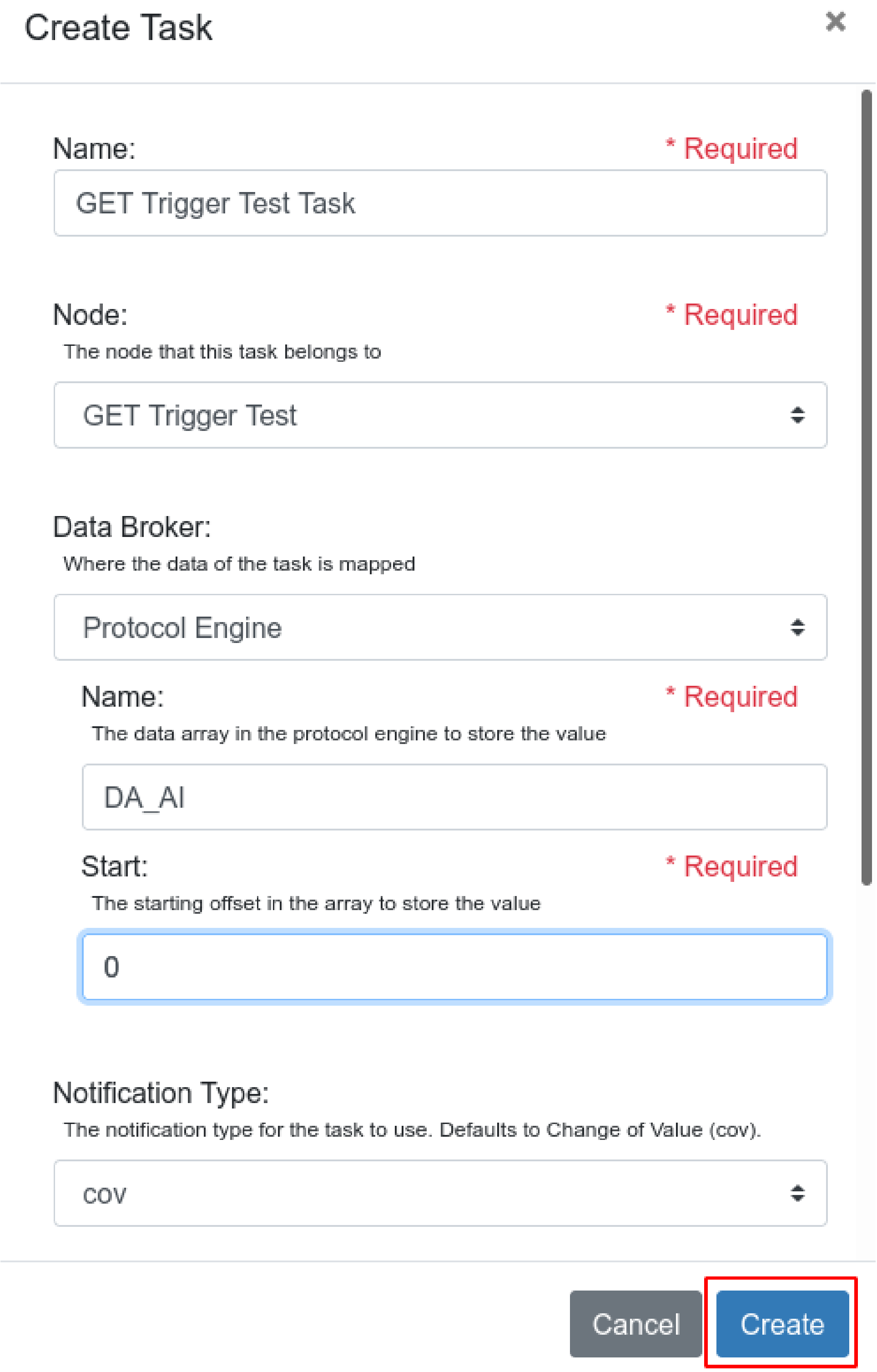
Name	Node	Data Broker	Notification Type	Interval	Cov Threshold	Actions

2. Fill out the fields in the form. The fields are as follows:

COLUMN TITLE	FUNCTION	LEGAL VALUES
Name	The unique name of the Task.	Text (string) – Must be unique
Node	The Node that this task will reference when making requests.	Text (string)
Data Broker	The FieldServer data array map used to store data	**Protocol Engine** - The driver level data array map, configurable from the config.csv Application Engine - The in memory data object.
Data Broker PE:Name	The name of the data array to map values to.	Text (string) - DA_AI
Data Broker PE:Start	The offset within the data array to map values to.	Integer - 0

Data Broker AE:Path	The path into the AE data store where the values are mapped.	Text (string) - `example/path/to/location`

GET Task



Create Task

Name: * Required

GET Trigger Test Task

Node: * Required

The node that this task belongs to

GET Trigger Test

Data Broker:

Where the data of the task is mapped

Protocol Engine

Name: * Required

The data array in the protocol engine to store the value

DA_AI

Start: * Required

The starting offset in the array to store the value

0

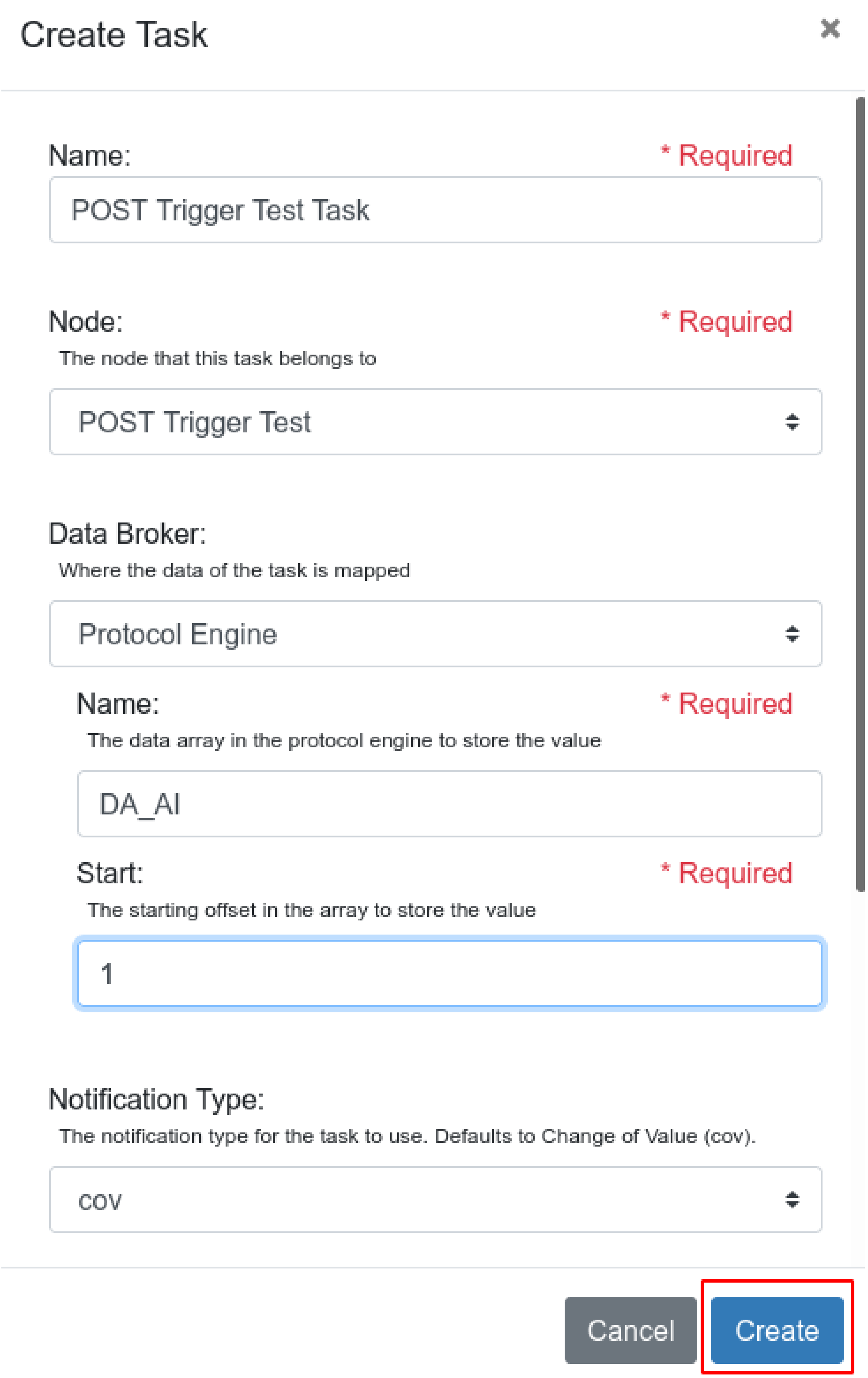
Notification Type:

The notification type for the task to use. Defaults to Change of Value (cov).

cov

GET Task

POST Task



Create Task

Name: * Required

POST Trigger Test Task

Node: * Required

The node that this task belongs to

POST Trigger Test

Data Broker:

Where the data of the task is mapped

Protocol Engine

Name: * Required

The data array in the protocol engine to store the value

DA_AI

Start: * Required

The starting offset in the array to store the value

1

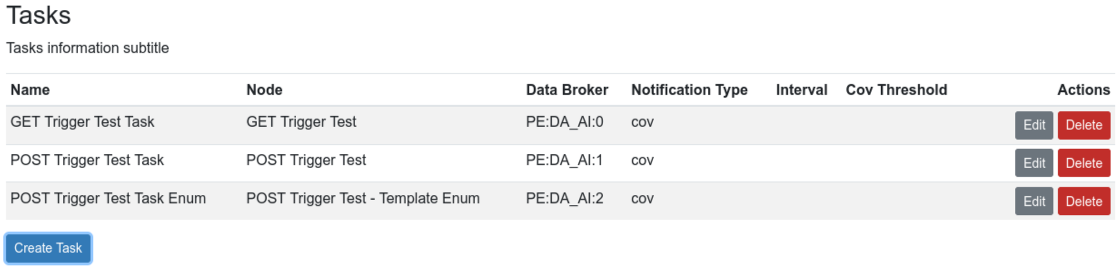
Notification Type:

The notification type for the task to use. Defaults to Change of Value (cov).

cov

POST Task

3. If successful, the new entry will be populated in the Tasks table



Tasks

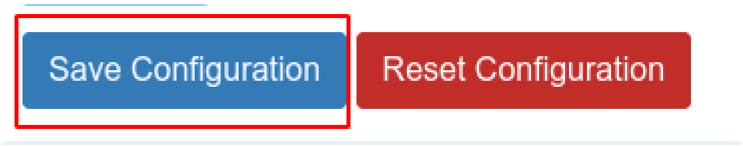
Tasks information subtitle

Name	Node	Data Broker	Notification Type	Interval	Cov Threshold	Actions
GET Trigger Test Task	GET Trigger Test	PE:DA_AI:0	cov			Edit Delete
POST Trigger Test Task	POST Trigger Test	PE:DA_AI:1	cov			Edit Delete
POST Trigger Test Task Enum	POST Trigger Test - Template Enum	PE:DA_AI:2	cov			Edit Delete

Create Tasks Filled

3.4 Saving the Server Configuration

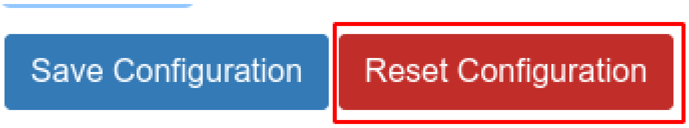
When the configuration is complete, click on the “Save Configuration” button to save all of the updates and changes. For the configuration to take effect, reboot the system.



Save Configuration

3.5 Resetting the Configuration

To clear the configuration and start over, click the “Reset Configuration” button. Then follow the instructions in the sections above to create new connections, nodes, and tasks.



Reset Configuration

4 HTTP(s) Request Methods and Data Templating Examples

The HTTP Trigger Driver allows URLS and Post Data to contain either static data, or dynamic data taken from the configured FieldServer data arrays to send to any API.

Through the REST request method, we offer different ways to send data. Either direct through the URL using GET, or through the Nodes POST or PUT data.

4.1 GET

A GET request requires a fully qualified endpoint that the data will ultimately be transferred to.

Generally, GET requests are read-only requests, usually for obtaining data from a server. API's can differ wildly, though, and some will allow a user to set data via a GET request. If you're using a GET request to retrieve data, please see our HTTP Trigger Driver Manual for more information, as this guide strictly focuses on the use cases where the HTTP Trigger Driver sends data from the FieldServer to an external API.

https://example.server/api?set&static=1 - This request will always send a static variable to the API. This is useful when a user just wants to know if a variable has changed, rather than what the value has changed to, such as a Variable Updated metric. Please reference your API documentation for GET configuration.

4.2 POST

POST requests are generally used to create new objects through an API. In addition, most API's support updating objects through POST. A POST request will require a URL endpoint, a Data payload, and an optional Headers object. Please reference your API documentation for POST configuration.

- **URL** - `https://example.server/api` - The URL is the endpoint that the data will be sent to.
- **Header** - `{"Content-Type": "application/json"}` - Headers may be required, as the API may specify a content type, or content size, or authentication via this Headers field.
- **Post Data** - `{"parameter": 123}` - The data to be sent to the API endpoint. Please reference your API documentation for POST configuration.

4.3 PUT

See Section 4.2, as they are functionally the same. And depending on the API, interchangeable. Please reference your API's documentation for PUT configuration.

5 Templating

A templating system was created in order to send data from the FieldServer to an external API. Since the Tasks are only invoked through a data change event, only the updated value is sent with the Node request.

Currently, we support two ways to template data to send it to an API via the URL.

5.1 Data Broker Templating

The Data Broker Template allows the HTTP Trigger Driver to get the value directly from the Data Arrays.

GET - `https://example.server/api/?get_value={{TemplateDataBroker:PE:DA_AI:0}}`

POST - `https://example.server/api/` - `{"param": "test", "value": {{TemplateDataBroker:PE:DA_AI:0}}`

5.2 Enumeration Templating

The Enumeration Template allows the HTTP Trigger Driver to get the value from a Data Array and convert it to it's string representation within the external API. Since this value is converted to a string, be sure to surround the enumeration in quotes if needed.

GET - https://example.server/api/?{{TemplateEnum:0=Off&1=On}}

POST - https://example.server/api/ - {"param": "test", "value": "{{TemplateEnum:0=Off&1=On}}"

This section will be updated as we release more ways of templating data.

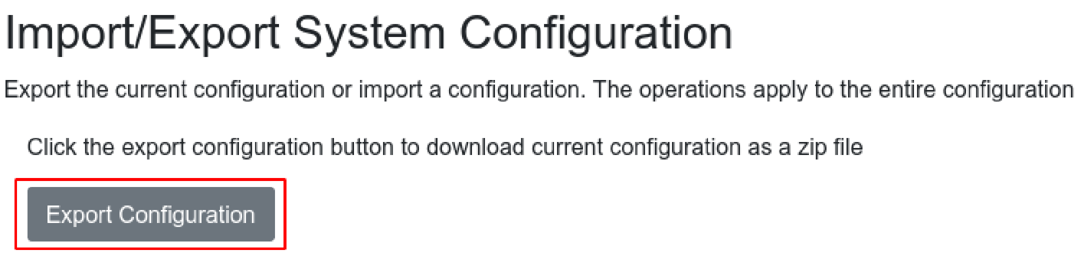
6 Importing and Exporting Configurations

It is possible to export the current configuration to back it up or simply to make some edits. Users can also import either the entire configuration via a zip file or a PE (Protocol Engine) configuration.

6.1 How to Export the Configuration

1. Goto the system configuration page http://{IP_ADDRESS}/chipkin/ui/#/chipkinLicenseDriver

2. Click the Export Configuration button.



Import/Export System Configuration

Export the current configuration or import a configuration. The operations apply to the entire configuration

Click the export configuration button to download current configuration as a zip file

Export Configuration

6.2 How to Import the Configuration

The file to import the configuration must be a zip file. The zip file should contain the following folders:

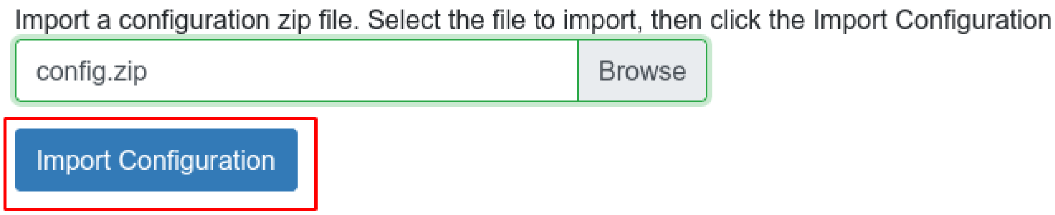
- ae - this folder contains any configuration files for the ae configuration
- documents - this folder contains any driver specific documents. For example, license product keys, etc.
- pe - this folder contains one config.csv file for the pe configuration.

To make sure the folder directory is correct, do an Export first, then extract the files, edit them, then zip them up again.

To import the configuration:

3. Goto the system configuration page http://{IP_ADDRESS}/chipkin/ui/#/chipkinConfiguration
4. Click the "Browse" button in the "Import/Export System Configuration" section and select the zip file containing the configuration to import.

5. Click the "Import Configuration" button and wait for the configuration to finish importing.
6. If successful, a success message will appear prompting a reboot of the FieldServer for the changes to take effect.



Import a configuration zip file. Select the file to import, then click the Import Configuration

config.zip

Import Configuration

6.3 Example PR Configuration

Bridge

Title

Example

```
Data_Arrays
Data_Array_Name , Data_Format   , Data_Array_Length
DA_AI           , float         , 200
```

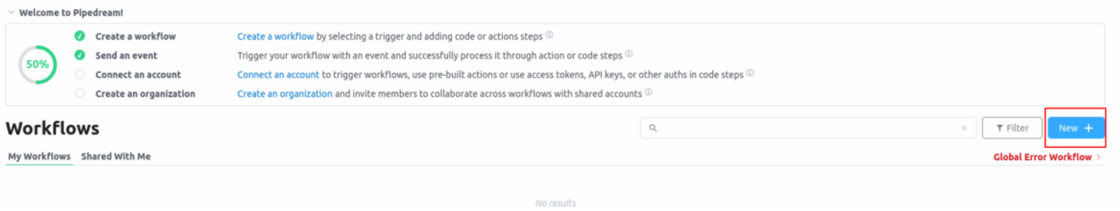
6.4 Example AE Configuration

```
{
    "ae": {
        "httpTriggerDriver": {
        "connections": [
            {
            "type": "ethernet",
            "name": "Ethernet",
            "parameters": { "port": "n1" }
            }
        ],
        "nodes": [
            {
                "connection": "Ethernet",
                "method": "GET",
                "name": "Example Node GET - Template Data Broker",
                "runImmediately": true,
                "url":
"https://{PIPEDREAM_ENDPOINT}?get_param={{TemplateDataBroker:PE:DA_AI:0}}",
                "headers": "",
                "postData": ""
            },
            {
                "connection": "Ethernet",
                "method": "POST",
                "name": "Example Node POST - Template Data Broker",
                "runImmediately": true,
                "url": "https://{PIPEDREAM_ENDPOINT}",
                "headers": "",
                "postData": "{\"post_param\": {{TemplateDataBroker:PE:DA_AI:1}}}"
            },
            {
                "connection": "Ethernet",
                "method": "POST",
                "name": "Example Node POST - Template Enum",
                "runImmediately": true,
                "url": "https://{PIPEDREAM_ENDPOINT}",
                "headers": "",
                "postData": "{\"post_param_enum\":
```

```
\"{{TemplateEnum:0=Off&1=Cold&2=Hot}}}\""
            }
        ],
        "tasks": [
            {
            "node": "Example Node GET - Template Data Broker",
            "notificationType": "cov",
            "name": "Example Task GET - Template Data Broker",
            "dataBroker": { "pe": { "Name": "DA_AI", "Start": 0 } },
            "covThreshold": 1
            },
            {
            "node": "Example Node POST - Template Data Broker",
            "notificationType": "cov",
            "name": "Example Task POST - Template Data Broker",
            "dataBroker": { "pe": { "Name": "DA_AI", "Start": 1 } },
            "covThreshold": 1
            },
            {
            "node": "Example Node POST - Template Enumerations",
            "notificationType": "cov",
            "name": "Example Task POST - Template Enumerations",
            "dataBroker": { "pe": { "Name": "DA_AI", "Start": 2 } },
            "covThreshold": 1
            }
        ]
        }
    ],
    "nodes": [
        {
            "connection": "Ethernet",
            "method": "GET",
            "name": "Example Node GET - Template Data Broker",
            "runImmediately": true,
            "url":
"https://{PIPEDREAM_ENDPOINT}?get_param={{TemplateDataBroker:PE:DA_AI:0}}",
            "headers": "",
            "postData": ""
        },
        {
            "connection": "Ethernet",
            "method": "POST",
            "name": "Example Node POST - Template Data Broker",
            "runImmediately": true,
            "url": "https://{PIPEDREAM_ENDPOINT}",
            "headers": "",
            "postData": "{\"post_param\": {{TemplateDataBroker:PE:DA_AI:1}}}"
        },
        {
            "connection": "Ethernet",
            "method": "POST",
            "name": "Example Node POST - Template Enum",
            "runImmediately": true,
            "url": "https://{PIPEDREAM_ENDPOINT}",
            "headers": "",
            "postData": "{\"post_param_enum\":
\"{{TemplateEnum:0=Off&1=Cold&2=Hot}}}\""
        }
    ],
    "tasks": [
        {
        "node": "Example Node GET - Template Data Broker",
        "notificationType": "cov",
        "name": "Example Task GET - Template Data Broker",
        "dataBroker": { "pe": { "Name": "DA_AI", "Start": 0 } },
        "covThreshold": 1
        },
        {
        "node": "Example Node POST - Template Data Broker",
        "notificationType": "cov",
        "name": "Example Task POST - Template Data Broker",
        "dataBroker": { "pe": { "Name": "DA_AI", "Start": 1 } },
        "covThreshold": 1
        },
        {
        "node": "Example Node POST - Template Enumerations",
        "notificationType": "cov",
        "name": "Example Task POST - Template Enumerations",
        "dataBroker": { "pe": { "Name": "DA_AI", "Start": 2 } },
        "covThreshold": 1
        }
    ]
    }
}
```

7 Working Examples

1. Create an endpoint using Pipedream, to test against using the HTTP Trigger Driver.
   For now, we just need to login, for the service to create an endpoint to send data to.



Welcome to Pipedream!

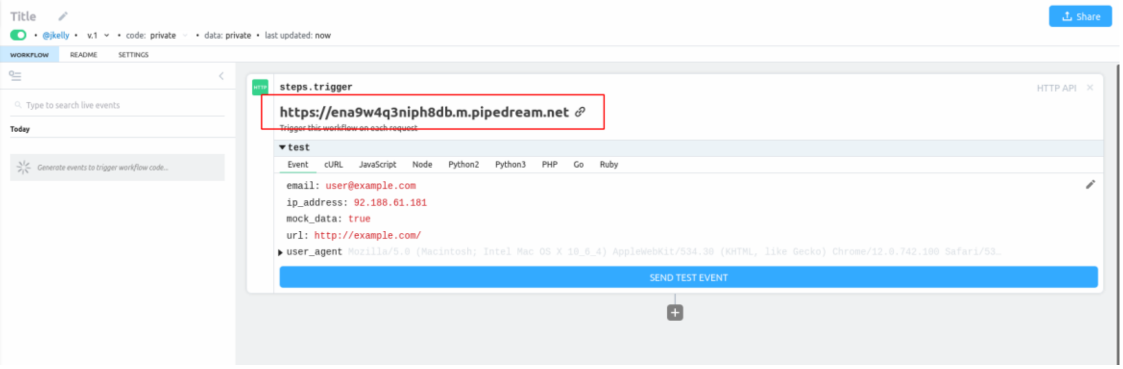
50%

- Create a workflow — Create a workflow by selecting a trigger and adding code or actions steps
- Send an event — Trigger your workflow with an event and successfully process it through action or code steps
- Connect an account — Connect an account to trigger workflows, use pre-built actions or use access tokens, API keys, or other auths in code steps
- Create an organization — Create an organization and invite members to collaborate across workflows with shared accounts

Workflows

New Pipedream Test

2. Replace `{PIPEDREAM_ENDPOINT}` in the example AE configuration with the api endpoint from pipedream (eg: `eni043q7zaeye3a.m.pipedream.net`)



Title
@jkelly • v.1 • code: private • data: private • last updated: now
Share
WORKFLOW
README
SETTINGS
Type to search live events
Today
Generate events to trigger workflow code...
HTTP API

steps.trigger

https://ena9w4q3niph8db.m.pipedream.net

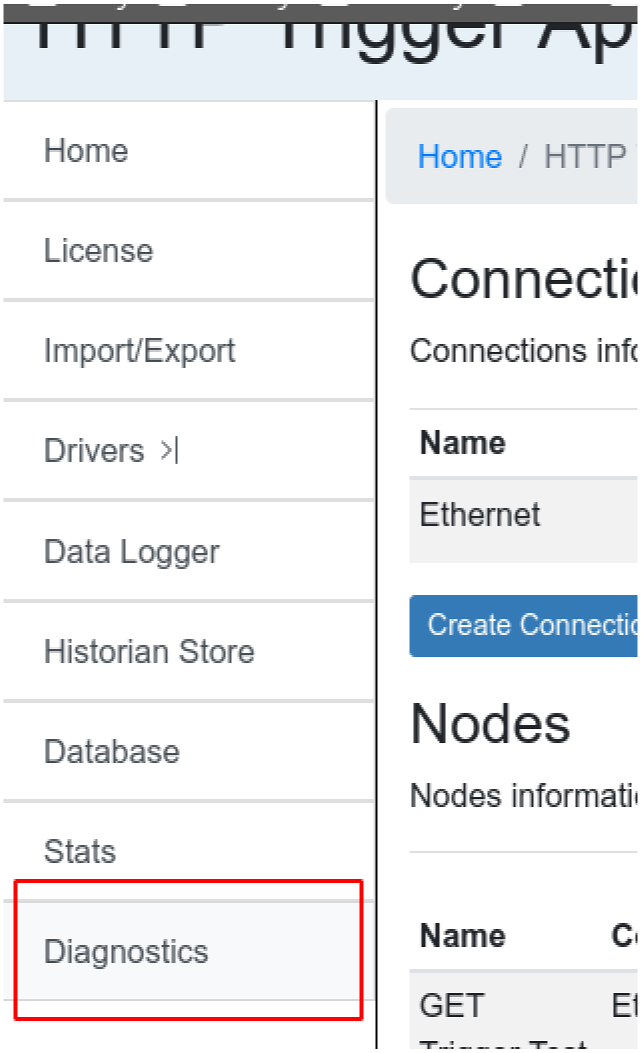
Trigger this workflow on each request

test
Event
cURL
JavaScript
Node
Python2
Python3
PHP
Go
Ruby

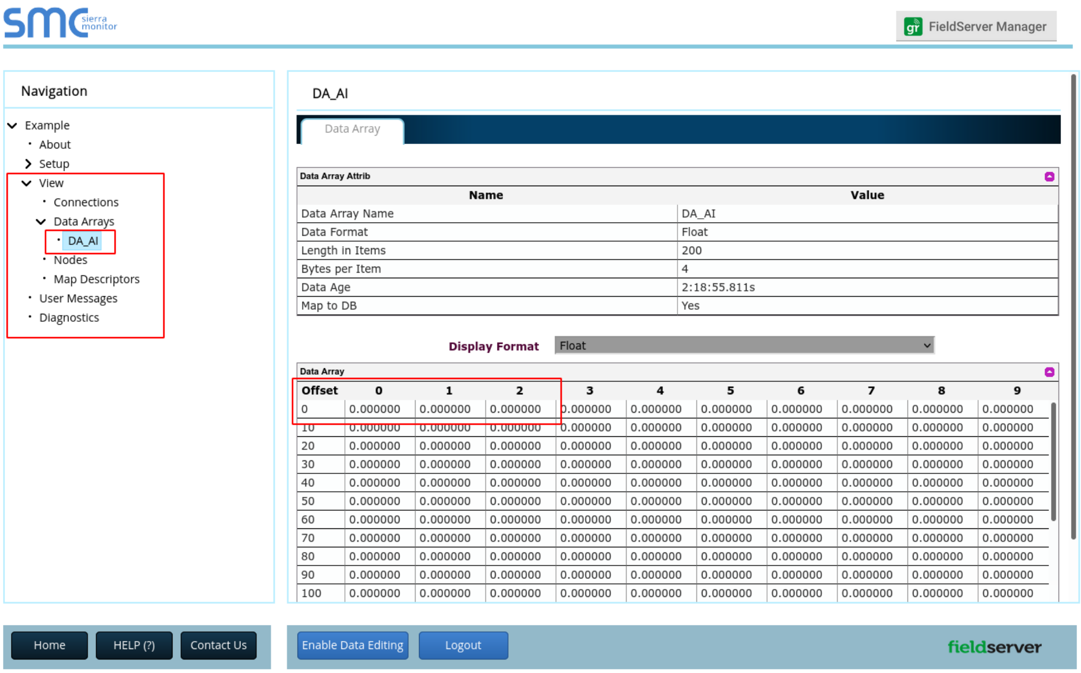
```
email: user@example.com
ip_address: 92.188.61.181
mock_data: true
url: http://example.com/
user_agent Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_4) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.100 Safari/53…
```

Pipedream Endpoint

3. Follow the steps for Importing a configuration outlined in Section 7.1 and 7.2 using the Example Configurations in Section 7.3
4. Navigate to the FieldServer UI, and poke a value into the DA_AI table



Click Diagnostics



DA_AI

Data Array Attrib

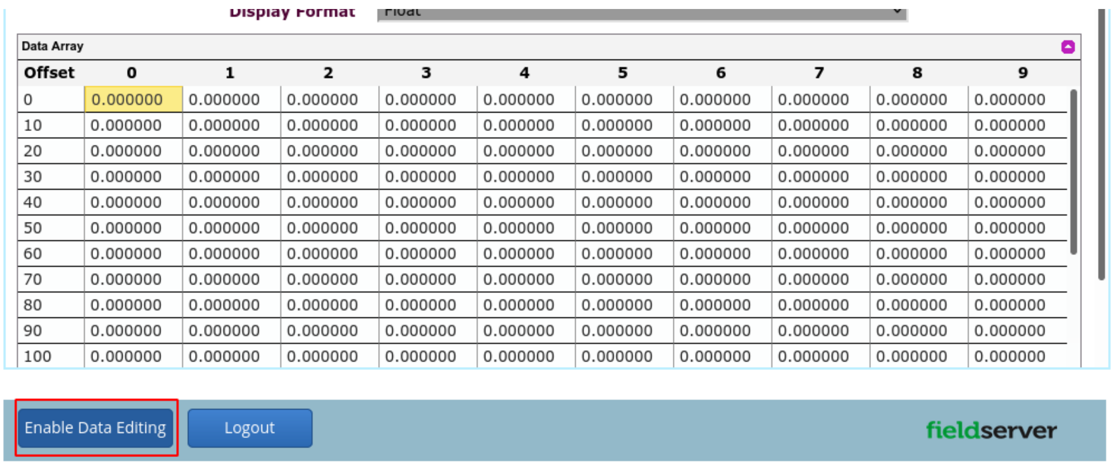
Name	Value
Data Array Name	DA_AI
Data Format	Float
Length in Items	200
Bytes per Item	4
Data Age	2:18:55.811s
Map to DB	Yes

Display Format: Float

Data Array

Offset	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
20	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
30	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
40	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
60	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
70	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
80	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
90	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
100	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Diagnostics Show Offsets

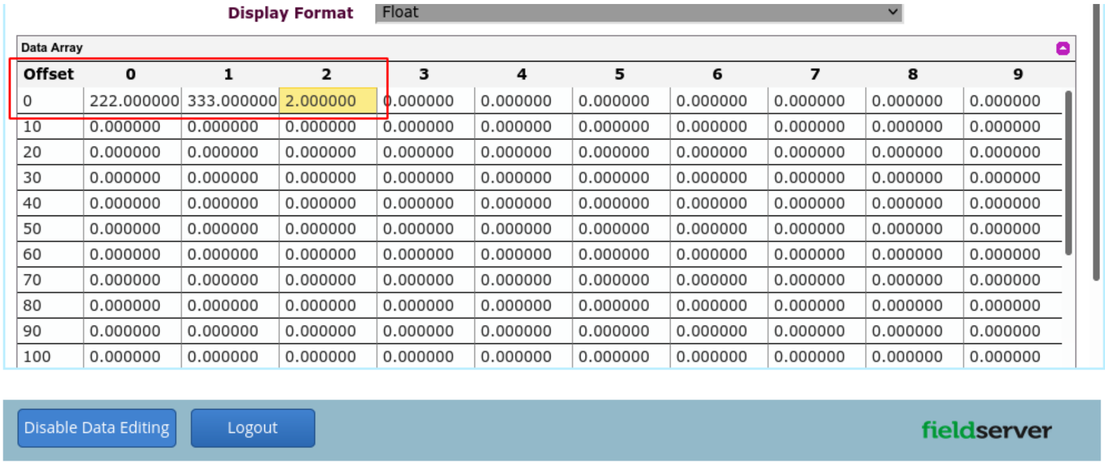


Display Format: Float

Data Array

Offset	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
20	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
30	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
40	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
60	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
70	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
80	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
90	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
100	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Diagnostics Enable Editing



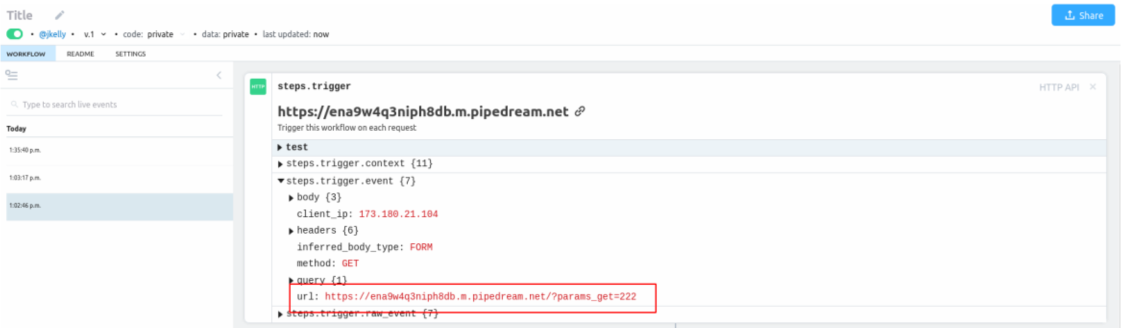
Display Format: Float

Data Array

Offset	0	1	2	3	4	5	6	7	8	9
0	222.000000	333.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
20	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
30	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
40	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
60	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
70	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
80	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
90	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
100	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Diagnostics Update Offsets

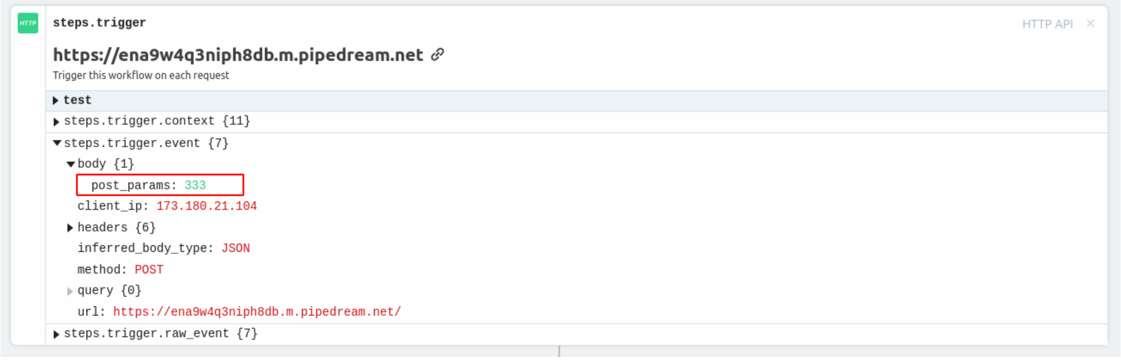
5. See the request come through within Pipedream.



Title
@jkelly • v.1 • code: private • data: private • last updated: now
Share
WORKFLOW
README
SETTINGS
Type to search live events
Today
1:35:40 p.m.
1:03:17 p.m.
1:02:46 p.m.
HTTP API

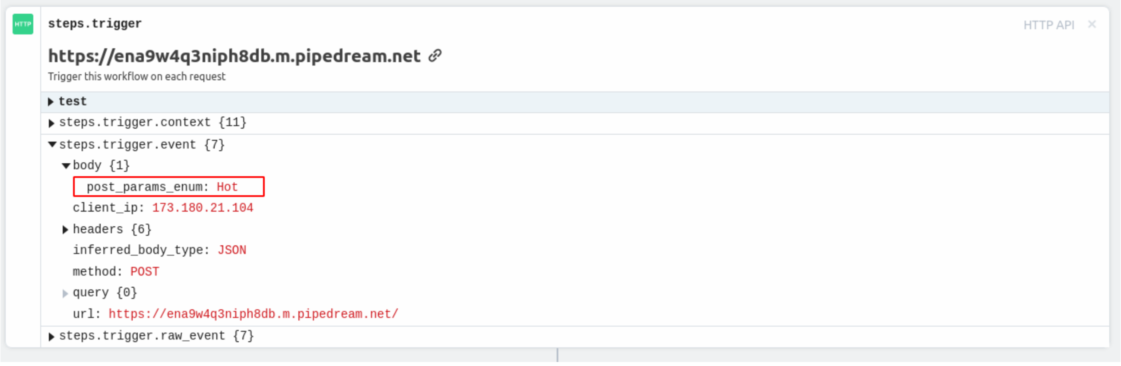
```
steps.trigger
https://ena9w4q3niph8db.m.pipedream.net
Trigger this workflow on each request
▸ test
▸ steps.trigger.context {11}
▾ steps.trigger.event {7}
  ▸ body {3}
    client_ip: 173.180.21.104
  ▸ headers {6}
    inferred_body_type: FORM
    method: GET
  ▸ query {1}
    url: https://ena9w4q3niph8db.m.pipedream.net/?params_get=222
▸ steps.trigger.raw_event {7}
```

GET request



```
steps.trigger
https://ena9w4q3niph8db.m.pipedream.net
Trigger this workflow on each request
▸ test
▸ steps.trigger.context {11}
▾ steps.trigger.event {7}
  ▾ body {1}
      post_params: 333
    client_ip: 173.180.21.104
  ▸ headers {6}
    inferred_body_type: JSON
    method: POST
  ▸ query {0}
    url: https://ena9w4q3niph8db.m.pipedream.net/
▸ steps.trigger.raw_event {7}
```

POST Template Data Broker



```
steps.trigger
https://ena9w4q3niph8db.m.pipedream.net
Trigger this workflow on each request
▸ test
▸ steps.trigger.context {11}
▾ steps.trigger.event {7}
  ▾ body {1}
      post_params_enum: Hot
    client_ip: 173.180.21.104
  ▸ headers {6}
    inferred_body_type: JSON
    method: POST
  ▸ query {0}
    url: https://ena9w4q3niph8db.m.pipedream.net/
▸ steps.trigger.raw_event {7}
```

POST Template Enumeration

8 Advanced Topics

8.1 Debugging the HTTP Trigger Driver

1. If the HTTP Trigger Driver is not pushing out values to the API, please verify that:

- The URL endpoint accepts the request of the type chosen.
- That the Content-Type matches the data, when the Method is set to POST or PUT.

8.2 How to Take a Diagnostic Log

Please see https://store.chipkin.com/articles/how-to-take-a-diagnostic-log-on-a-quickserverfieldserver for the most up-to-date information

9 Marketing

9.1 Case Study

{TO DO}

9.2 Keywords

HTTP, HTTP, REST, API

9.3 Glossary of Terms

1. **HTTP(s)** – Hypertext Transfer Protocol (secure)
2. **API** – Architecture Programming Interface
3. **REST** – Representational State Transfer
4. **URL** – Uniform Resource Locator
5. **PE** – Protocol Engine
6. **AE** – Application Engine

10 Revision History

This table summarizes the update history for this document. Please contact Chipkin for an updated version of this document if required.

DATE	RESP	DOC. REV.	COMMENT
20 Sep 2021	JJK	1	Created initial document
23 Sep 2021	JJK	2	Added examples using Pipedream, and images
29 Sep 2021	JJK	3	Removed license section
06 Oct 2021	YC	4	Updated to latest template