Managing State with Stateless Cloud Functions

January 24, 2018

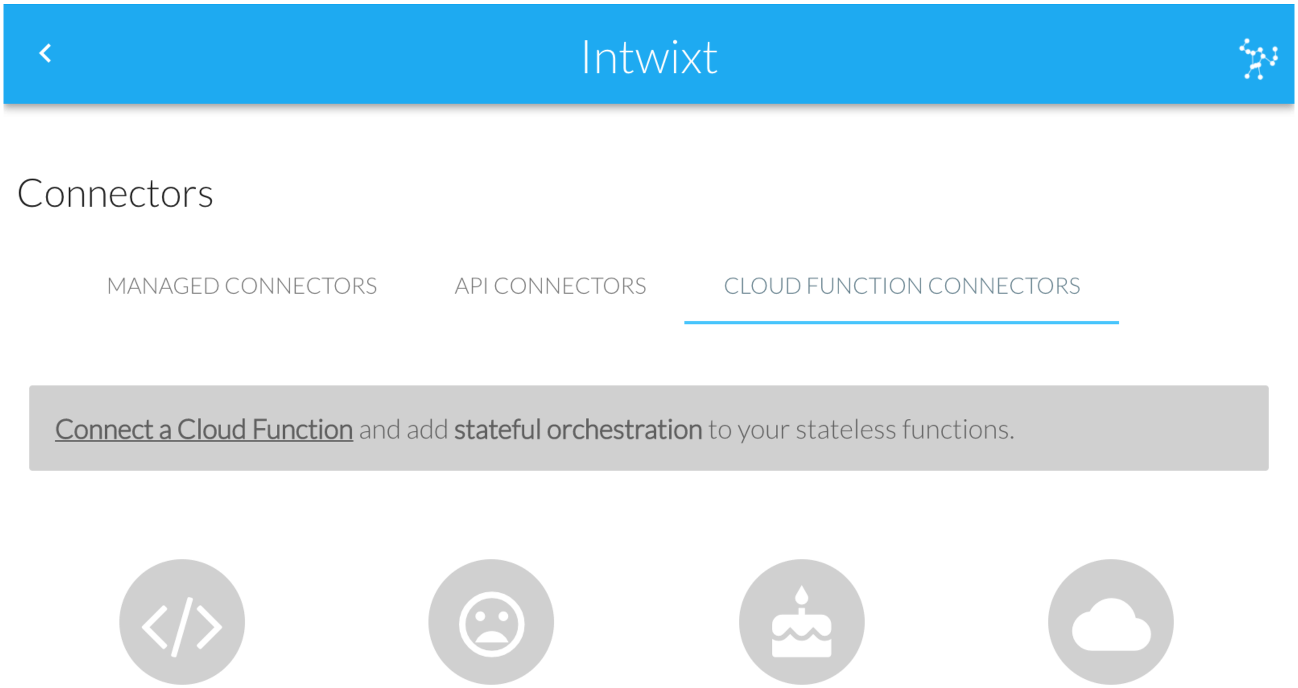
by Luke Birdeau

Function as a Service (FaaS) is a growing trend in cloud computing. It's an opinionated approach to helping users write cloud applications that scale. A central tenant of cloud functions (and what really makes them effective) is their stateless nature. They handle requests then disappear. Nothing is ever permanently "on" which is why they're so cost effective for the cloud provider to host and manage.

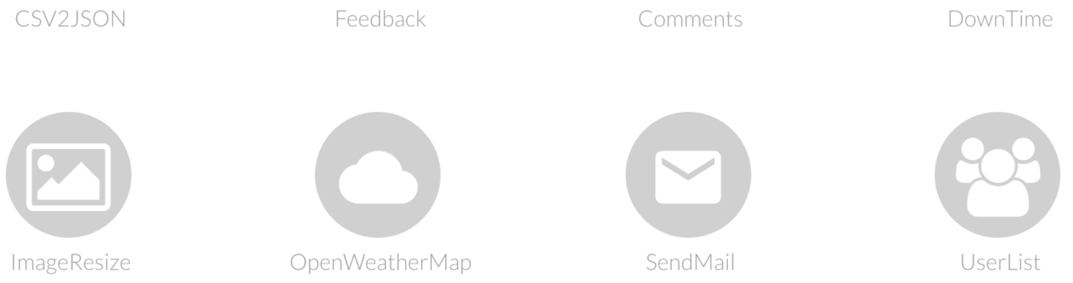
Of course, there are times when it does make sense to handle state. But this does not mean you should clutter your stateless function code with state-related functionality. This is better left to an orchestration engine or similar event-driven strategy that is **external** to your cloud function. This gives you a proper separation of concerns between what is stateless and what is not (which is what you're really after with serverless architectures). I'll use the remainder of this post to detail how we do this in Intwixt.

Step 1. Create A Cloud Connector

Launch the Intwixt Web app (https://my.intwixt.com). Click the global menu link in the top-right corner and choose the menu item, CONNECTORS. When the page loads, choose the tab, CLOUD FUNCTION CONNECTORS. The cloud functions you connect will be listed here.

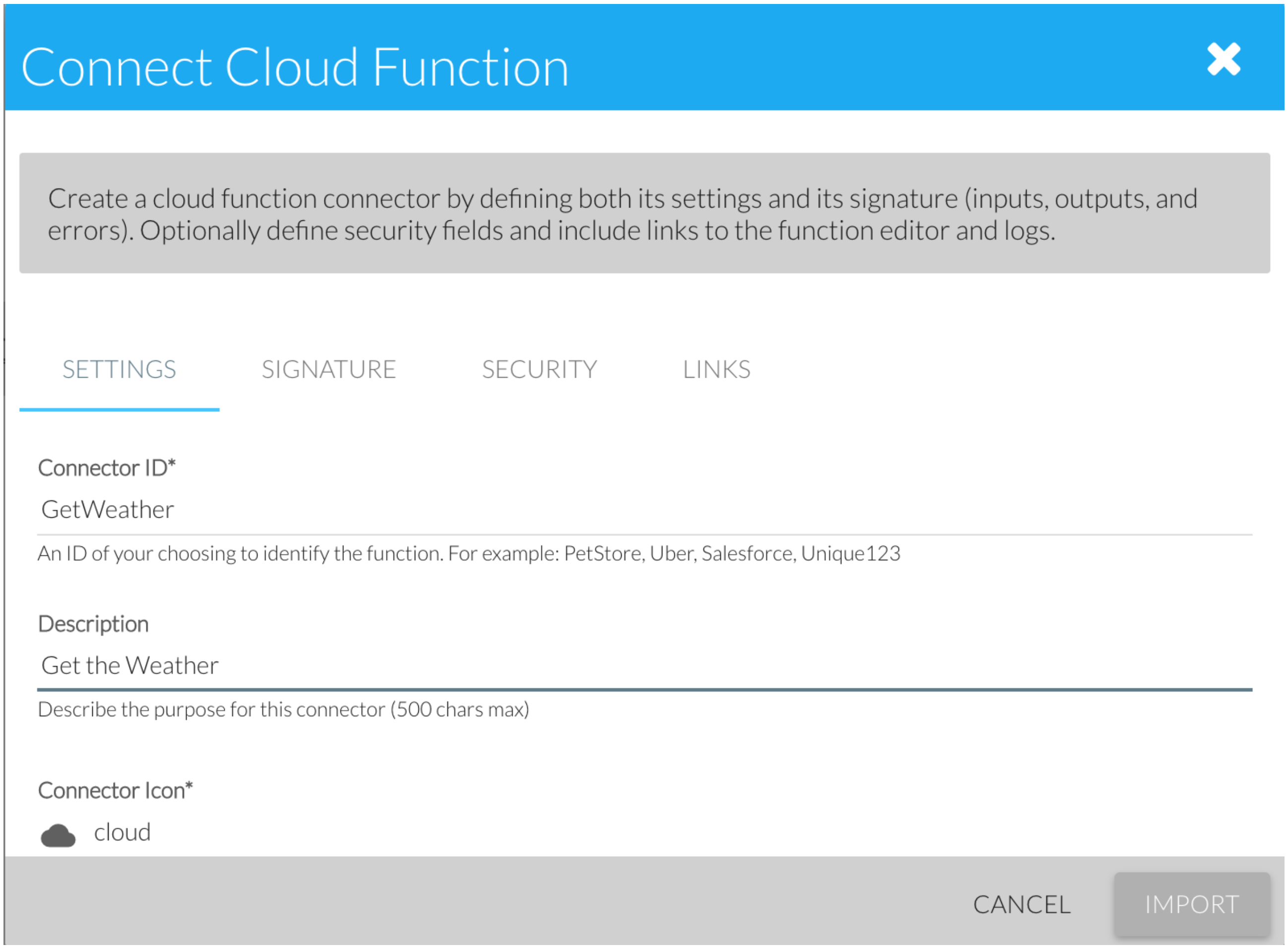


Connect a Cloud Function and add **stateful orchestration** to your stateless functions.



The connectors page, showing the user's private cloud function connectors.

Begin by clicking the link, **Connect a Cloud Function**, to open the **Connect Cloud Function** wizard.



Connect Cloud Function

Create a cloud function connector by defining both its settings and its signature (inputs, outputs, and errors). Optionally define security fields and include links to the function editor and logs.

Connector ID*
GetWeather
An ID of your choosing to identify the function. For example: PetStore, Uber, Salesforce, Unique123

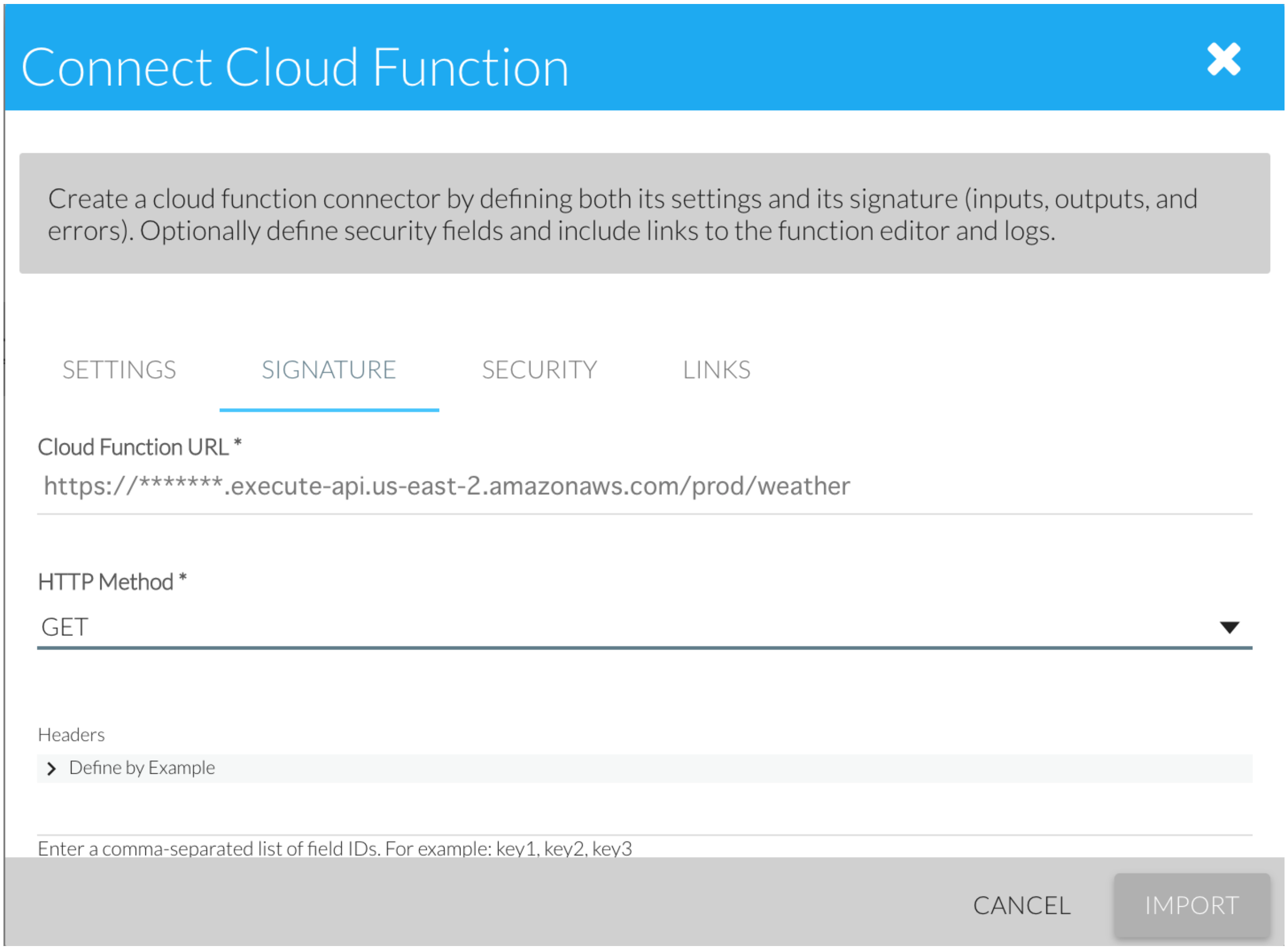
Description
Get the Weather
Describe the purpose for this connector (500 chars max)

Connector Icon*
cloud

Connect Cloud Function wizard showing SETTINGS panel

Enter a unique name to identify your cloud function within Intwixt. This will become the connector alias that Intwixt uses as you reference it in your flows, so choose something concise and descriptive. We also recommend you choose a relevant icon as it is used to represent your cloud function visually within a flow.

Once you are satisfied with the settings, activate the SIGNATURE tab. Enter a URL and choose the HTTP Method. This example uses an HTTP GET.


Connect Cloud Function

Create a cloud function connector by defining both its settings and its signature (inputs, outputs, and errors). Optionally define security fields and include links to the function editor and logs.

Cloud Function URL *
https://*******.execute-api.us-east-2.amazonaws.com/prod/weather

HTTP Method *
GET

Headers
Define by Example

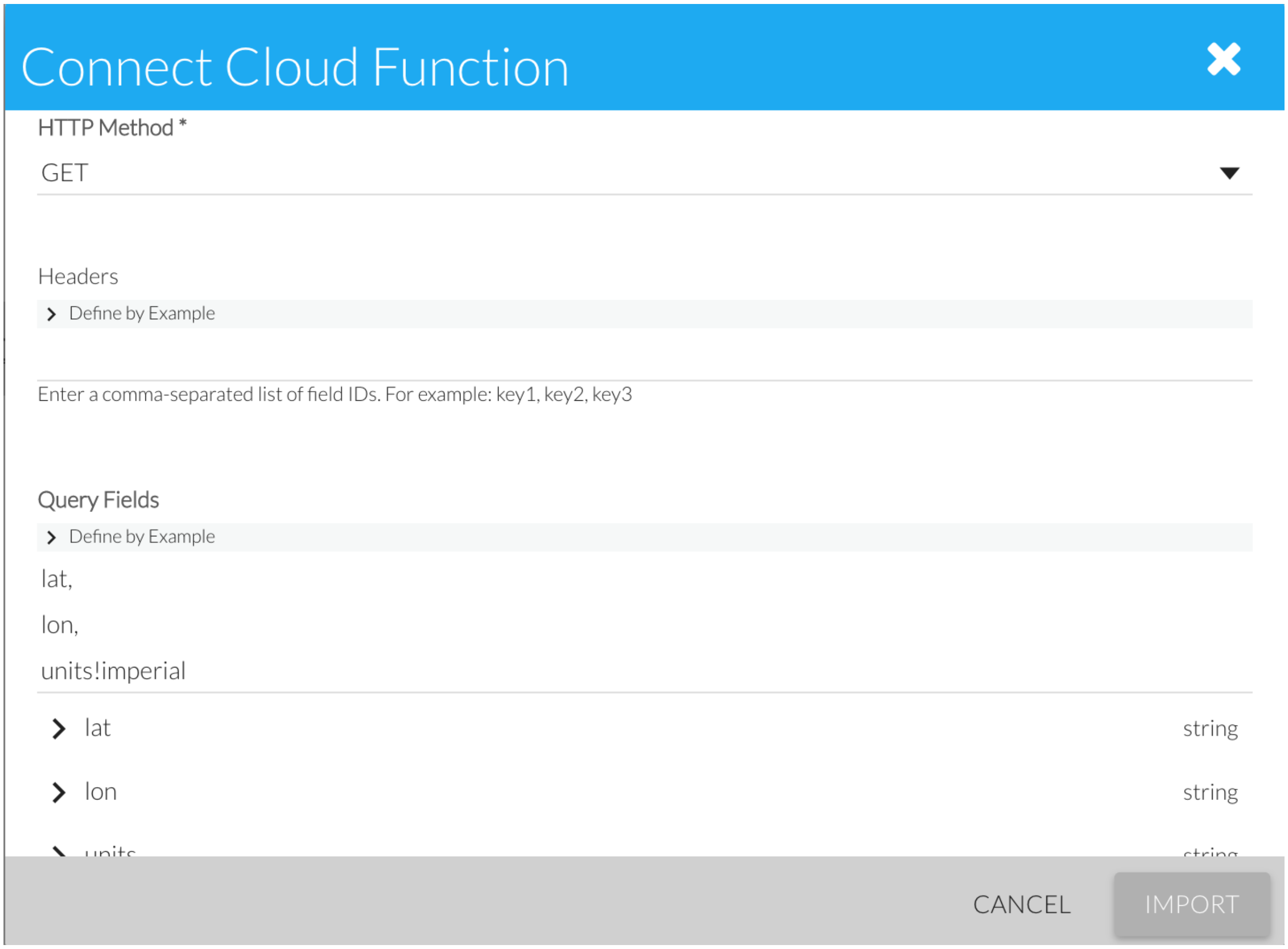
Enter a comma-separated list of field IDs. For example: key1, key2, key3

Configuring the function signature

The next step is to define any header or query parameters. The GetWeather function being connected requires three HTTP query parameters:

- **lat** | the latitude
- **lon** | the longitude
- **units** | The temperature units (imperial)

Defining a basic schema is relatively straight-forward and requires no more than a comma-separated list of field names. You can add more detail, but most schemas can be defined using nothing more than a comma. I'll also add a default value (imperial) by using an exclamation mark as a flag (**!imperial)**. Refer to the Intwixt Docs for a full list of escape characters.


Connect Cloud Function

HTTP Method *

GET

Headers

Define by Example

Enter a comma-separated list of field IDs. For example: key1, key2, key3

Query Fields

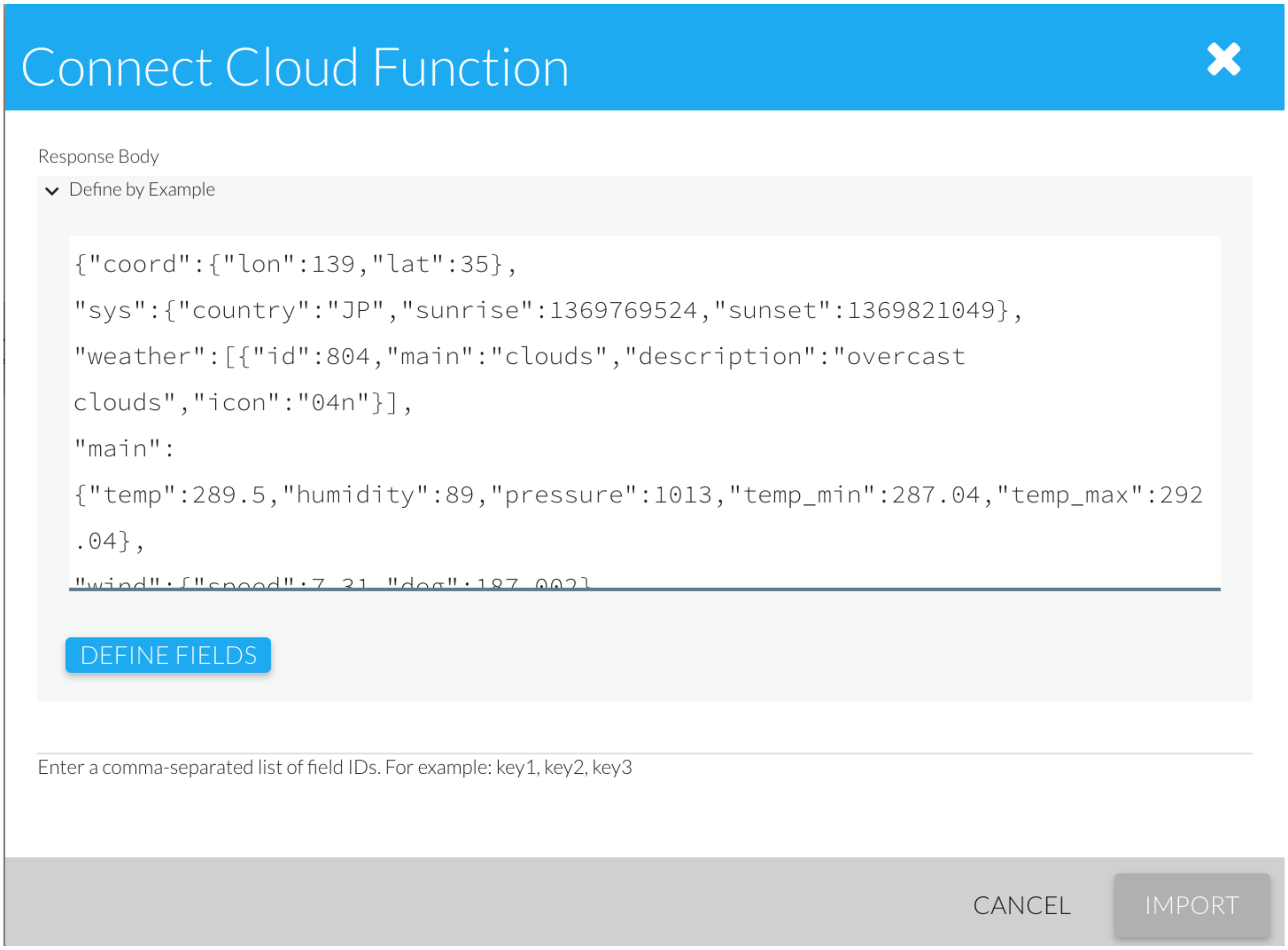
Define by Example

lat,
lon,
units!imperial

lat	string
lon	string
units	string

Define the query fields using a sample JSON message

The final step for the signature definition is to define the body (what will be returned from the cloud function when called at runtime). Because the JSON output is complex, it's easier to define the model by using a sample JSON document. Paste the sample JSON and then click DEFINE FIELDS.



Response Body

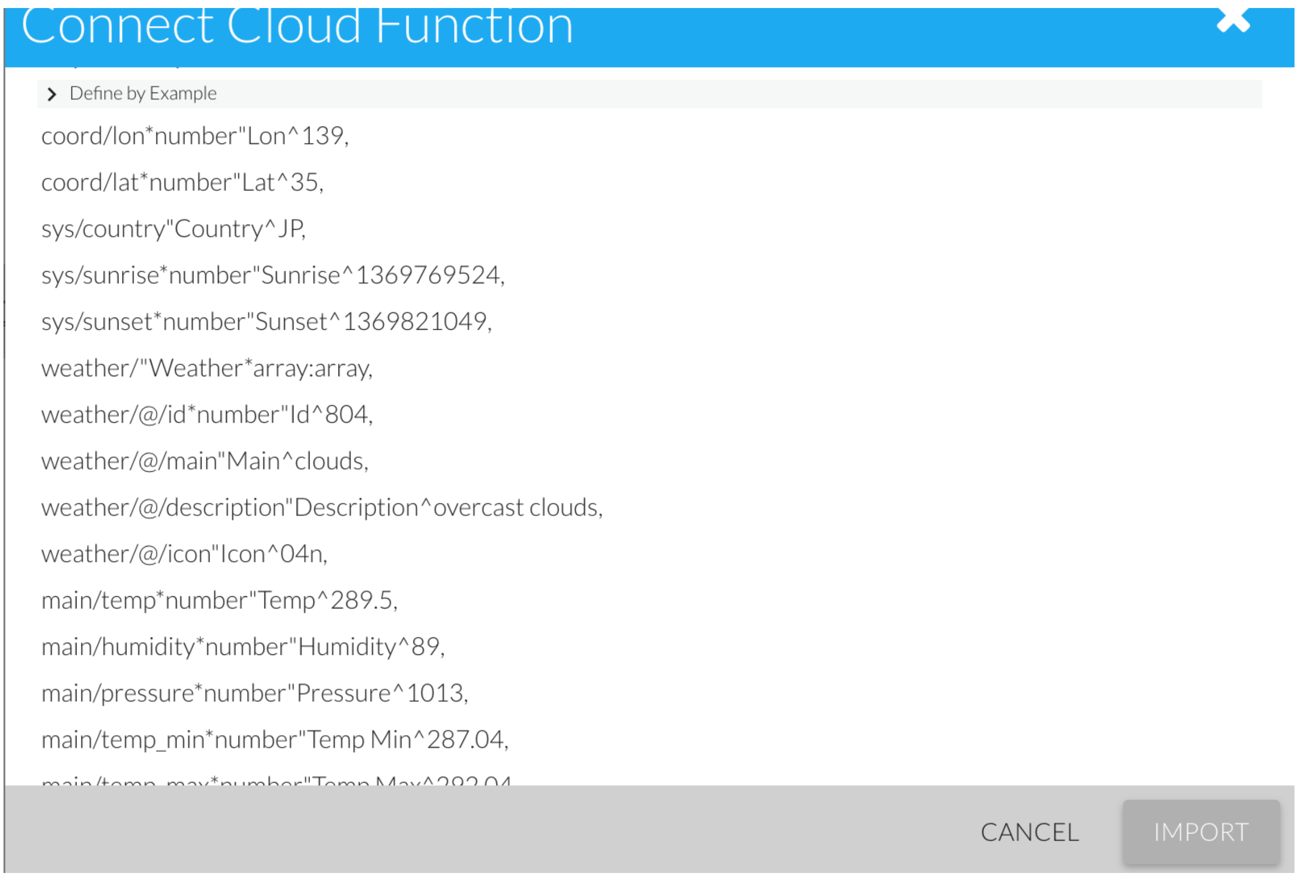
Define by Example

```
{"coord":{"lon":139,"lat":35},
"sys":{"country":"JP","sunrise":1369769524,"sunset":1369821049},
"weather":[{"id":804,"main":"clouds","description":"overcast
clouds","icon":"04n"}],
"main":
{"temp":289.5,"humidity":89,"pressure":1013,"temp_min":287.04,"temp_max":292
.04},
"wind":{"speed":7.31,"deg":187.002},
```

Enter a comma-separated list of field IDs. For example: key1, key2, key3

Define the schema for the response body by using a sample JSON document

The wizard will parse the JSON into the necessary model definition. You can make further edits to the generated model as necessary to fully customize it to fit your needs.



coord/lon*number"Lon^139,

coord/lat*number"Lat^35,

sys/country"Country^JP,

sys/sunrise*number"Sunrise^1369769524,

sys/sunset*number"Sunset^1369821049,

weather/"Weather*array:array,

weather/@/id*number"Id^804,

weather/@/main"Main^clouds,

weather/@/description"Description^overcast clouds,

weather/@/icon"Icon^04n,

main/temp*number"Temp^289.5,

main/humidity*number"Humidity^89,

main/pressure*number"Pressure^1013,

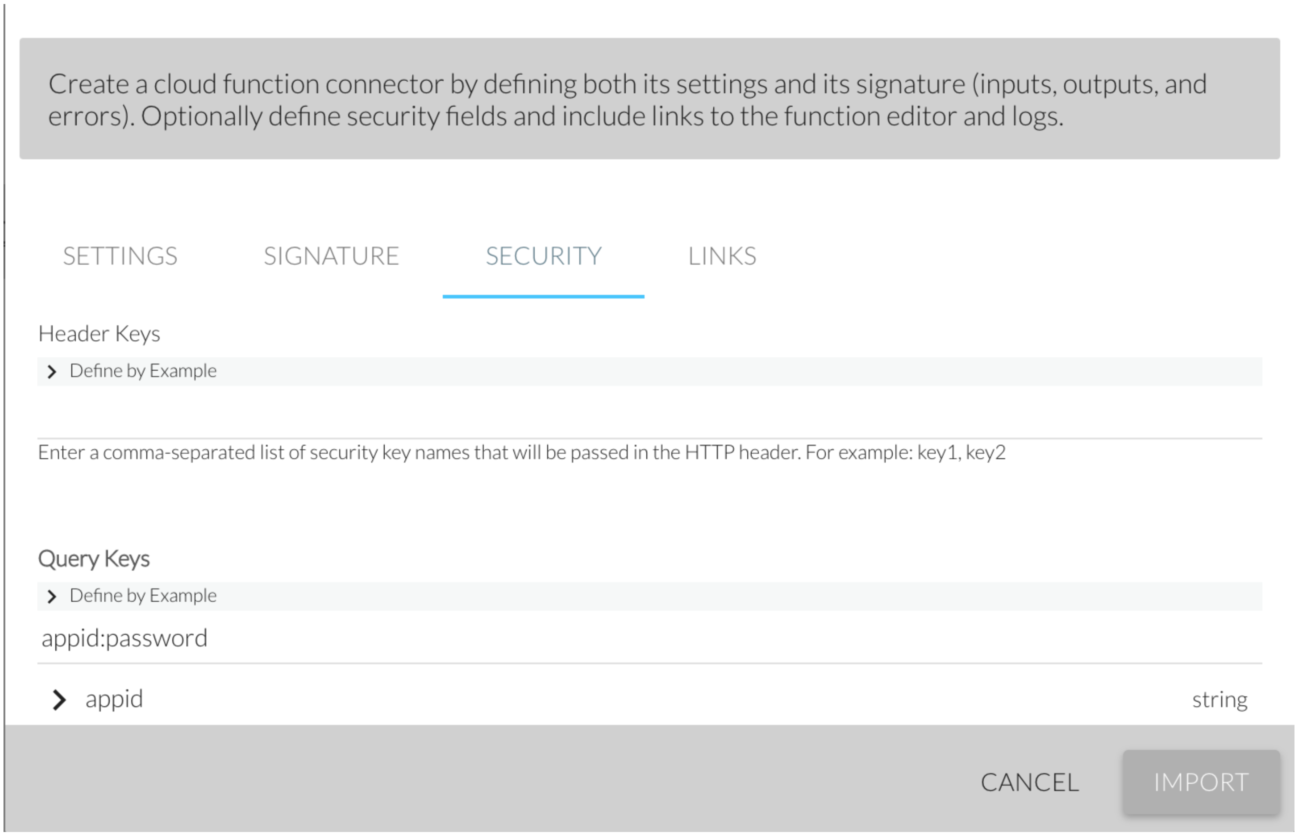
main/temp_min*number"Temp Min^287.04,

main/temp_max*number"Temp Max^292.04,

The schema as generated using a sample JSON document

With the signature now defined, the next step is to define security fields (if required by the cloud function). Security field values are managed separately from connectors. This ensures your credentials are properly externalized and encrypted by Intwixt and available for use only when your flow is invoked at runtime.

The GetWeather cloud function being used in this example has one security field named **appid**. It must be passed as an HTTP query parameter. I will add a flag to this field to designate it as a password field (**:password**). This makes sure that the field values cannot be seen when entered by the user (an HTML password input type)



Create a cloud function connector by defining both its settings and its signature (inputs, outputs, and errors). Optionally define security fields and include links to the function editor and logs.

Header Keys

Enter a comma-separated list of security key names that will be passed in the HTTP header. For example: key1, key2

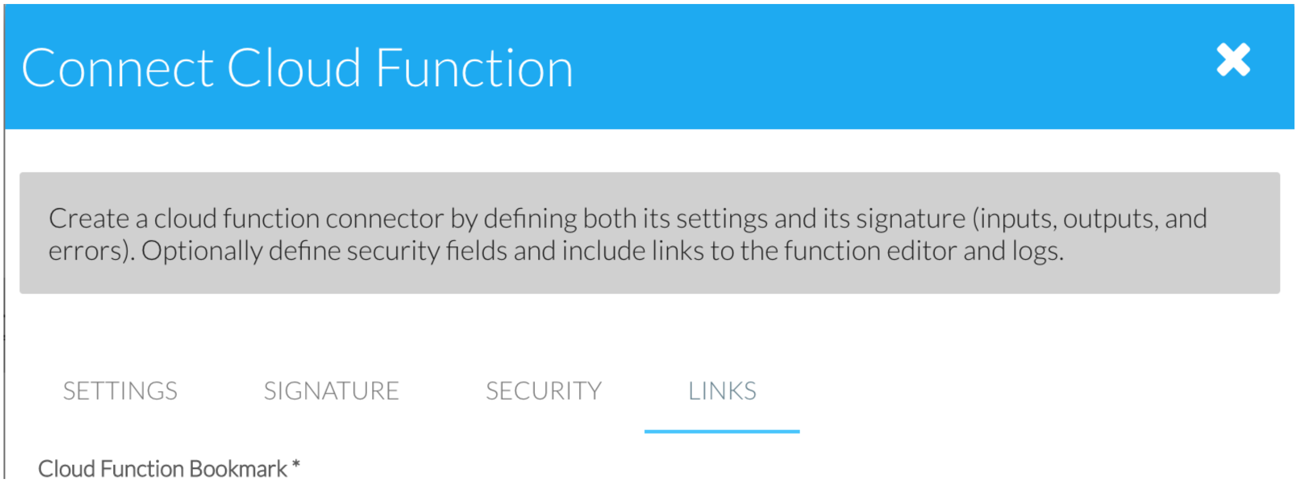
Query Keys

appid:password

appid string

Define security fields.

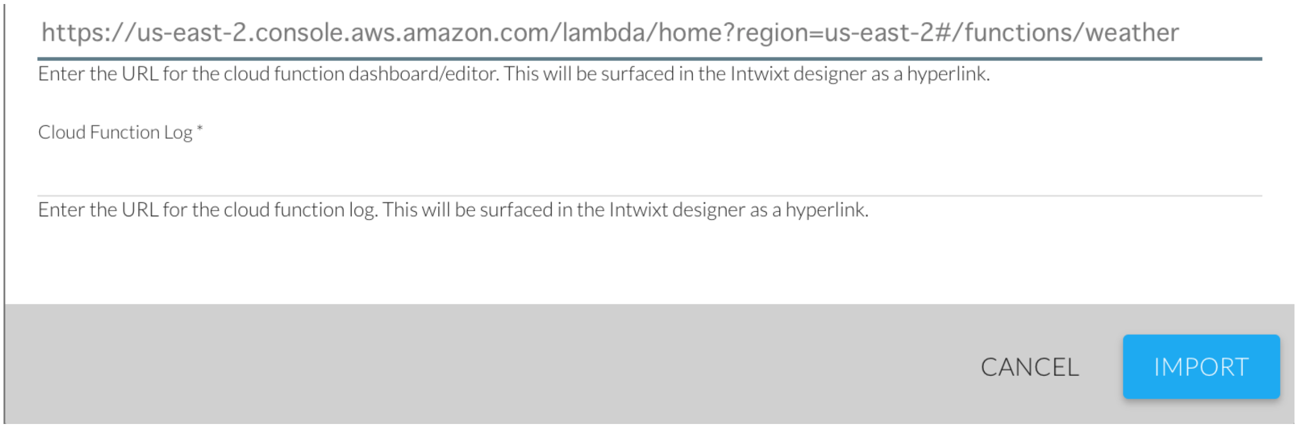
The final step is to include a link to the cloud function source page. This will be surfaced in the Intwixt designer, allowing me to link to the cloud function source for easier debugging and testing. Use the LINKS tab to add this optional value as shown here.



Connect Cloud Function

Create a cloud function connector by defining both its settings and its signature (inputs, outputs, and errors). Optionally define security fields and include links to the function editor and logs.

Cloud Function Bookmark *



https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions/weather

Enter the URL for the cloud function dashboard/editor. This will be surfaced in the Intwixt designer as a hyperlink.

Cloud Function Log *

Enter the URL for the cloud function log. This will be surfaced in the Intwixt designer as a hyperlink.

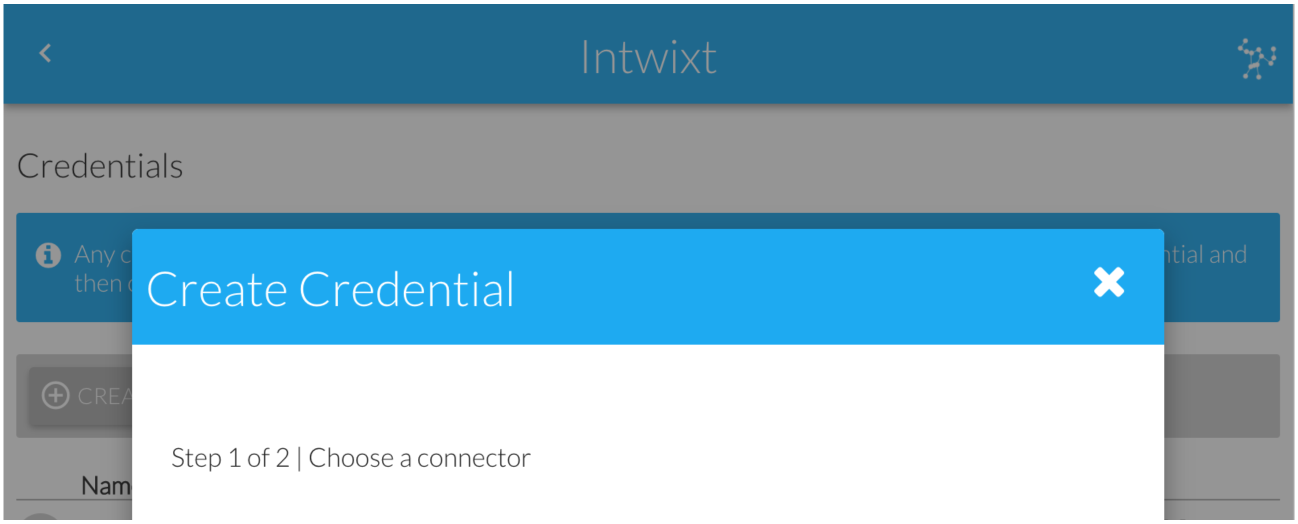
Link to the cloud function dashboard.

Click the IMPORT button to finalize your choices.

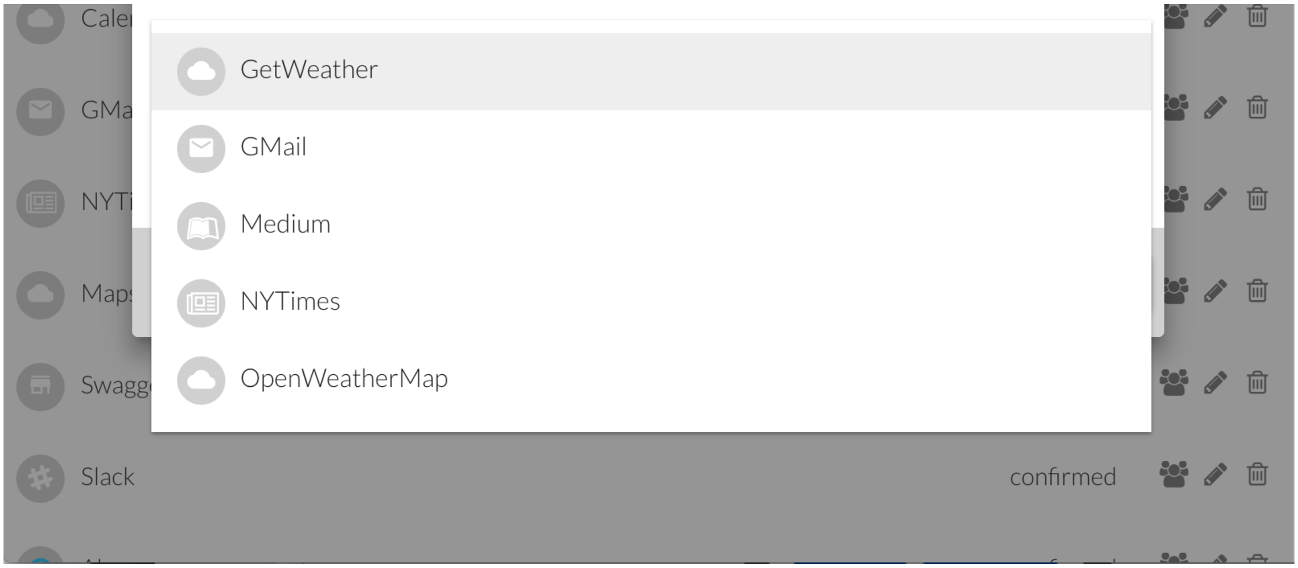
Step 2: Create A Credential

The GetWeather cloud function requires a security credential which I'll create now.

Click the global menu link in the top-right corner and choose the menu item, CREDENTIALS. When the credentials page loads, click the button, **CREATE NEW** to load the **Create Credential** Dialog. Choose the connector you just created from the list of options (e.g., GetWeather).



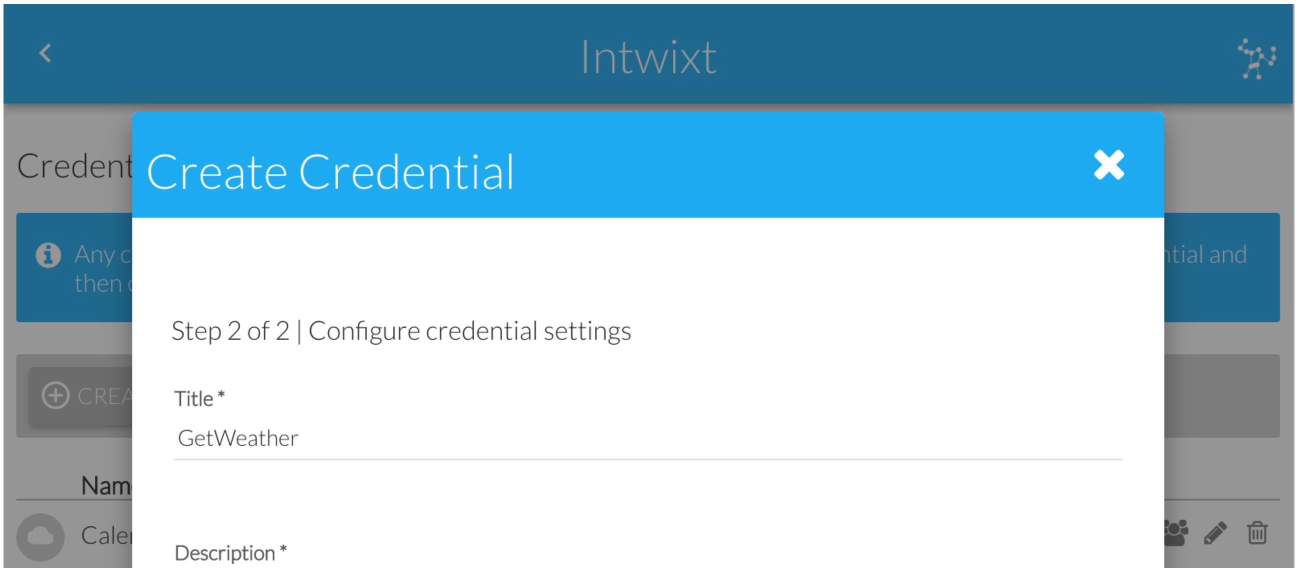
Step 1 of 2 | Choose a connector



Create Credential Dialog | Choose Connector

Once the connector is chosen, the credential settings will be shown. Complete every field. The one custom field, **appid**, is the field that was defined just a moment ago when the connector was created. I need to include the security key required by the cloud function as shown here. (Notice how **Appid** is a *password* field. This is due to the configuration choice made when the connector was created.)

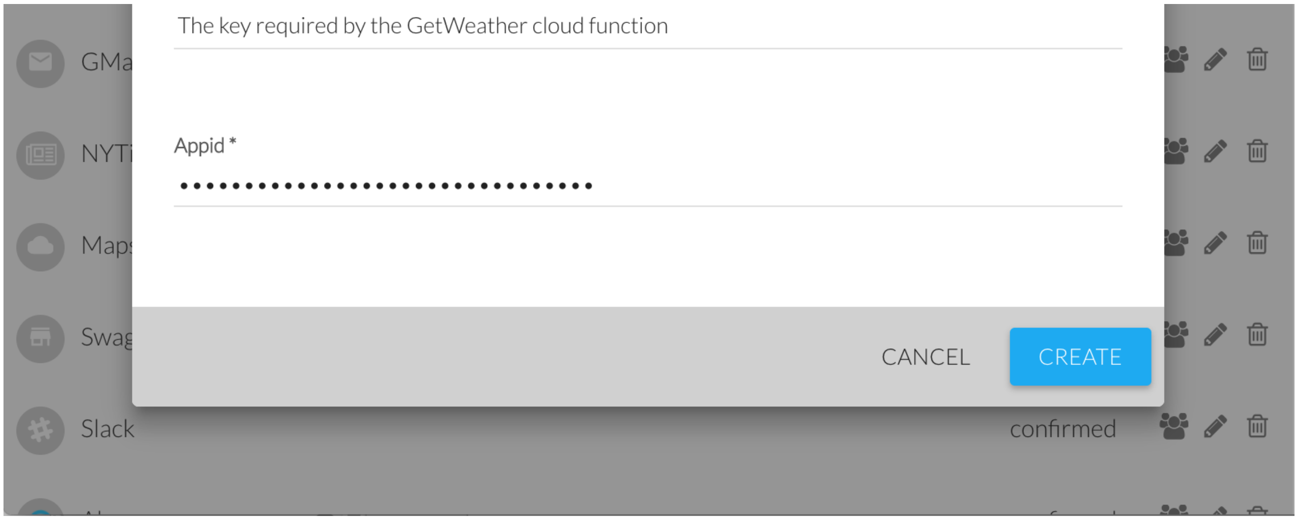
When done, click **CREATE** to complete the session and dismiss the dialog.



Step 2 of 2 | Configure credential settings

Title *
GetWeather

Description *



The key required by the GetWeather cloud function

Appid *

••••••••••••••••••••••••••••••••

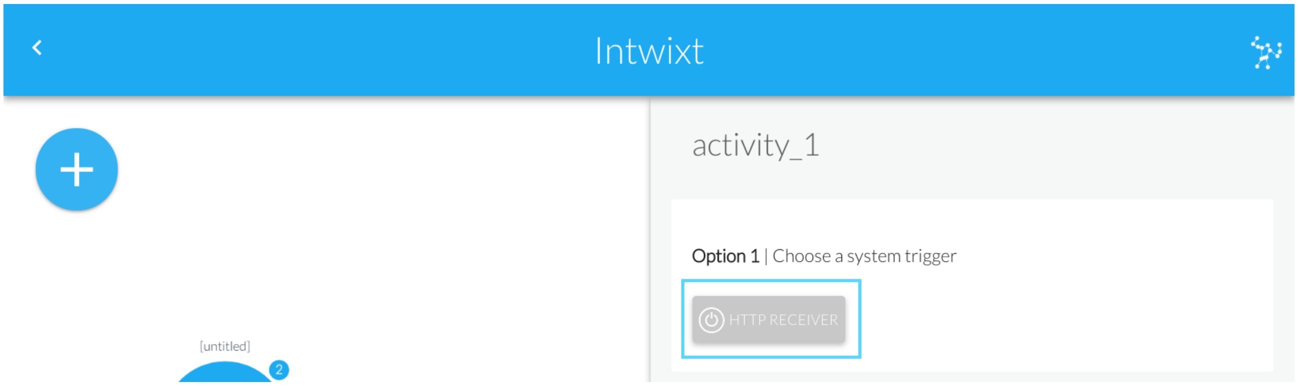
Create Credential Dialog | Configure Settings

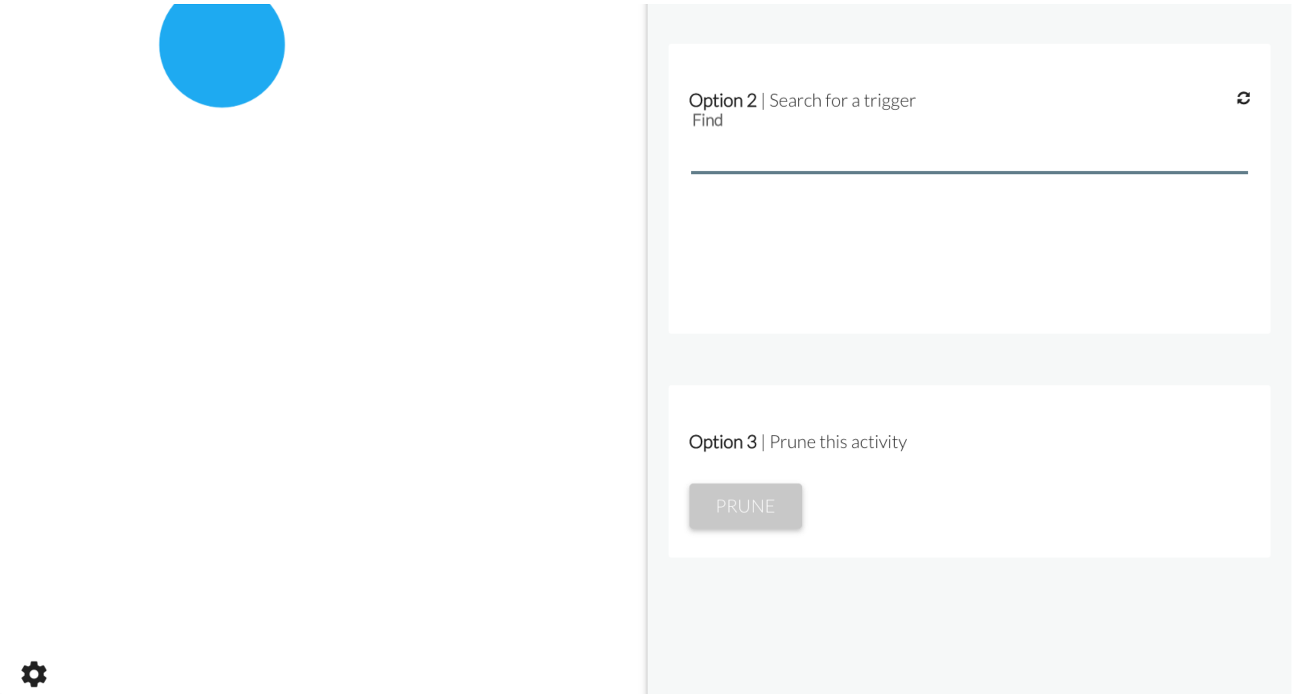
Step 3: Design A Flow

The last step is to design a flow that will invoke the cloud connector.

Click the global menu link in the top-right corner and choose the menu item, FLOWS. When the flows page loads, click the button, **CREATE > GENERIC FLOW** to load the **Create Generic Flow** Dialog. Enter a title and then click **CREATE**. A new blank flow will appear in the designer.

Click the **"+"** icon in the upper left to add a trigger to the canvas. Click the **HTTP RECEIVER** button to allow this trigger to be invoked over HTTP. This will make it easy to test the flow directly from the designer.





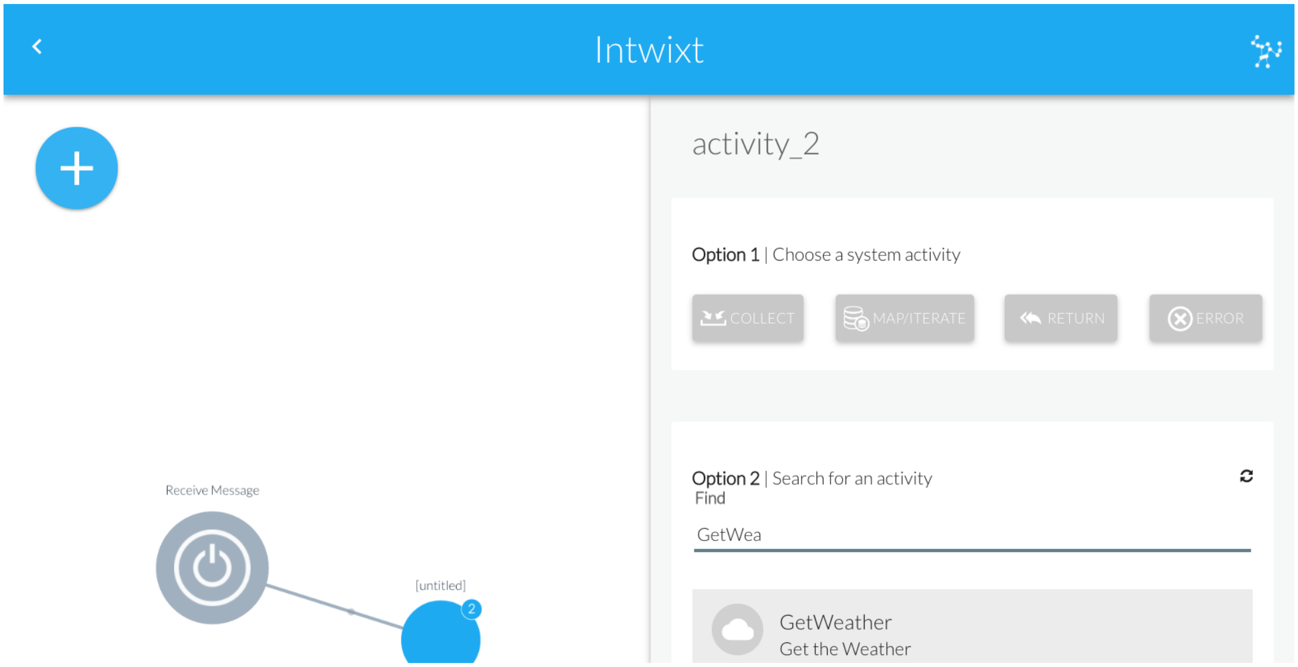
Option 2 | Search for a trigger

Find

Option 3 | Prune this activity

Intwixt process designer showing the type selection panel for a trigger.

Once the trigger type has been set, drag the **"+"** icon from the upper-left until it connects to the Receive Message tirgger. When the type selection panel loads on the right, search for the new connector (e.g., **GetWeather**).



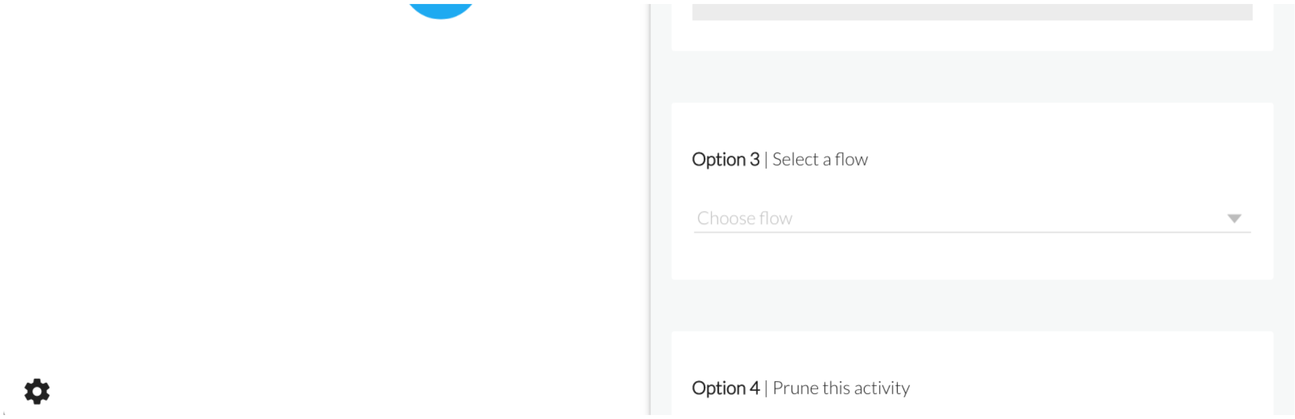
activity_2

Option 1 | Choose a system activity

Option 2 | Search for an activity

Find: GetWea

GetWeather
Get the Weather



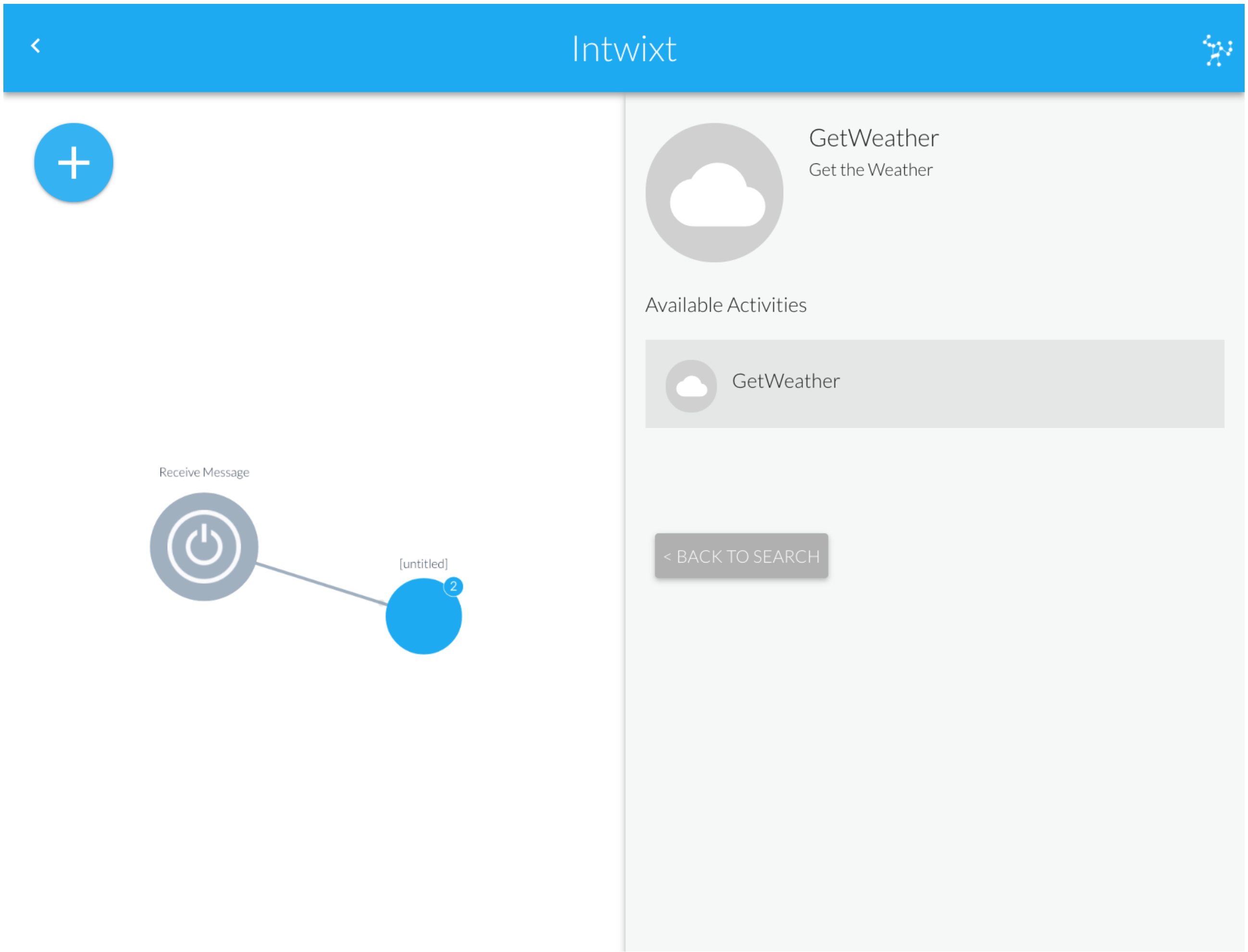
Option 3 | Select a flow

Choose flow

Option 4 | Prune this activity

Intwixt Designer | Activity Type Selection Panel | Search for Connector

Click the GetWeather result item (shown above) to reveal the activity of the same name. Click it once more (shown below) to complete your selection.



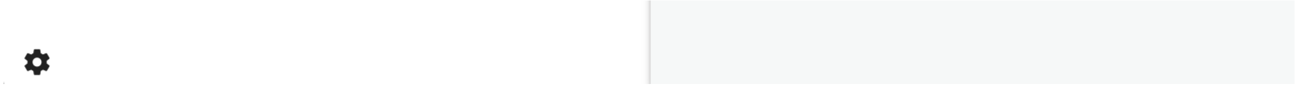
GetWeather

Get the Weather

Available Activities

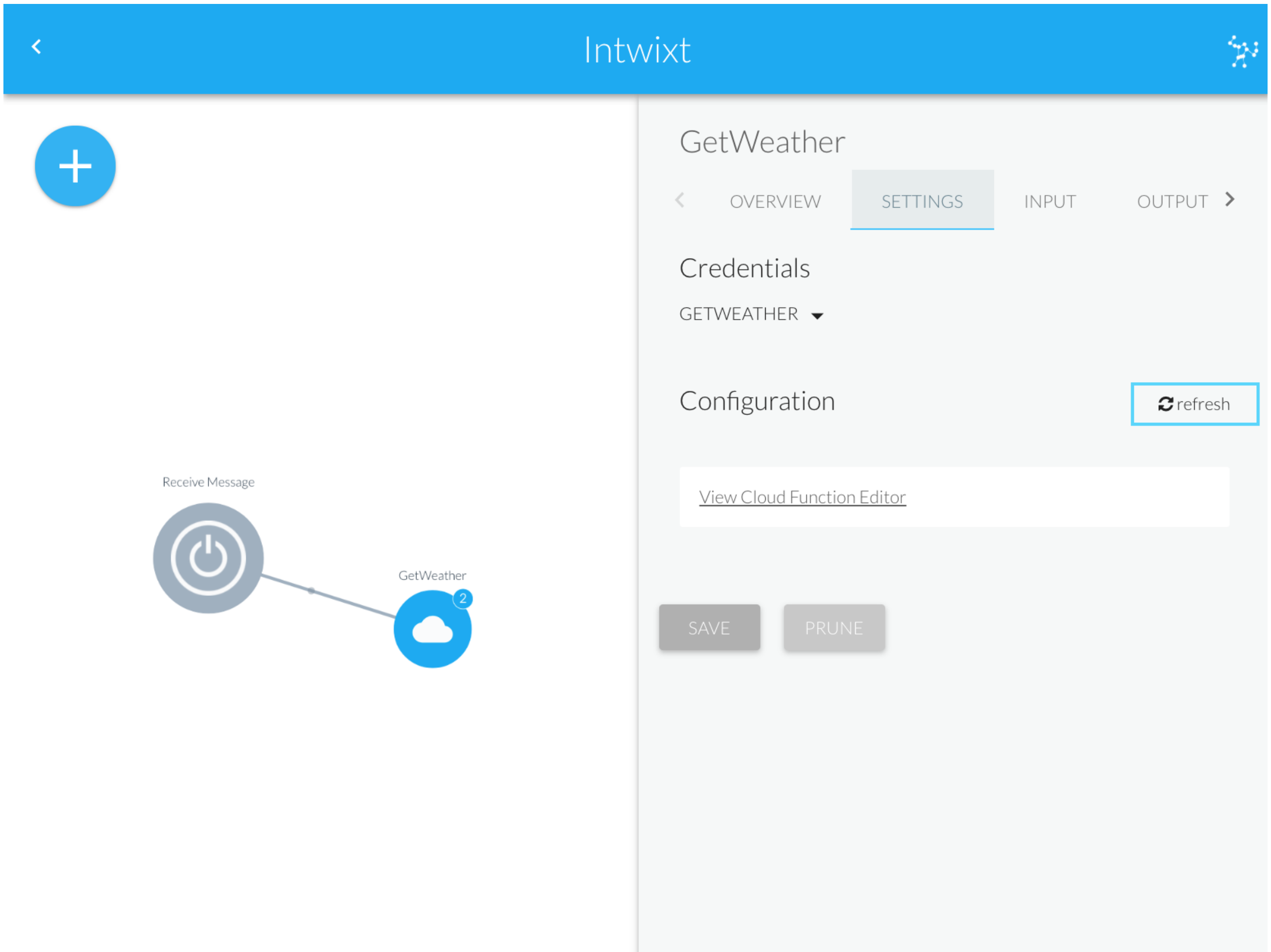
GetWeather

< BACK TO SEARCH



Intwixt Designer | Activity Type Selection Panel | Choose Connector

The activity icon, title, and settings will now be applied. Click the SETTINGS tab on the right side to reveal the Credentials. Take note that the credentials that were created in the prior step have been automatically selected. Note number **2** in the flow. This denotes that this newly added activity has 2 missing values. If you do not see this value, click the **refresh** icon.

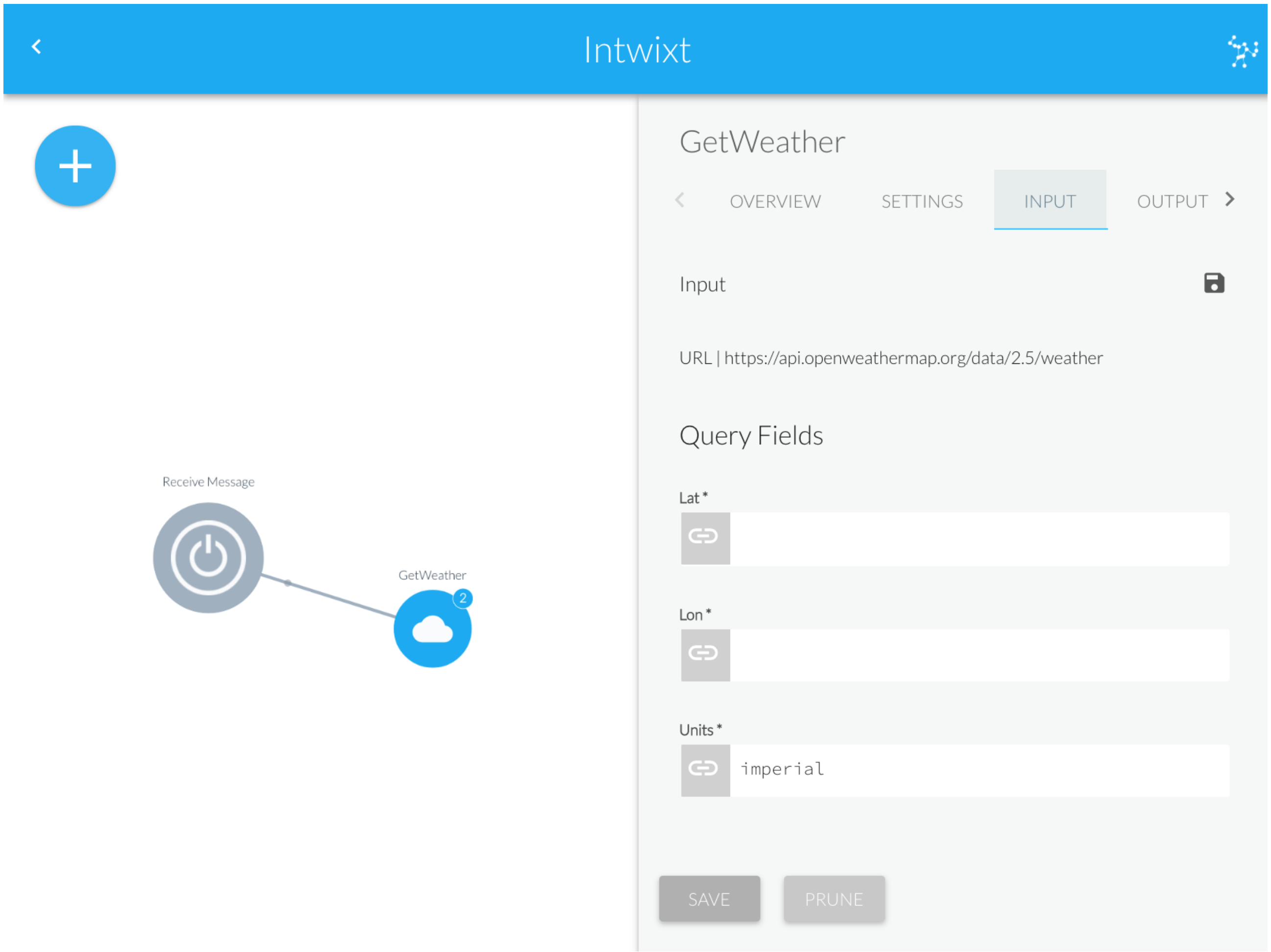


Credentials

GETWEATHER

Activity Configuration Panel | SETTINGS Tab

The number, **2**, is shown because the activity is missing two required inputs: **lat** and **lon**. The **units** field already contains the default value that was set when the connector was created (imperial). Click on the INPUT tab in the UI to reveal these two empty fields.



Input

URL | https://api.openweathermap.org/data/2.5/weather

Query Fields

Lat *

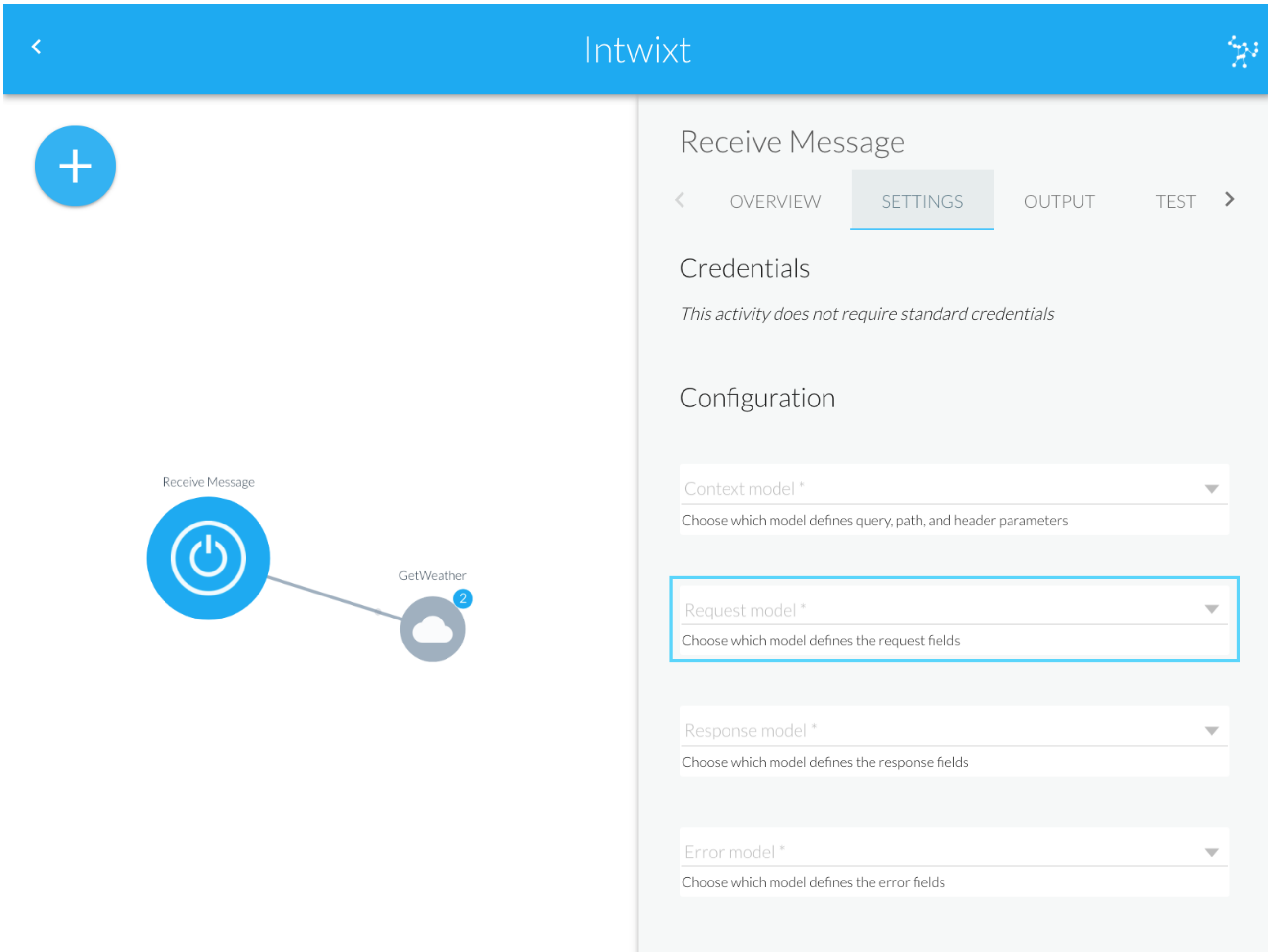
Lon *

Units * imperial

Activity Configuration Panel | INPUT Tab

I could enter static values, but it's better to pass latitude and longitude to the HTTP Receiver trigger and run the test repeatedly with different values. I need to use the mapper.

Select the Receive Message trigger in the canvas and click the SETTINGS tab.



Receive Message

Credentials

This activity does not require standard credentials

Configuration

Context model *
Choose which model defines query, path, and header parameters

Request model *
Choose which model defines the request fields

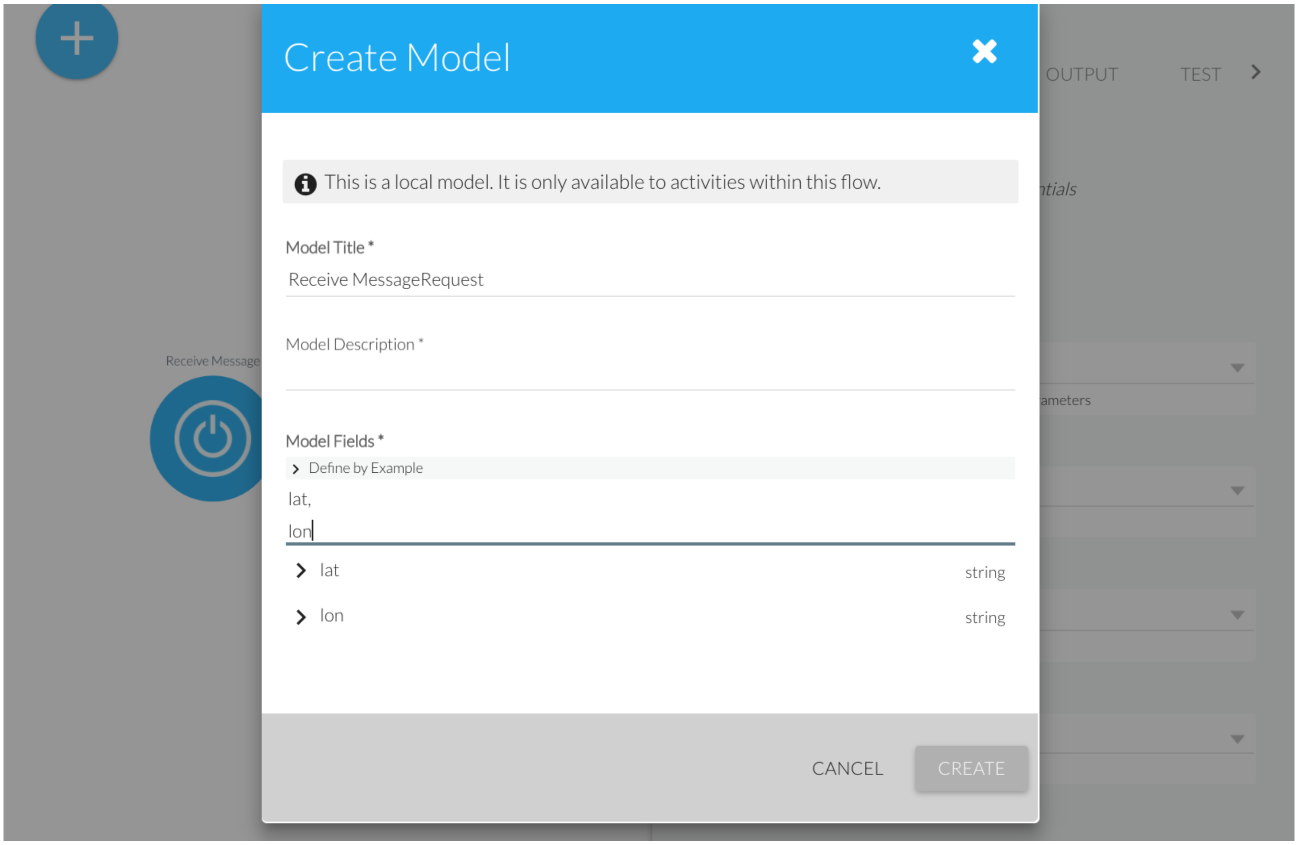
Response model *
Choose which model defines the response fields

Error model *
Choose which model defines the error fields

Trigger Configuration Panel | SETTINGS Tab

Click on the **Request Model** dropdown field (highlighted above). This will expand a list of options. Choose **Create new local model** to open the **Create Model** Dialog.

When it loads, provide a comma-separated list of field names (**lat**, **lon**) and then click the CREATE button.



Create Model

This is a local model. It is only available to activities within this flow.

Model Title *
Receive MessageRequest

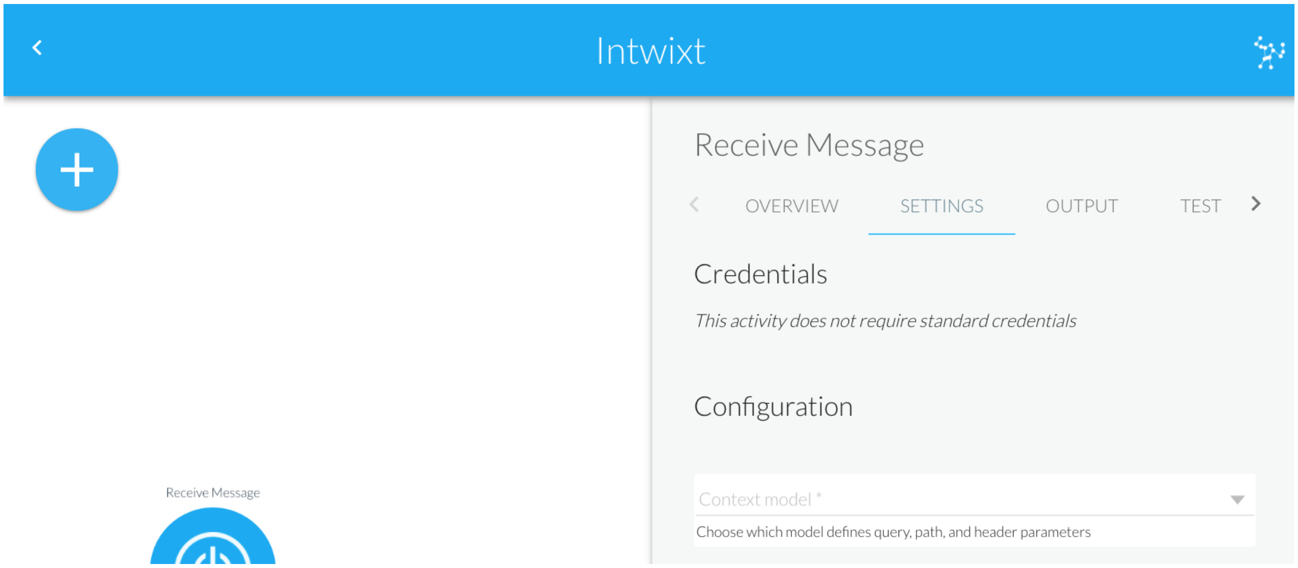
Model Description *

Model Fields *
Define by Example
lat,
lon

lat	string
lon	string

Create Model Dialog | Create local model

The new model will now appear in the right panel. It shows the fields, **lat** and **lon** (which are now required when a request is made).



Intwixt

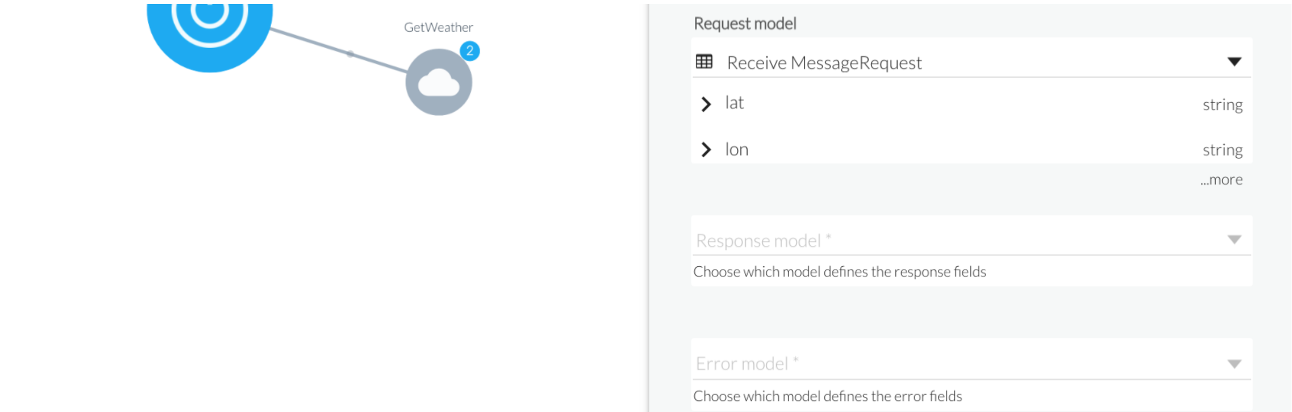
Receive Message

Credentials

This activity does not require standard credentials

Configuration

Context model *
Choose which model defines query, path, and header parameters



Request model

Receive MessageRequest

lat	string
lon	string

...more

Response model *

Choose which model defines the response fields

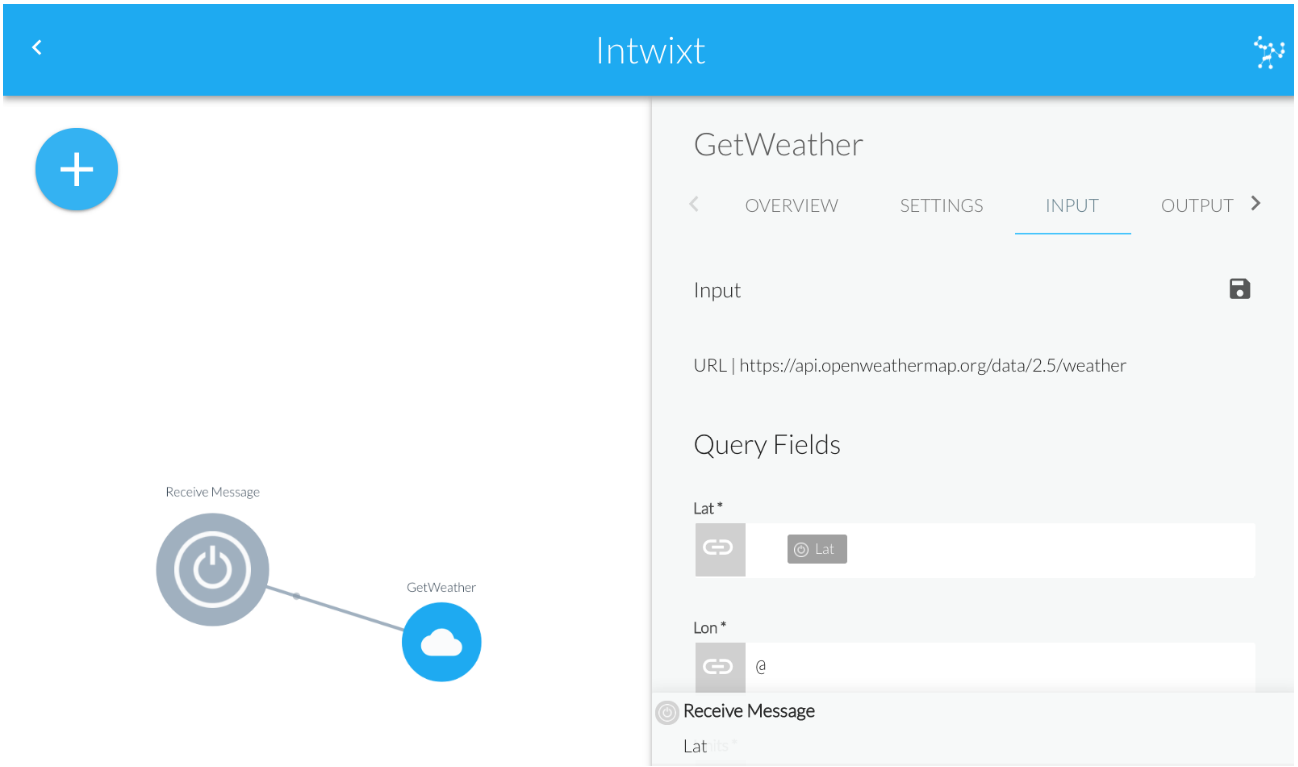
Error model *

Choose which model defines the error fields

Trigger Configuration Panel | SETTINGS Tab

Now that the HTTP Receiver defines these fields, downstream activities can map to their values.

I begin by clicking the target activity in the canvas (GetWeather) and selecting the INPUT tab. I expose the mapper and upstream data by typing the **"@"** symbol. This triggers a list of all possible mapping inputs. Choose **Lat** as the mapping value for the Lat field and **Lon** for the Lon field.



GetWeather

Input

URL | https://api.openweathermap.org/data/2.5/weather

Query Fields

Lat *

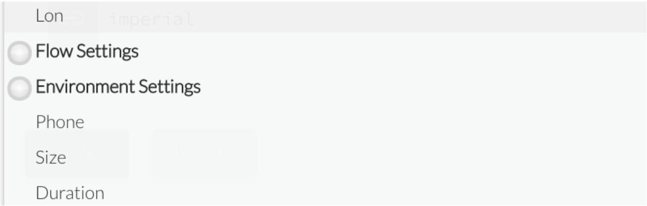
Lat

Lon *

@

Receive Message

Lat



Lon imperial

○ Flow Settings

○ Environment Settings

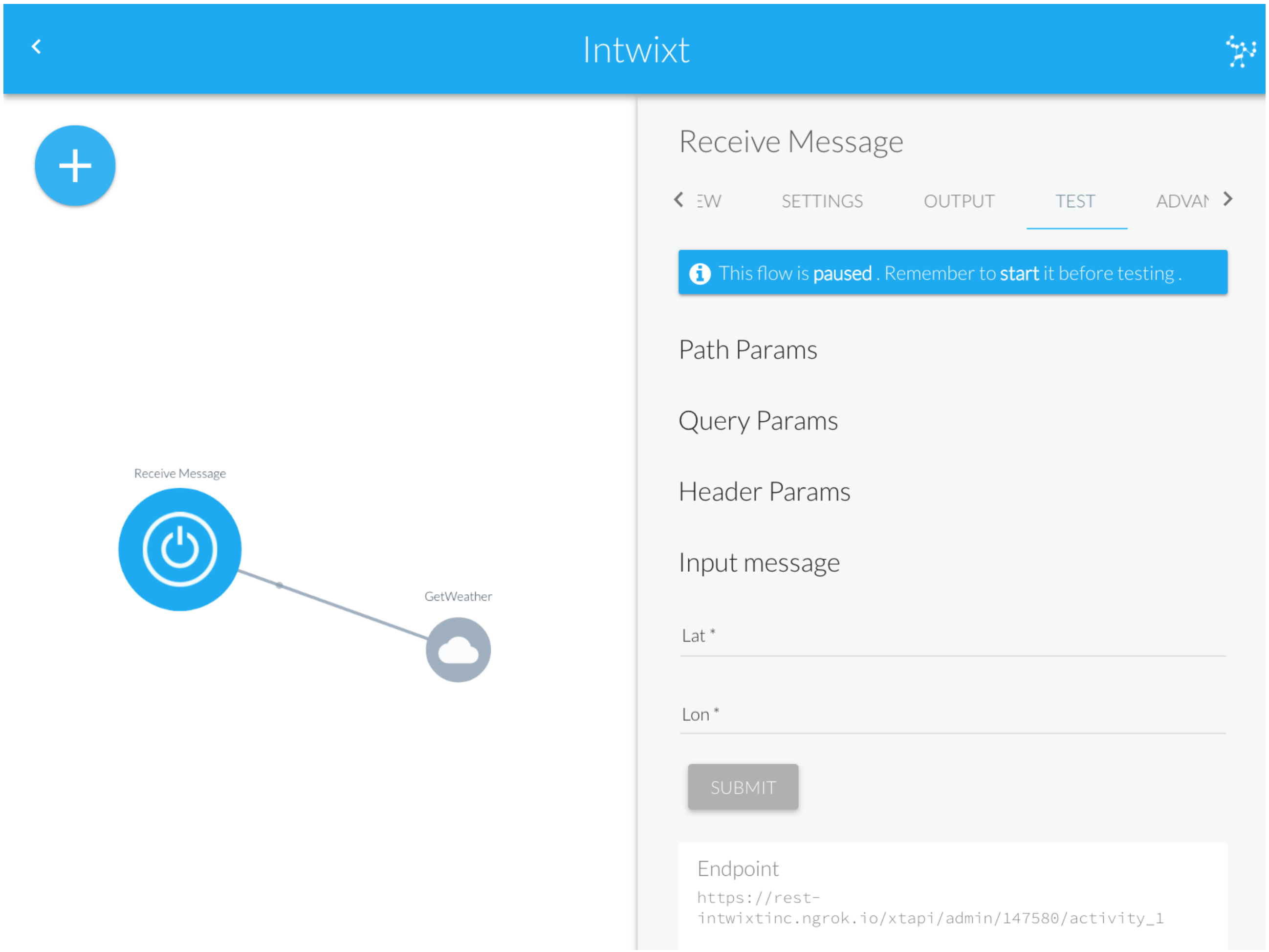
Phone

Size

Duration

Intwixt Mapper showing available upstream fields, including environment settings.

All required configuration is now set (and the **2** to-dos that were once in the designer are now gone). Run a test by selecting the Receive Message trigger in the designer canvas and then activating the TEST tab. Activate the flow by clicking the **start** link ("Remember to **start** it before testing"). A message will appear, letting you know when the flow is fully activated (usually in under a second).



Receive Message

EW SETTINGS OUTPUT TEST ADVAN

This flow is **paused** . Remember to **start** it before testing .

Path Params

Query Params

Header Params

Input message

Lat *

Lon *

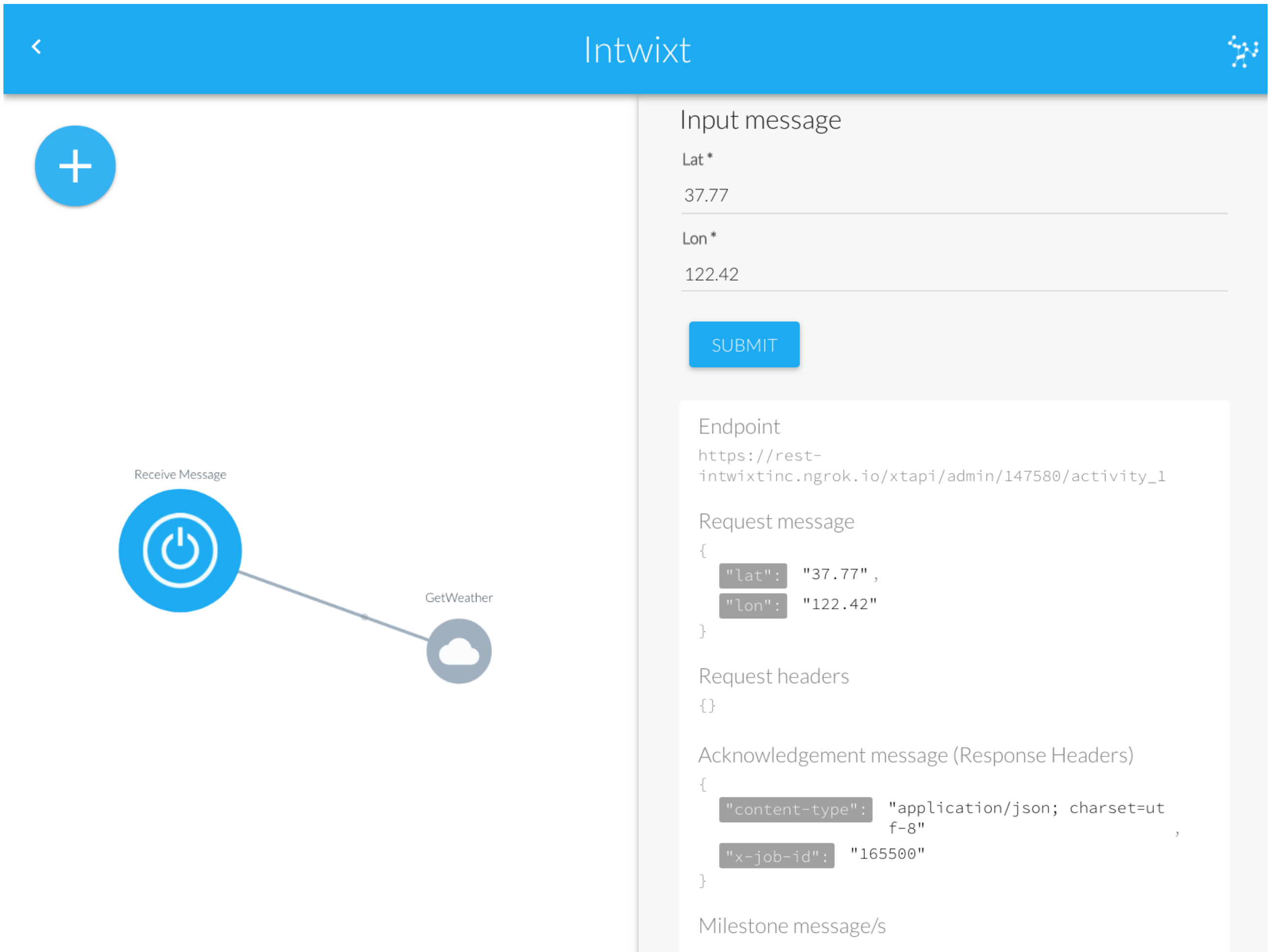
SUBMIT

Endpoint

```
https://rest-intwixtinc.ngrok.io/xtapi/admin/147580/activity_1
```

Trigger Configuration Panel | TEST Tab

Enter test values for latitude and longitude and then click SUBMIT to run the test. (The flow does not yet contain a **Return** activity, so there will be no HTTP output except for an acknowledgment message with the job ID.)



Input message

Lat *
37.77

Lon *
122.42

Endpoint
```
https://rest-intwixtinc.ngrok.io/xtapi/admin/147580/activity_1
```

Request message
```
{
  "lat":  "37.77",
  "lon":  "122.42"
}
```

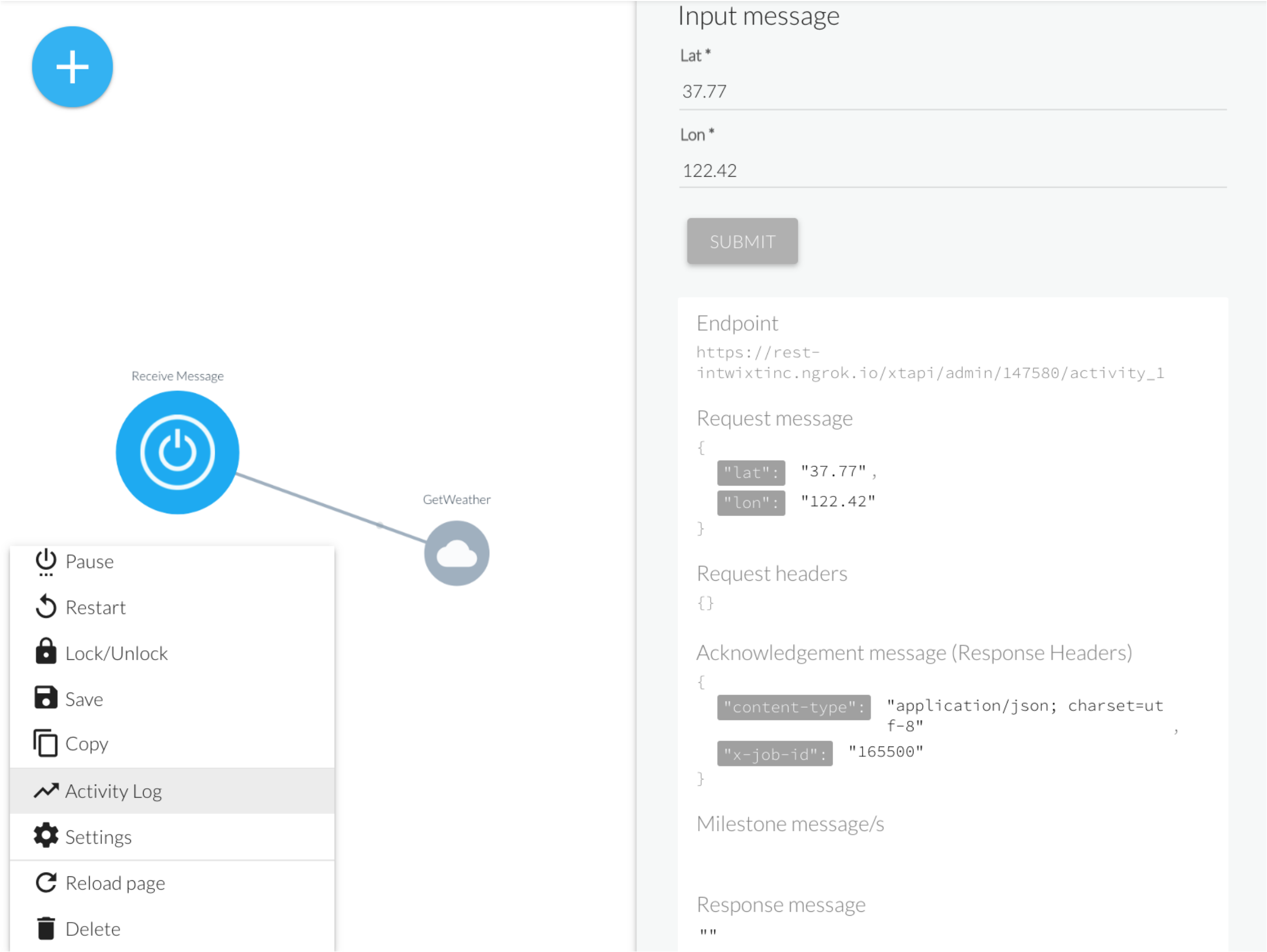
Request headers
```
{}
```

Acknowledgement message (Response Headers)
```
{
  "content-type":  "application/json; charset=utf-8",
  "x-job-id":  "165500"
}
```

Milestone message/s

Trigger Configuration Panel | TEST Tab with results

In order to see the results, click the menu button in the lower left corner and choose **Activity Log**.



Input message

Lat *

37.77

Lon *

122.42

Endpoint

```
https://rest-intwixtinc.ngrok.io/xtapi/admin/147580/activity_1
```

Request message

```
{
  "lat": "37.77",
  "lon": "122.42"
}
```

Request headers

```
{}
```

Acknowledgement message (Response Headers)

```
{
  "content-type": "application/json; charset=utf-8",
  "x-job-id": "165500"
}
```

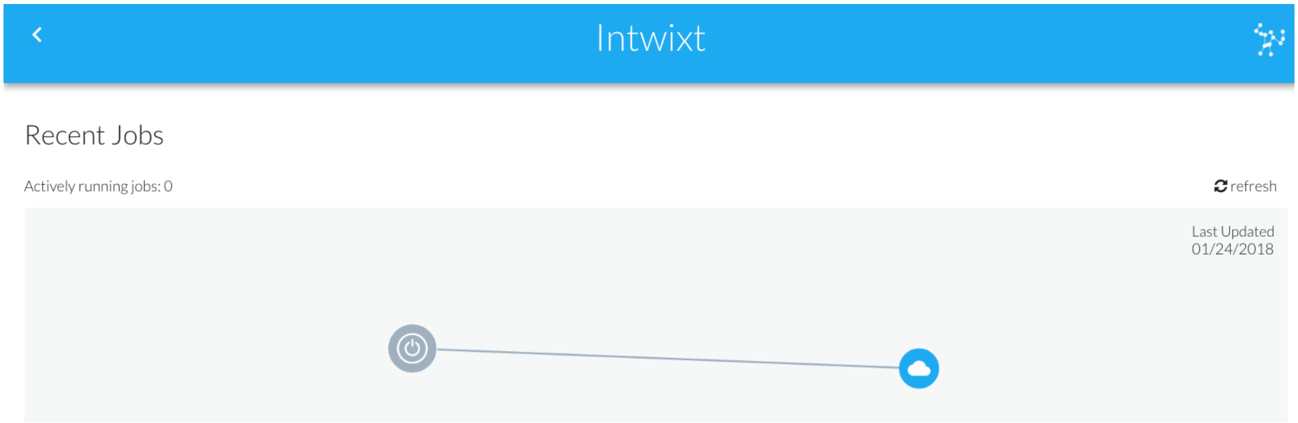
Milestone message/s

Response message

```
""
```

Intwixt Designer | Settings Menu

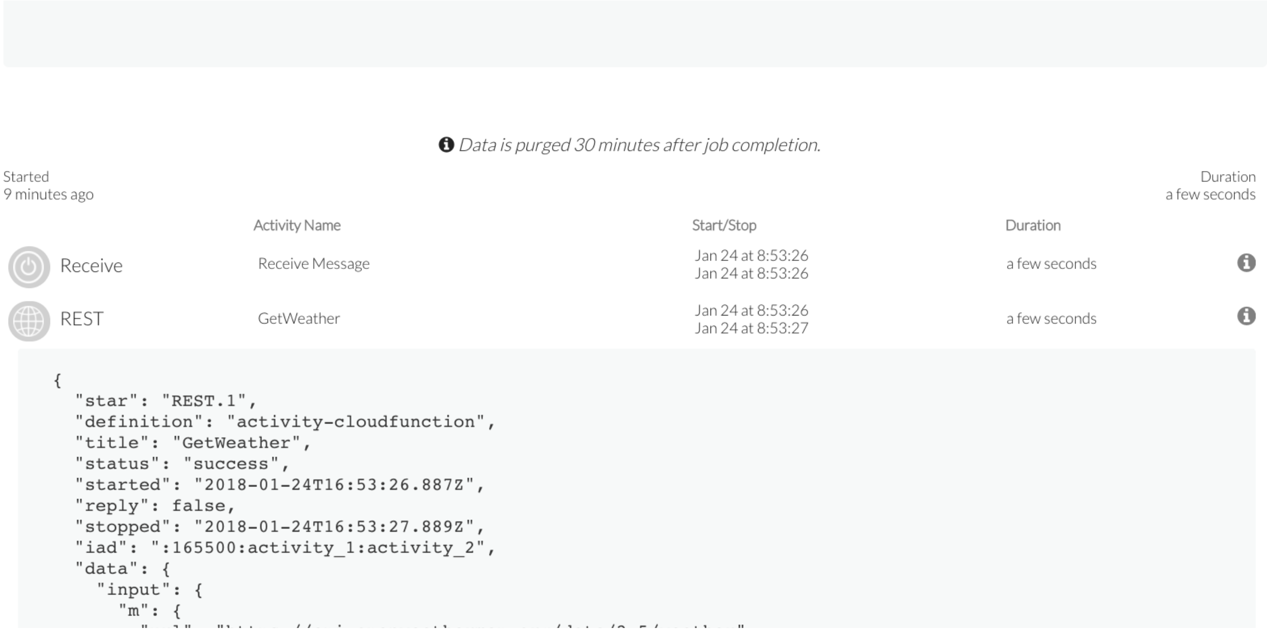
When the Activity Log loads, expand the messages to reveal the message exchange and test results.



Recent Jobs

Actively running jobs: 0

Last Updated 01/24/2018



Data is purged 30 minutes after job completion.

Started
9 minutes ago

Duration
a few seconds

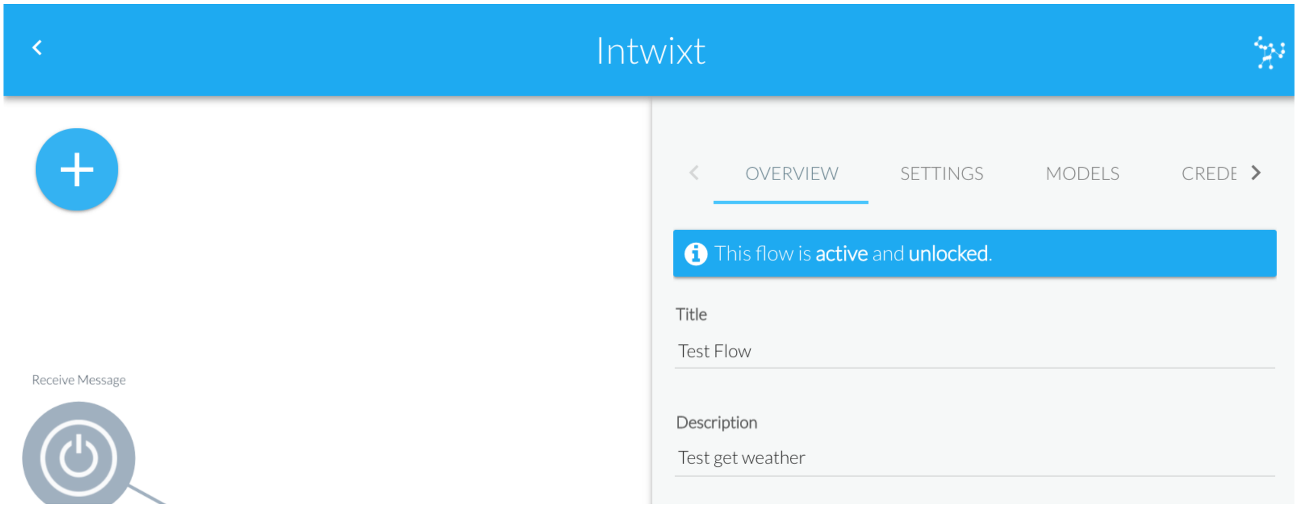
	Activity Name	Start/Stop	Duration
Receive	Receive Message	Jan 24 at 8:53:26 Jan 24 at 8:53:26	a few seconds
REST	GetWeather	Jan 24 at 8:53:26 Jan 24 at 8:53:27	a few seconds

```
{
  "star": "REST.1",
  "definition": "activity-cloudfunction",
  "title": "GetWeather",
  "status": "success",
  "started": "2018-01-24T16:53:26.887Z",
  "reply": false,
  "stopped": "2018-01-24T16:53:27.889Z",
  "iad": ":165500:activity_1:activity_2",
  "data": {
    "input": {
      "m": {
```

Activity Log | Results

Conclusion

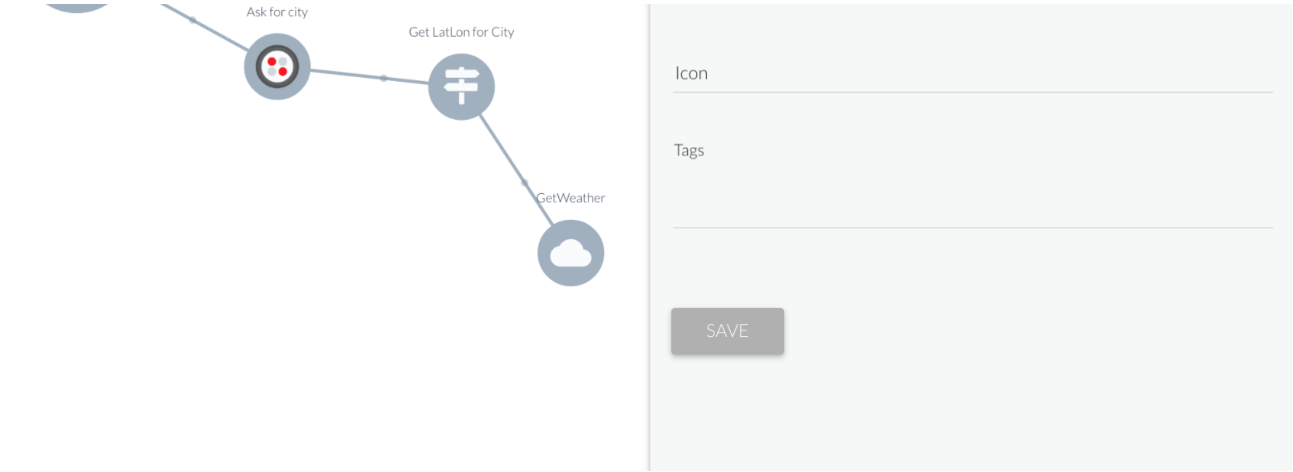
A single step flow isn't very interesting, but it begins to reveal what's possible as you connect multiple services into cohesive application units. Here, for example, is an extended example that begins by sending an SMS to a user and asking them for a city. When the user eventually responds, a cloud function is called that resolves the latitude and longitude for the provided city. The output is then passed to the GetWeather cloud function to get the final value.



This flow is active and unlocked.

Title
Test Flow

Description
Test get weather



Intwixt Designer showing a more advanced flow integrating APIs and Cloud Functions

Importantly, all cloud functions involved in the flow are stateless as intended. Only Intwixt has a notion of state, and once the interaction is complete, all vestiges disappear.

Comment 1 Likes Share

Comments (0)

Newest First Subscribe via e-mail

Preview POST COMMENT…

Posted in how to, connectors *and* *tagged with* faas, function as a service, stateless, serverless

Newer / Older

TERMS OF SERVICE PRIVACY POLICY VIDEOS

email info@intwixt.com
phone 415.570.8502

Disclaimer: Intwixt is a general integration platform, capable of connecting third party services through standard Web APIs. Logos, titles, and descriptions do not imply affiliation or partnership