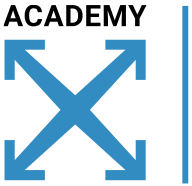


Edge Analytics Online Training

ADVANCED EDITION

B1 – Universal Connectors

Build reusable modules for REST APIs

crosser

Session B1

Agenda

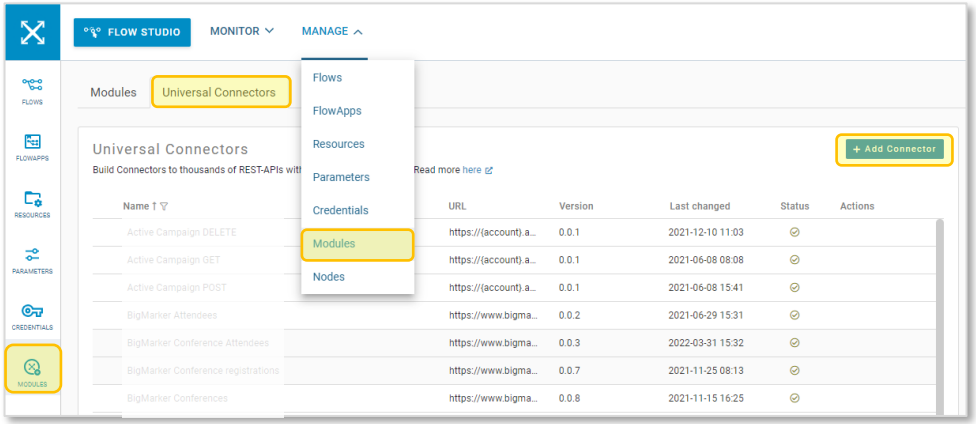
- Introduction to Universal Connectors
- The Universal Connector Wizard
- Exercises

crosser

Universal Connector

Introduction

- Use it to build your own modules that connect to REST API:s
- Simplify reuse by hiding details and parameterizing requests
- Configure requests by combining user settings with message data
- Use credentials from the vault (Basic, Bearer, Oauth)
- Ends up in the module library
- Can be used to get data (input) or send data (output)
- Add custom icon and documentation
- Create with Step-by-step Wizard
- Available from:
  Modules page → Universal Connector



Open Weather

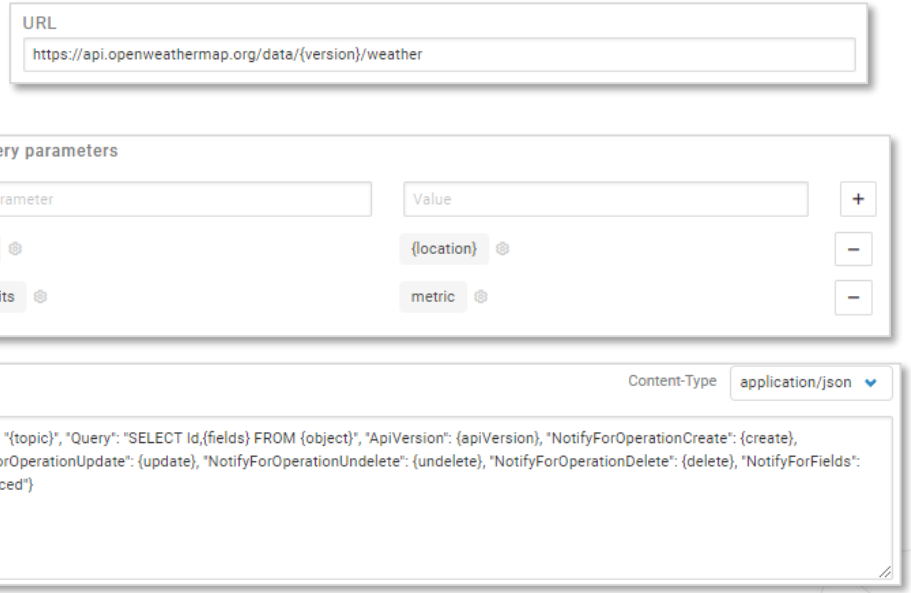
https://api.openweathermap.org/data/{version}/weather?q={cityName}

crosser

Universal Connector

Template syntax

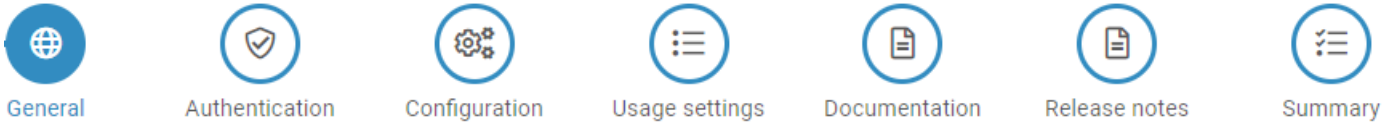
- All text input on the Authentication and Configuration steps support template syntax:
  - URLs
  - Query parameter name and value
  - Header name and value
  - Body content
- Anything inside {} will be treated as a parameter
  - Parameters must be a single word without space or other separators
  - Parameter names matched against properties on incoming messages (root level)
- Parameters can get values from:
  - User setting (module UI)
  - Incoming messages



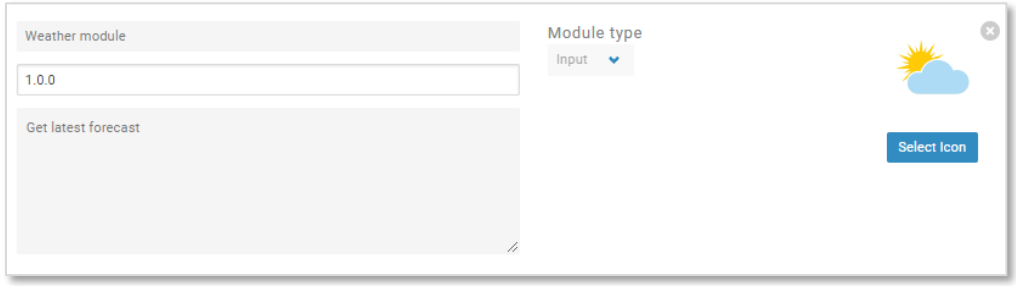
crosser

Universal Connector

The Wizard – Step 1: General



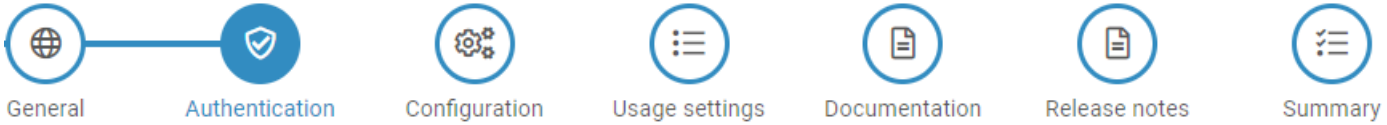
- Name
  - Preferably unique, to avoid confusion
- Version (x.x.x)
  - Must use this scheme
  - Must increase with each new version
- Description (optional)
  - Only seen in the wizard
- Input or Output
  - When set to ‘Input’ a ‘Target Property’ is added to hold the received data.
  - Also used by the ‘Module Types’ grouping in the Flow Studio module browser
- Icon
  - Supported formats: PNG, JPG, **SVG**
  - Recommended size: 256 x 256 px



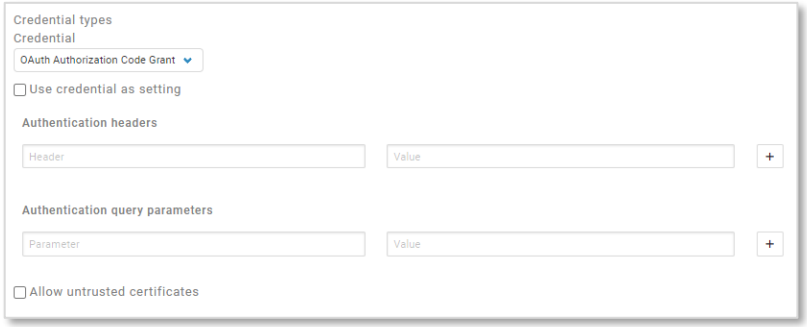
crosser

Universal Connector

The Wizard – Step 2: Authentication



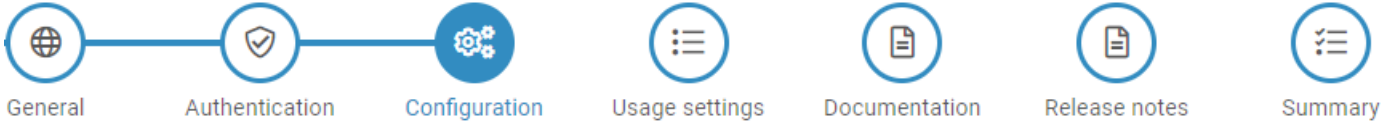
- Standard authentication methods:
  - Basic
  - API keys (Bearer)
  - Oauth (Authorization Code Grant and Client Credential Grant)
- Custom authentication methods:
  - Add custom headers and/or query parameters
  - Value can be:
    - Static – Added in the Wizard
    - A setting - e.g: {appKey}
      - A user setting
      - Message data
      - From a stored credential



crosser

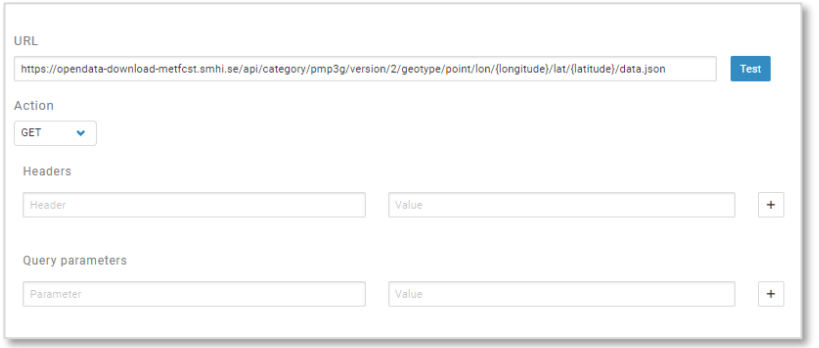
Universal Connector

The Wizard – Step 3: Configuration



Define your API request

- **URL**
  - Add domain and path, no query parameters
  - Make any parts of the URL configurable through template syntax. Even the whole URL can be configurable.
- **Action**
  - GET, PUT, POST, DELETE, OPTIONS, HEAD, PATCH, TRACE
- **Headers/Query parameters**
  - Lists of Key/Value pairs, use template syntax for configurable keys/values (cannot be combined with query parameters in the URL, you have to chose one option)
- **JSON convert response body** (enable if JSON payloads)



crosser

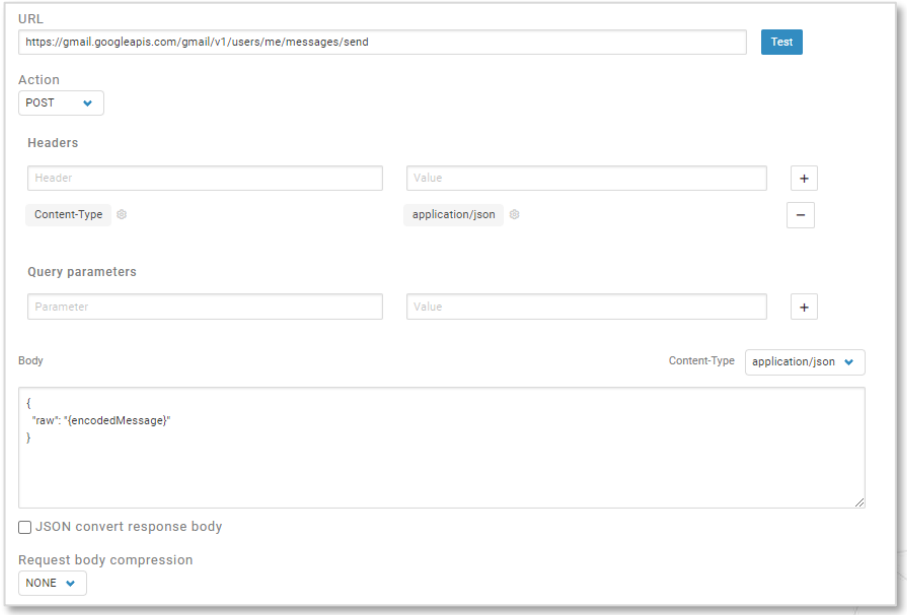
Universal Connector

The Wizard – Step 3: Configuration (contd.)

General — Authentication — Configuration — Usage settings — Documentation — Release notes — Summary

For POST, PUT, PATCH:

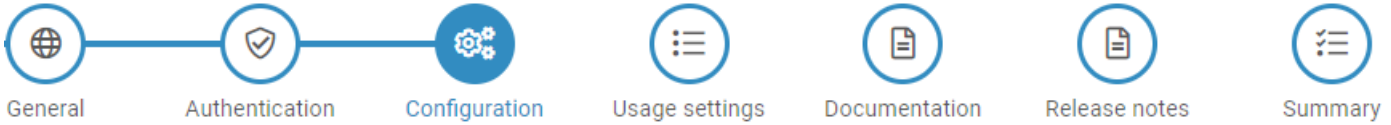
- **Content-Type**
  - Pre-defined types:
    - application/json
    - application/xml
    - text/plain
  - Use custom header for other types
- **Body**
  - Take the whole body from the incoming messages: {body}
  - Use template syntax to specify a specific body structure with some data from the messages or settings: From={from}&To={to}
- **Request body compression** (useful if large payloads are expected)

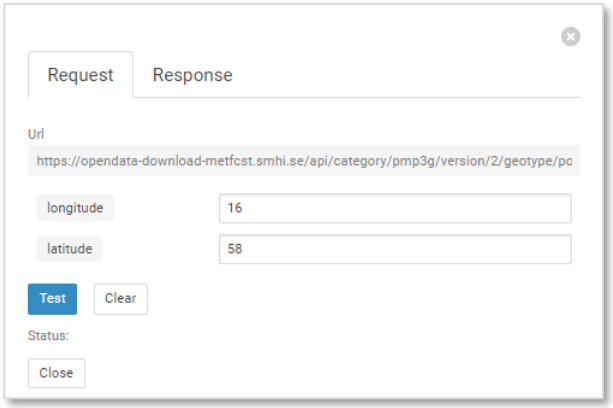


crosser

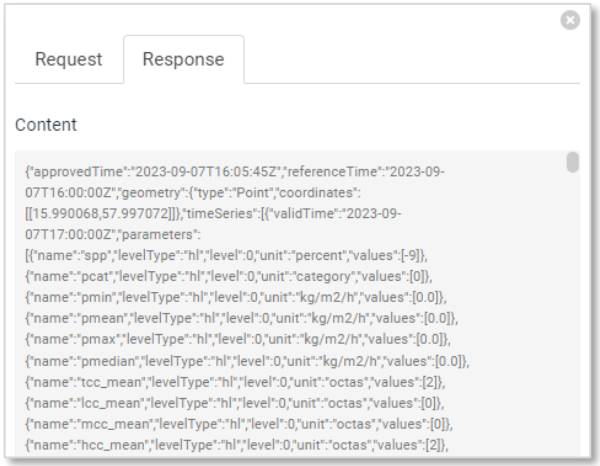
Universal Connector

The Wizard – Step 3: Test





Test result

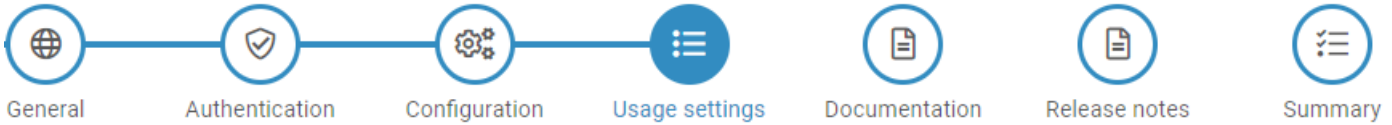


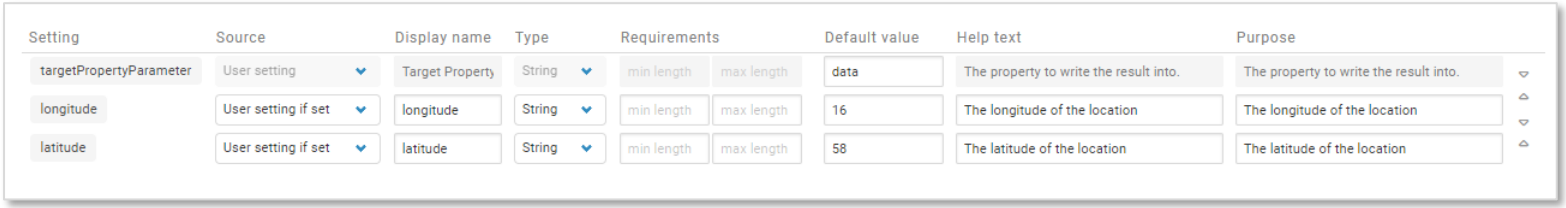
- Test and verify the connection in the wizard
- Use settings when testing
- Shows status and response message
- Cannot use credentials from the vault

crosser

Universal Connector

The Wizard – Step 4: Usage settings





Setting	Source	Display name	Type	Requirements		Default value	Help text	Purpose
targetPropertyParameter	User setting	Target Property	String	min length	max length	data	The property to write the result into.	The property to write the result into.
longitude	User setting if set	longitude	String	min length	max length	16	The longitude of the location	The longitude of the location
latitude	User setting if set	latitude	String	min length	max length	58	The latitude of the location	The latitude of the location

- **Source (where to get parameter value from)**
  - *User Setting*
  - *Message parameter*
  - *User setting if set*

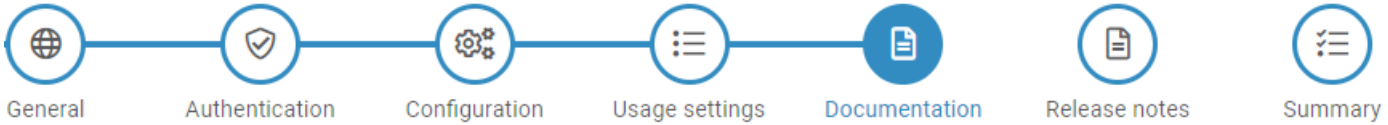
User settings (module UI)

- **Display name** – The name the property will have in the module settings UI
- **Type -** String, Number, Boolean, Credential
- **Requirements** (optional) – length, min/max
- **Default value** (optional) – Set the default value for the property
- **Help text** (optional) – Property help text in module settings documentation
- **Purpose** (optional) – Property purpose text in module settings documentation

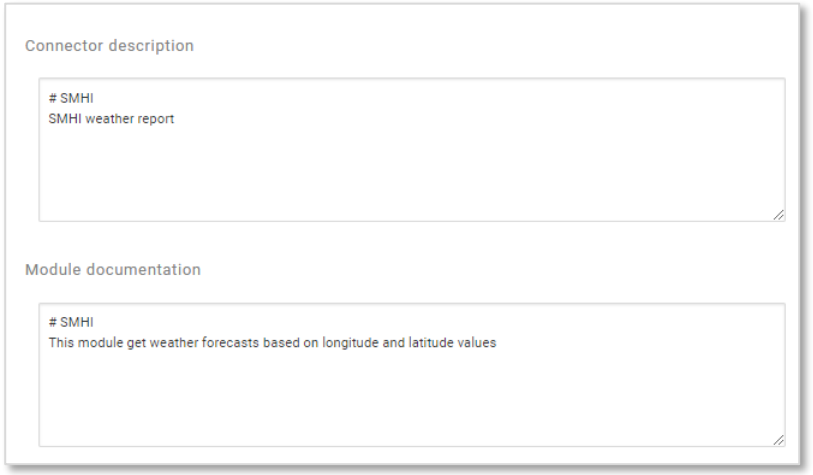
crosser

Universal Connector

The Wizard – Step 5: Documentation



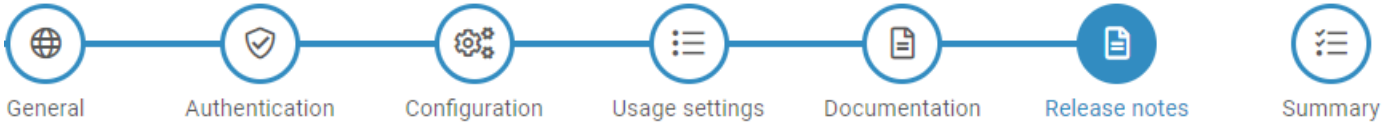
- Add documentation (optional)
  - Use Markdown syntax
- Connector description
  - Short description of the module
  - This is the text that will be seen in the tooltips when browsing the Module library in the Flow Studio
- Module documentation
  - Describe what the module is doing
  - This text is the top text in the Module / Documentation tab.
- Post Settings documentation
  - This is the text that appears after the settings table
  - Use to describe conditions on input messages and examples



crosser

Universal Connector

The Wizard – Step 6: Release notes

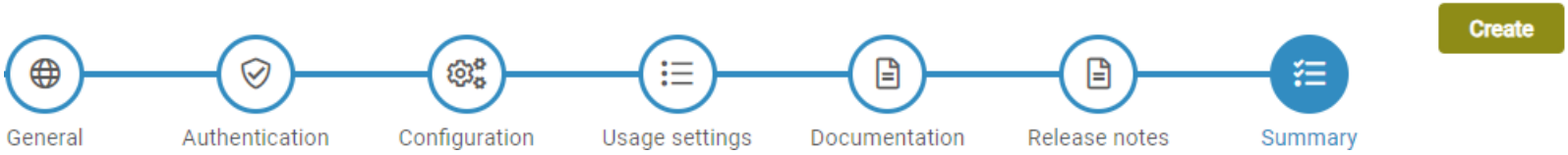


- Add release notes (optional)
  - Use Markdown syntax
  - Describe changes in the new version
  - See Crosser modules for styling examples

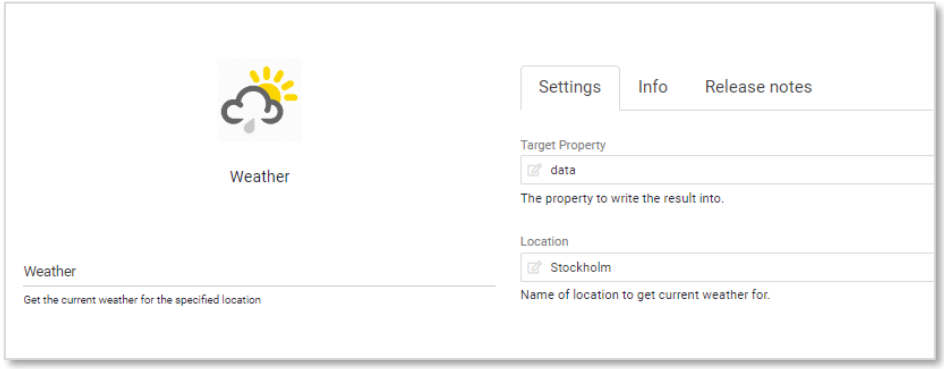
crosser

Universal Connector

The Wizard – Step 7: Summary



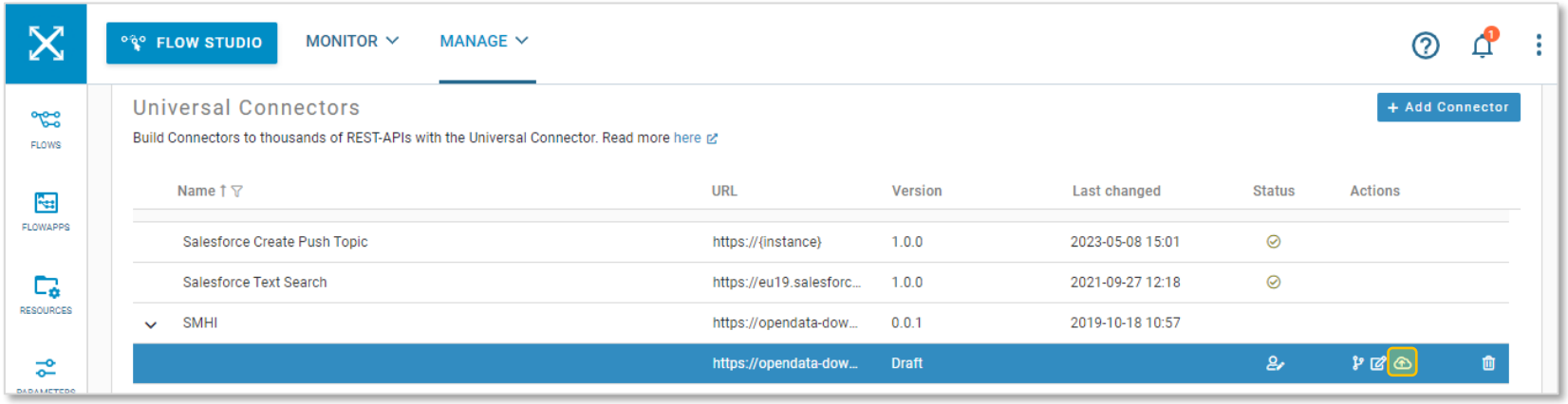
- Review the look and feel of the UI
- Press Create to create the Universal Connector



crosser

Universal Connector

Publish your new connector module



- Click Publish version to publish it as a new module
  - Now it will be available in the Flow Studio as any other module
  - Only available within your organization
  - Once published the version will be read-only
- Make changes
  - It is possible to create a new version of the Universal Connector
  - It is also possible to create a new one form an existing Universal Connector

crosser

EXERCISE B1

Build your first Universal Connector modules

Exercise B1.1

Overview

In this exercise you will build a connector to get stock prices using an online service

- You need to sign-up for a free account at Polygon.io, select the Basic option.

We will then start by building a connector that makes a static request to get data for a specific ‘ticker’ and then make it more configurable/dynamic.

Exercise B1.1

First version – Static request

1. Create a new Universal Connector
2. *General* Settings:
    1. Name: Polygon Get Ticker
    2. Module Type: Input
    3. Version: 1.0.0
    4. Select Icon (optional): For example, make a screenshot of their icon on the web page
3. *Configuration* Settings:
    1. URL: Go to https://polygon.io/docs/stocks/get_v1_open-close__stocksticker___date and copy the example URL
    2. Action: GET
    3. JSON convert response body: Enabled
    4. Test the module using the Test feature (You should get an output similar to the example in Polygon's documentation)

crosser

Exercise B1.1

First version – Static request (contd.)

5. Create and Publish the module
6. Create a new Flow and add the module.
7. Run the Flow in a remote session and check the output.

crosser

Exercise B1.1

Second version – Dynamic request

The module is not very useful as it is now. We can only get data for a single ticker symbol (AAPL) and at a fixed date. Also, the API key is exposed in the URL. Let's make the request more dynamic and secure!

Make the ticker symbol and date configurable, and get the API key from the credentials vault:

1. Create a new draft version of the module (1.1.0)
2. On the *Authentication* step:
    1. Credential: API Key
3. On the *Configuration* step:
    1. In the URL replace "AAPL" with "{tickerSymbol}" and the date with "{date}" (https://api.polygon.io/v1/open-close/{symbol}/{date})
    2. In the URL, remove the query string (everything from the '?' to the end of the URL). Note that you cannot have both query parameters added in the URL and in the configuration list. You may want to copy the API key, you will need it later.
    3. In the *Query Parameter* list, add a parameter 'adjusted' with a value 'true' (don't forget to click the '+' button to add it to the list).

crosser

Exercise B1.1

Second version – Dynamic request (contd.)

3. On the *User Settings* step, make the following changes:
    1. On the row for the *tickerSymbol* setting:
        1. Source: User setting
        2. Type: String
        3. Display name: "Symbol"
        4. Help Text and Purpose: "Ticker symbol, e.g. 'AAPL' for Apple"
    2. On the row for the *date* symbol:
        1. Source: Message parameter
        2. Purpose: "Date for which to get closing price."
4. Test the module on the Configuration step. Note that you have to enter the API key manually. The test tool cannot access the credentials vault.
5. On the *Summary* step check the UI and the documentation to see the result of your configuration.
6. Update and Publish the new version

crosser

Exercise B1.1

Second version – Dynamic request (contd.)

We will now update our Flow so that it runs once per day and give us the pricing details from the day before (Polygon only makes yesterday's data available with the free account)

Create a new Flow, or use the one you have:

1. Add a *Scheduler* module and configure it to run once per day, at a time of your choice. Make sure to leave "RunOn Start" enabled (will make testing easier)
2. Add a *Time Stamp* module:
    1. Target Property: date
    2. Output Format: Custom
    3. Custom Format: yyyy-MM-dd
    4. Offset Time Unit: Days
    5. Offset Value: -1

Exercise B1.1

Second version – Dynamic request (contd.)

3. Add the latest version of your Universal Connector module:
   1. Symbol: Any symbol of your choice, e.g. “GOOGL” for Google
   2. Credential: Create a new credential (‘+’ button):
      1. Name: Polygon API key
      2. API Key: Copy the API key from the Polygon docs or your previous version of the module
4. Run the Flow and check the output

Exercise B1.1

Wrap-up

Things to test/consider:

- Why did we add an offset of '-1' in the Time Stamp module and why did we have to use a custom format?
- Why could we replace the API key in the URL with a pre-defined credential type?
    - Take a look at *the request.headers* object in the debug output and note that the API key is not exposed.
- Take a look at the *request.url* property in the debug output to see how the URL has been constructed by the module.
- Make a new version of the module and try the *Documentation* step, where do the texts end up in the UI?
- How can you change the Flow to get pricing details for several symbols?

crosser

Exercise B1.2

Overview

In this exercise you will build an output connector. We will use an internal endpoint but you can modify it for use with an external service as an extra exercise.

We will create a message based on data received in the previous exercise and then POST this to a REST endpoint to a fake text messaging service.

crosser

Exercise B1.2

Build an output module

1. Create a new Universal Connector
2. *General* Settings:
    1. Name: Send Message
    2. Module Type: Output
    3. Version: 1.0.0
3. *Configuration* Settings:
    1. URL: http://localhost:9090/sendMessage
    2. Action: POST
    3. Add a "Content-Type" header and set the value to "application/x-www-form-urlencoded"
    4. Body: "Body={message}&From={from}&To={to}"
    5. JSON convert response body: Enabled

crosser

Exercise B1.2

Build an output module (contd.)

3. On the *User Settings* step, make the following changes:
    1. On the row for the *message* setting:
        1. Source: Message parameter
    2. On the rows for the *from/to* symbols:
        1. Source: User setting if set
        2. Display Name: "From"/"To"
        3. Type: String
        4. Purpose: "Sending/Receiving phone number with '+46123456…' syntax."
4. Create and Publish the new module

crosser

Exercise B1.2

Use the output module

Open the Flow you created in the previous exercise and make the following changes:

1. Add a *Text Template* module:
    1. Target Property: message
    2. Template: "The closing price for {data.body.symbol} on {date} was {data.body.close} dollars"
2. Add your *Send Message* module:
    1. Enter some (fake) phone numbers in the From/To settings
3. Add a *HTTP Listener* module:
    1. The HTTP Request path to match: sendMessage
4. Run the Flow and check the output from your module:
    1. Check the *request.body* and *request.headers* objects

crosser

Exercise B1.2

Wrap-up

- Why did we set the “Target Property” in the *Message Template* module to “message”?
- Why did we have to add the *HTTP Listener* module (what happens if you disable it)?
- Open the settings for your module in the Flow Studio and expand the “Read-only” section at the bottom, what do you see here?

Extra exercise:

The output module is very close to what is needed to use a real messaging service. You can sign up for a free Twilio account and then make these modifications to the module:

- Select ‘Username and Password’ credential on the authentication step
- Change the URL to “https://api.twilio.com/2010-04-01/Accounts/{accountSID}/Messages.json”
- Add a new User setting for the *accountSID* parameter
- Now you can send stock info to your mobile phone!

crosser

SESSION – B1 END