Module Design – Call an API from a Logic Apps workflow using a custom connector

## Summary

Learn how to wrap an API in a custom connector and use it in a Logic Apps workflow.

## Learning objectives

At the end of this module the student should be able to:

* Create a custom Logic Apps connector
* Define the connectors behaviour using an OpenAPI definition
* Use the connector from a Logic App

## Prerequisites

* Have a basic familiarity with the authoring and running of Logic Apps
* Basic understanding of APIs and REST

## Expected Audience

Developer

## Design author and/or SMEs

James Elfred, [jameselfred@gmail.com](mailto:jameselfred@gmail.com), Alistair Matthews, [alistairmatthews@contentmaster.com](mailto:alistairmatthews@contentmaster.com)

## Image and video requirements

Screenshots of Azure

## Additional references

* What is Azure Logic Apps? <https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-overview>
* Custom connectors in Logic Apps: <https://docs.microsoft.com/en-us/azure/logic-apps/custom-connector-overview>

## Additional comments

# Unit 1: Motivation

## Learning objective(s)

By the end of this unit, the student will be able to identify a business problem that they can address by using a custom Logic Apps connector.

## Enablement tasks / details

You are the senior lead developer at a picture framing company. Some of your company’s picture-frame-related data is only accessible through a custom-built, in-house REST API. You have plans to build many automated workflows using Logic Apps that need access to this data, such as automating emailed quotes to customers.

You’ll need to create a custom connector for the frame data REST API in order for your developers to access it from your Logic Apps.

This unit also contains:

* Learning Objectives
* Prerequisites

# Unit 2: Learning: Connectors for Azure Logic Apps

## Learning objective(s)

By the end of this unit, the student will be able to explain the role of connectors in Azure Logic apps.

## Enablement tasks / details

### Motivation

### Information presented in this unit

* Overview of Azure Logic Apps (be brief!)
* Concept of connectors, triggers, actions, references
* Overview of prebuilt connectors for Azure Logic Apps (categories, how to find them etc.)
* Describe the technical scenario we are using throughout this module – we want to call an API from our Logic Apps workflow. In this module, the trigger for the simple workflow will be a HTTP trigger and the response

### How to do the task

## Video requirements

N/A

## Knowledge test objectives

## External dependencies

* Azure Logic Apps.
* Azure Logic Apps API

## Necessary development resources

NONE

## Reference links

* Create a custom connector in Azure Logic Apps: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/create-logic-apps-connector>
* Create a custom connector from an OpenAPI definition: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/define-openapi-definition>

# Unit 3: Exercise – Setup a Logic App and a test API

The module should NOT copy the documentation example. Instead, we should take advantage of the sandbox experience. In this unit, give the user the steps needed to setup a pre-baked API in their sandbox session.

* Download a .NET Core Web API project from GitHub
* Deploy API to Azure sandbox
* Test using cUrl or another command-line tool
* The repo should also contain an OpenAPI definition of the API
* API should be related to the scenario (picture-framing business). For example, pass frame dimensions to the API and it gives back picture area. Pass dimensions and material of frame to the API and it gives back price.

Also layout the instructions needed to create a Logic App in the portal that is triggered by HTML and responds over HTML. The Logic App should accept parameters that can be passed on to the custom connector. Instead of just responding over HTML, add an email connector and send out an email with the results of the API call. How exactly the Logic App is triggered and responds is up to you – just giving an example of what I think might be the most straightforward way of demonstrating the concept.

# Unit 4: Learning – Access an API with a Logic Apps custom connector

## Learning objective(s)

By the end of this unit, the student will be able to explain the role of custom connectors and how to use the Azure Portal to create one for use with a Logic App to connect to a REST API.

## Enablement tasks / details

### Motivation

### Information presented in this unit

* **Characteristics of a custom connector**
* **Ways to create a custom connector (from scratch, OpenAPI, Postman etc. )**
* The role of Json for outlining the API Authentication
* The use of API key definitions
* The role of actions, triggers and references
* Testing limitations i.e. at present restricted to Microsoft Flow and PowerApps
* Describe the technical scenario we are using throughout this module – we want to call an API from our Logic Apps workflow. In this module, the trigger for the simple workflow will be a HTTP trigger and the response

### How to do the task

* The process of creating a custom connector in the Azure Portal
* How to import an OpenAPI definition via the Azure Portal

## Video requirements

N/A

## Knowledge test objectives

## External dependencies

* Azure Logic Apps.
* Azure Logic Apps API

## Necessary development resources

NONE

## Reference links

* Create a custom connector in Azure Logic Apps: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/create-logic-apps-connector>
* Create a custom connector from an OpenAPI definition: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/define-openapi-definition>

# Unit 5: Exercise – Create and call a Logic Apps custom connector that wraps an API

## Lab environment

Azure Portal

Sandbox

## Enablement tasks / details

### Motivation

### Exercise steps

* Get an API key
* Create an Azure Logic Apps custom connector
* Import the OpenAPI definition into the Azure Logic Apps connector created earlier
* Create a Logic App
* Download the Custom Connector
* Test the Logic App by adding a trigger and an action that use the custom connector

## Validation steps

N/A

## External dependencies

* Azure Logic Apps.
* Azure Logic Apps API

## Necessary development resources

* An OpenAPI definition for the frame data REST API. This must be less than 1 MB
* An OpenAPI key for the frame data Rest API

## Reference links

* Create a custom connector in Azure Logic Apps: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/create-logic-apps-connector>
* Create a custom connector from an OpenAPI definition: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/define-openapi-definition>

# Unit 6: Learning - Make your custom connector available to the entire organization

* Talk about [sharing a connector](https://docs.microsoft.com/en-us/connectors/custom-connectors/share)s
* Talk about [certifying a connector](https://docs.microsoft.com/en-us/connectors/custom-connectors/submit-certification)
* Quick 1 – 2 question knowledge check at the end of this unit

# Unit 7: Summary + Clean-up

## Summary statement

In this module you have learned how to create a custom Logic Apps connector. To connect Logic Apps to a to custom REST API, you must describe that API by supplying a definition in OpenAPI format. Then you can call functions in that API from actions in a Logic App.

Now that you’ve creating the connector, you can also use it to connect Microsoft Flow and PowerApps to your REST API.

## Cleanup requirements

The exercises in this module are sandboxed. Changes will be automatically lost after the sandbox expires. Use the azure-sandbox-cleanup.md include file to point this out to students.

## Reference Links

* Create a custom connector in Azure Logic Apps: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/create-logic-apps-connector>
* Create a custom connector from an OpenAPI definition: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/define-openapi-definition>
* Use a custom connector from a logic app: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/use-custom-connector-logic-apps>
* Use a custom connector from a flow: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/use-custom-connector-flow>
* Use a custom connector from a PowerApps app: <https://docs.microsoft.com/en-gb/connectors/custom-connectors/use-custom-connector-powerapps>