✓ 100 XP

Access an API with a Logic App custom connector

5 minutes

As the lead developer for a print framing company, you want to call an in-house Web API from a Logic Apps workflow. The API calculates a price for a picture frame, based on the dimensions of the frame. To access the API from your Logic Apps workflow, you'll need to first create a custom connector.

In this unit, you'll learn how to create and use custom connectors within a Logic App.

Why custom connectors?

Sometimes pre-built connectors are not enough to satisfy your workflow scenario. If you have in-house, custom, or less-known API you want to call from your workflow, the out-of-the-box connectors you get with Logic Apps won't help. You must create a connector that describes that API. Include Triggers when you want the Logic App to respond to an event in that API. Include Actions when you want the Logic App to make a call to that API.

Describing a Web API for a custom connector

You can create a custom connector in the Azure portal and reference the target API. The API must describe itself to the custom connector, so that the connector can present the right methods and parameters to Logic Apps. In the Web API, you can implement this description by using an **OpenAPI** definition or a **Postman** collection. Let's examine these two methods of describing an API a little further.

Postman is an app that requires you to provide the request URL and any authentication needed. You also specify the API Key and the content type along with the definition of the request body in JSON format. When you send the request from Postman, the API returns a response. Postman uses the response to generate a collection. You can export the collection and use it to describe the API to a custom connector.

An **OpenAPI** definition file is a JSON file that lists the API's methods, parameters, and outputs. In .NET, you can create an OpenAPI file by adding the **Swashbuckle** NuGet package to your API's project. Then, add the code below to the Web API's source code.

In the **Startup.cs** file, add a `using` statement for Swashbuckle:

c# Copy

```c#
using Swashbuckle.AspNetCore.Swagger;
```

Then add this code to the **ConfigureServices** method:

c# Copy

```c#
services.AddSwaggerGen(c =>
{
  c.SwaggerDoc("v1", new Info { Title = "My API", Version = "v1" });
});
```

Finally, add this code to the **Configure** method:

c# Copy

```c#
// Enable middleware to serve generated Swagger as a JSON endpoint.
app.UseSwagger();

// Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
// specifying the Swagger JSON endpoint.
app.UseSwaggerUI(c =>
{
  c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
});
```

When this code is deployed, it installs the Swagger UI, which is a web page that you can use to test your Web API methods. It also generates and publishes a JSON file, compliant with the OpenAPI standard, that describes your API. You can download that file and use it to create a custom connector in Logic Apps.

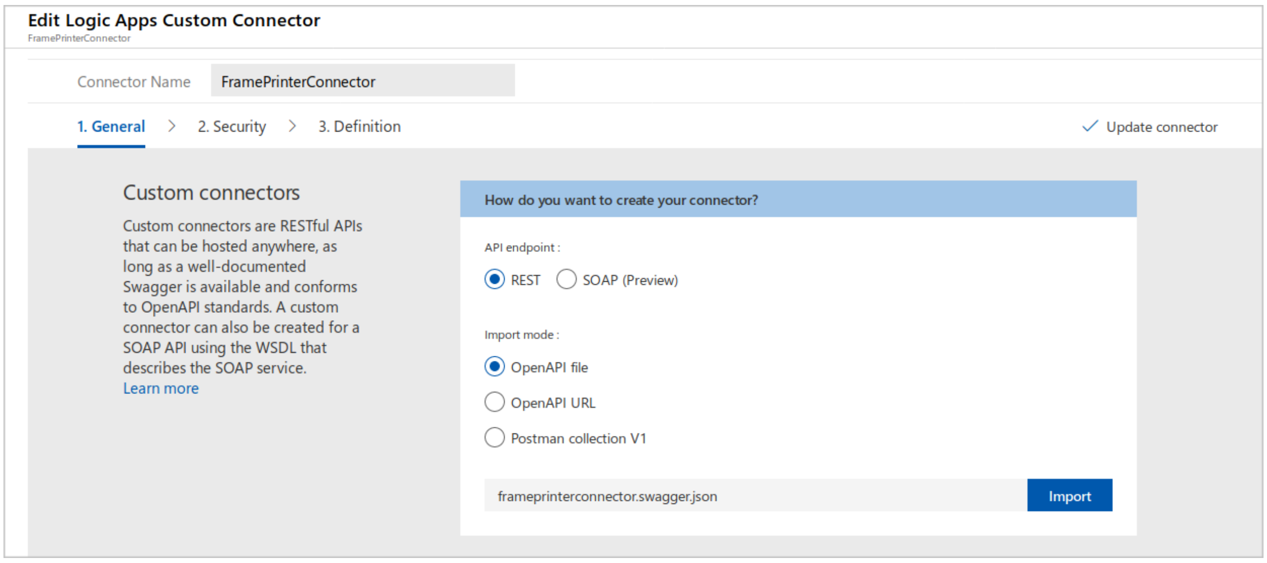
Note

In the previous exercise, you deployed a pre-built Web API to Azure. This included the above Swagger code by default. In the next exercise, you will upload the OpenAPI file that Swagger creates, to the custom connector.

How to create a custom connector in the Azure portal

Once you have implemented Postman or OpenAPI, you can build custom connectors in the Azure portal in this way:

1. Create a new Logic Apps custom connector resource.
2. Upload the Postman collection or OpenAPI definition file.



Edit Logic Apps Custom Connector
FramePrinterConnector

Connector Name: FramePrinterConnector

1. General > 2. Security > 3. Definition ✓ Update connector

Custom connectors

Custom connectors are RESTful APIs that can be hosted anywhere, as long as a well-documented Swagger is available and conforms to OpenAPI standards. A custom connector can also be created for a SOAP API using the WSDL that describes the SOAP service.
Learn more

How do you want to create your connector?

API endpoint: [✓ REST] SOAP (Preview)

Import mode: [✓ OpenAPI file] OpenAPI URL Postman collection V1

frameprinterconnector.swagger.json Import

3. Use the configuration wizard to complete the connector. For example, you can configure the title of the connector users will see in the Logic Apps designer, add descriptions to help Logic Apps developers to use the connector correctly, and configure secure connections.
4. Save your new connector. You can now add the connector to Logic Apps.

You will create a connector in the next exercise.

Next unit: Exercise - Create and call a custom connector from a Logic Apps workflow

Continue >