# Calling an API using UiPath

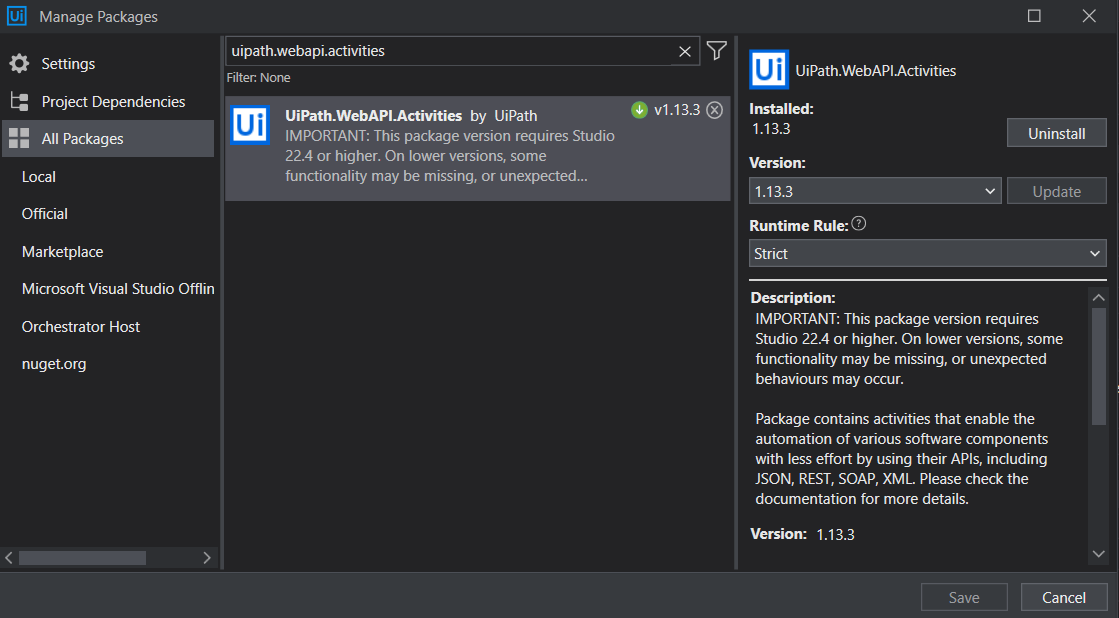
The default installation of UiPath Studio only provides the **Orchestrator HTTP Request** activity to call the UiPath Orchestrator API. Calling any other web API requires installing UiPath’s **Web API** package. This package includes activities to perform HTTP and SOAP requests and manipulate JSON and XML files.

This guide will go through some of the activities within the Web API package to create API requests and manipulate API responses. To successfully complete this guide, you should already have UiPath installed and be able to comfortably navigate the program.

**Note:** This package requires Studio version 22.4 or higher for proper functionality.

## Installing the Web API package

To install the WebAPI package:

1. In the Design panel, select **Manage Packages**.
2. Select **All Packages**, then type UiPath.WebAPI.Activities into the search bar. UiPath will start searching for the package automatically.
3. Select the **UiPath.WebAPI.Activities** result. Select **Install**, and then select **Save**. The package will begin installation.

UiPath.WebAPI.Activities package when installed.

## Making HTTP requests

The Web API package is now installed and ready to use. You can use the **HTTP Request** activity inside this package to call APIs through the HTTP protocol:

1. Search HTTP Request in the Activities panel. This activity is located under **App Integration->Web**.
2. Graphical user interface

   Description automatically generatedDrag and drop the **HTTP Request** activity into your workflow. This will automatically open the **HTTP Request Wizard**. The **HTTP Request Wizard** is one way you can input your endpoint, request methods, authentication token, and parameters, as well as preview your response and status code without having to execute your workflow.

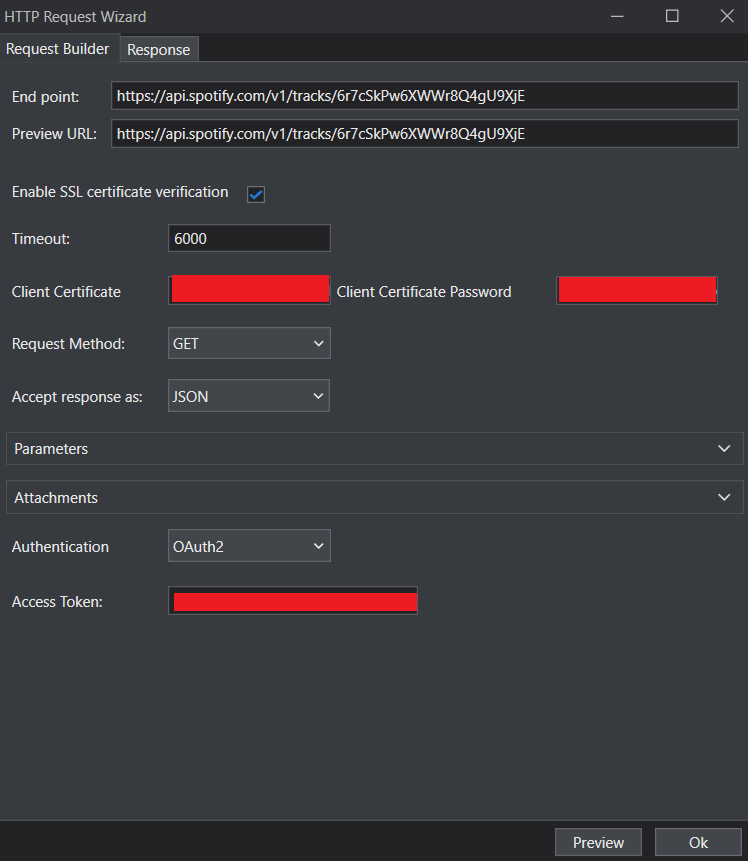
A screenshot of a computer

Description automatically generated with medium confidenceAlternatively, you can close the HTTP Request Wizard and enter your inputs into the activity’s **Properties** panel.

1. In the **Request Builder** tab in the HTTP Request Wizard, create your request by entering your inputs and customizing the settings.
2. Select **Preview** to preview your response result in the **Response** tab.

**Caution:** This will execute POST and DELETE requests, so be careful!

1. When finished, select **OK** to exit the HTTP Request Wizard.

The example below uses the HTTP Request Wizard to retrieve track information for the song “Until” by Cécile McLorin Salvant through the [Spotify Web API](https://developer.spotify.com/documentation/web-api/).

GET request for track information for "Until" through the HTTP Wizard.

Text

Description automatically generated

GET request response preview.

The following example uses the activity’s **Properties** panel to create a new playlist called “UiPath Playlist” on an account called “thepaperplane”. Note that the **Body** parameter takes in a string that is formatted like a JSON object.

Text

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidencePOST request to create a new playlist.

The request creates a new playlist called "UiPath Playlist" on thepaperplane's account.

## Making SOAP Requests

Use the **SOAP Requests** activity in **App Integration->Web** to call APIs using the SOAP protocol. This activity also uses a Wizard to enter requests and invoke responses.

Graphical user interface, application, Teams

Description automatically generated**Note:** While the HTTP Requests activity is available across all frameworks, the SOAP Requests activity is currently only available for projects using the *Windows – Legacy* framework that is currently being phased out.

SOAP Request Wizard

## Manipulating JSON files

The **WebAPI** package provides several activities to deserialize and extract information from JSON and XML files, including those created through API requests.

The **Deserialize JSON** activity takes in a string and deserializes it into a JSON object. Let’s use the previous Spotify GET example to learn how this activity can be used to extract the title and artist for a song.

1. A screenshot of a computer

   Description automatically generated with medium confidenceIn the **Properties** panel for the same **HTTP Request** activity as the Spotify GET example, create a new string variable in the **Response content** textbox.
2. Add a **Deserialize JSON** activity. Modify the Properties to where the **JSON string** argument is the Response content variable you just created, **TypeArgument** is Newtonsoft.Json.Linq.JObject, and **JsonObject** is a new variable that will store the JSON object output.
3. Add an **Assign** activity where the variable is songName, and the value is [JSONobjectvariable].SelectToken(“name”).ToString(). The **SelectToken** function uses JPath to search the JSON object for the header **name**, which holds the name of the song. The statement then grabs its value and converts it into a string.
4. Add another **Assign** activity for the variable artistName and the value [JSONobjectvariable].SelectToken("$['album']['artists'][0]['name']").ToString(). This statement traverses through the nested JSON object to find the artist’s name.
5. Add a **Write Line** activity with the text songName + “ by “ + artistName, then run the workflow. The output should print “Until by Cécile McLorin Salvant.”

Graphical user interface, application

Description automatically generated