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.

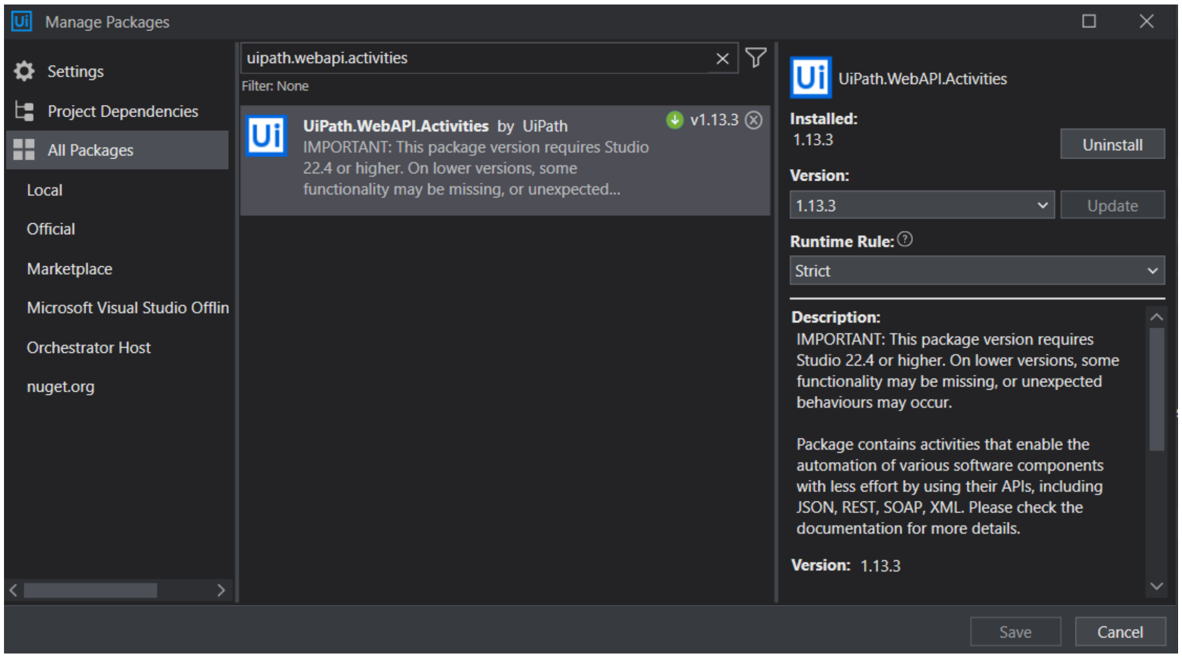
This document will go through some of the activities within the Web API package to create API requests and manipulate API responses.

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

Filter: None

UiPath.WebAPI.Activities by UiPath v1.13.3
IMPORTANT: This package version requires Studio 22.4 or higher. On lower versions, some functionality may be missing, or unexpected...

UiPath.WebAPI.Activities

Installed: 1.13.3

Version: 1.13.3

Runtime Rule: Strict

Description:
IMPORTANT: This package version requires Studio 22.4 or higher. On lower versions, some functionality may be missing, or unexpected behaviours may occur.

Package contains activities that enable the automation of various software components with less effort by using their APIs, including JSON, REST, SOAP, XML. Please check the documentation for more details.

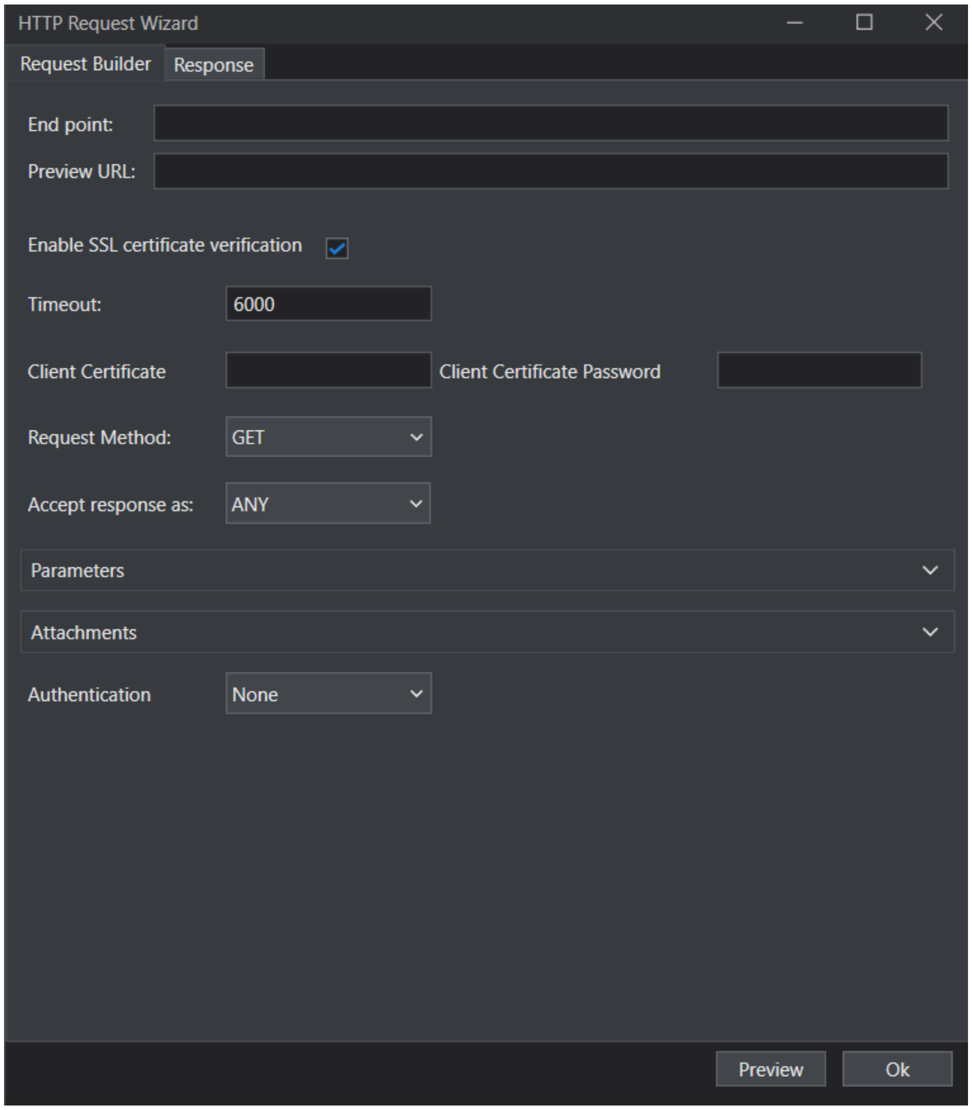
Version: 1.13.3

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. Drag 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.


HTTP Request Wizard

Request Builder | Response

End point:

Preview URL:

Enable SSL certificate verification ✓

Timeout: 6000

Client Certificate

Client Certificate Password

Request Method: GET

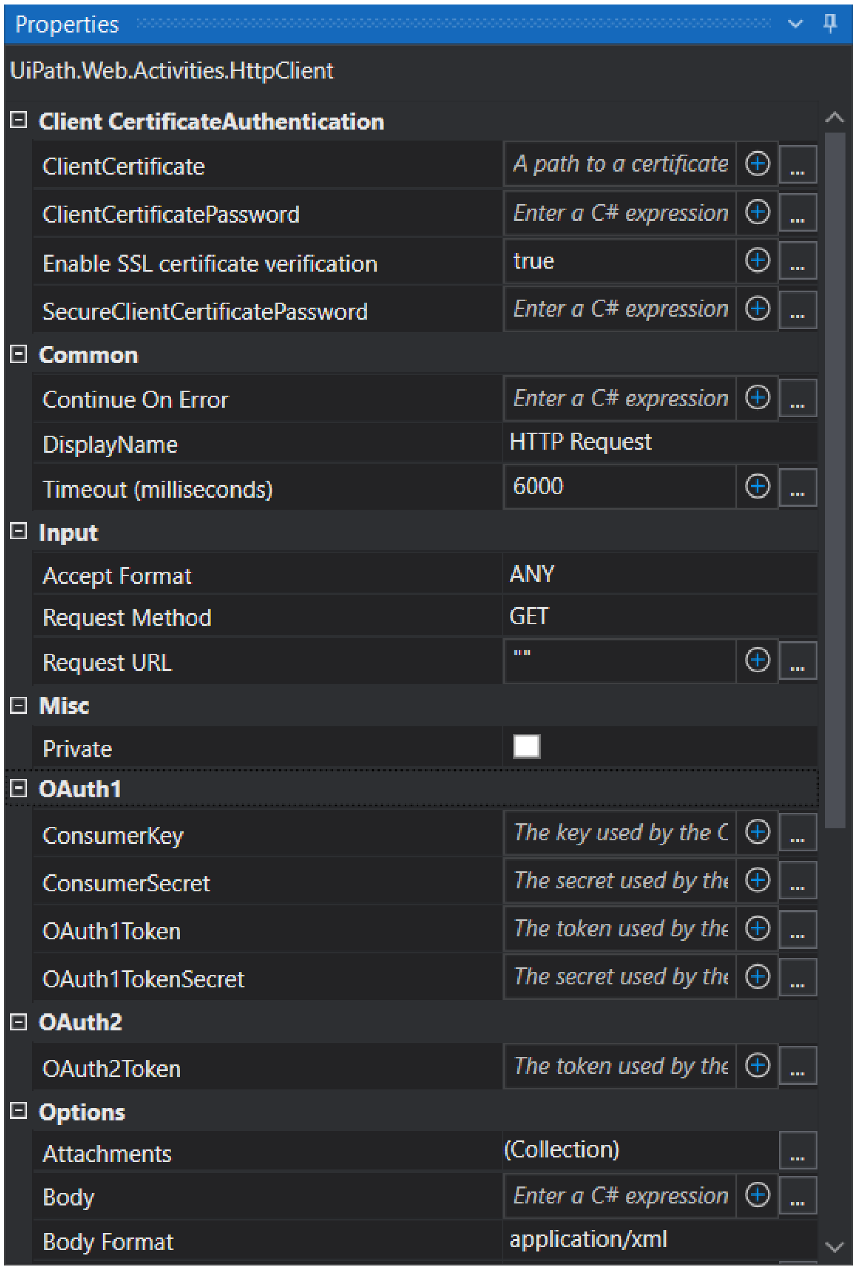
Accept response as: ANY

Parameters

Attachments

Authentication: None

Alternatively, you can close the HTTP Request Wizard and enter your inputs into the activity's **Properties** panel.



Properties

UiPath.Web.Activities.HttpClient

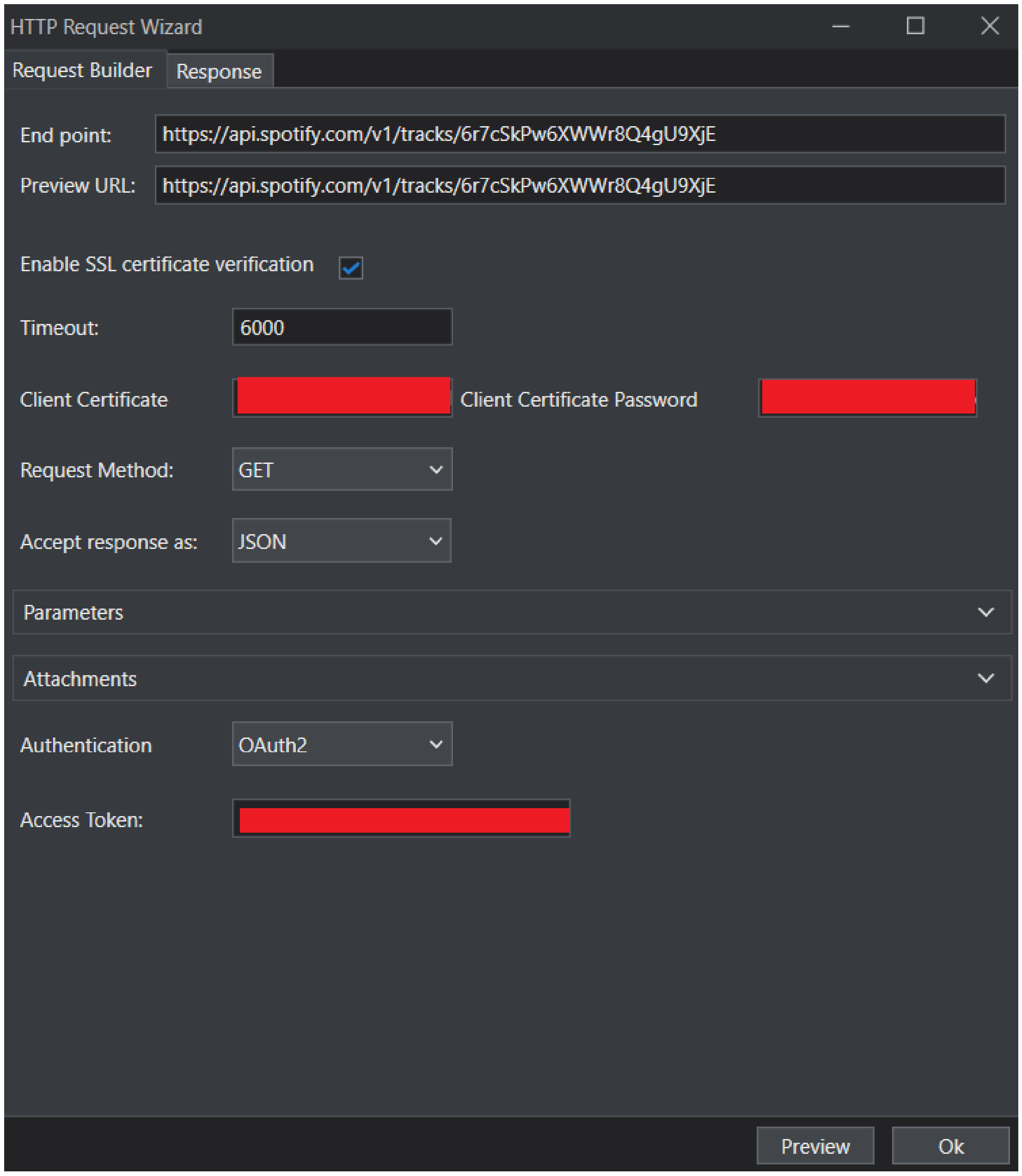
Client CertificateAuthentication	
ClientCertificate	A path to a certificate
ClientCertificatePassword	Enter a C# expression
Enable SSL certificate verification	true
SecureClientCertificatePassword	Enter a C# expression
Common	
Continue On Error	Enter a C# expression
DisplayName	HTTP Request
Timeout (milliseconds)	6000
Input	
Accept Format	ANY
Request Method	GET
Request URL	""
Misc	
Private	☐
OAuth1	
ConsumerKey	The key used by the C
ConsumerSecret	The secret used by the
OAuth1Token	The token used by the
OAuth1TokenSecret	The secret used by the
OAuth2	
OAuth2Token	The token used by the
Options	
Attachments	(Collection)
Body	Enter a C# expression
Body Format	application/xml

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

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

5. 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.



End point: https://api.spotify.com/v1/tracks/6r7cSkPw6XWWr8Q4gU9XjE

Preview URL: https://api.spotify.com/v1/tracks/6r7cSkPw6XWWr8Q4gU9XjE

Enable SSL certificate verification ✓

Timeout: 6000

Client Certificate [redacted] Client Certificate Password [redacted]

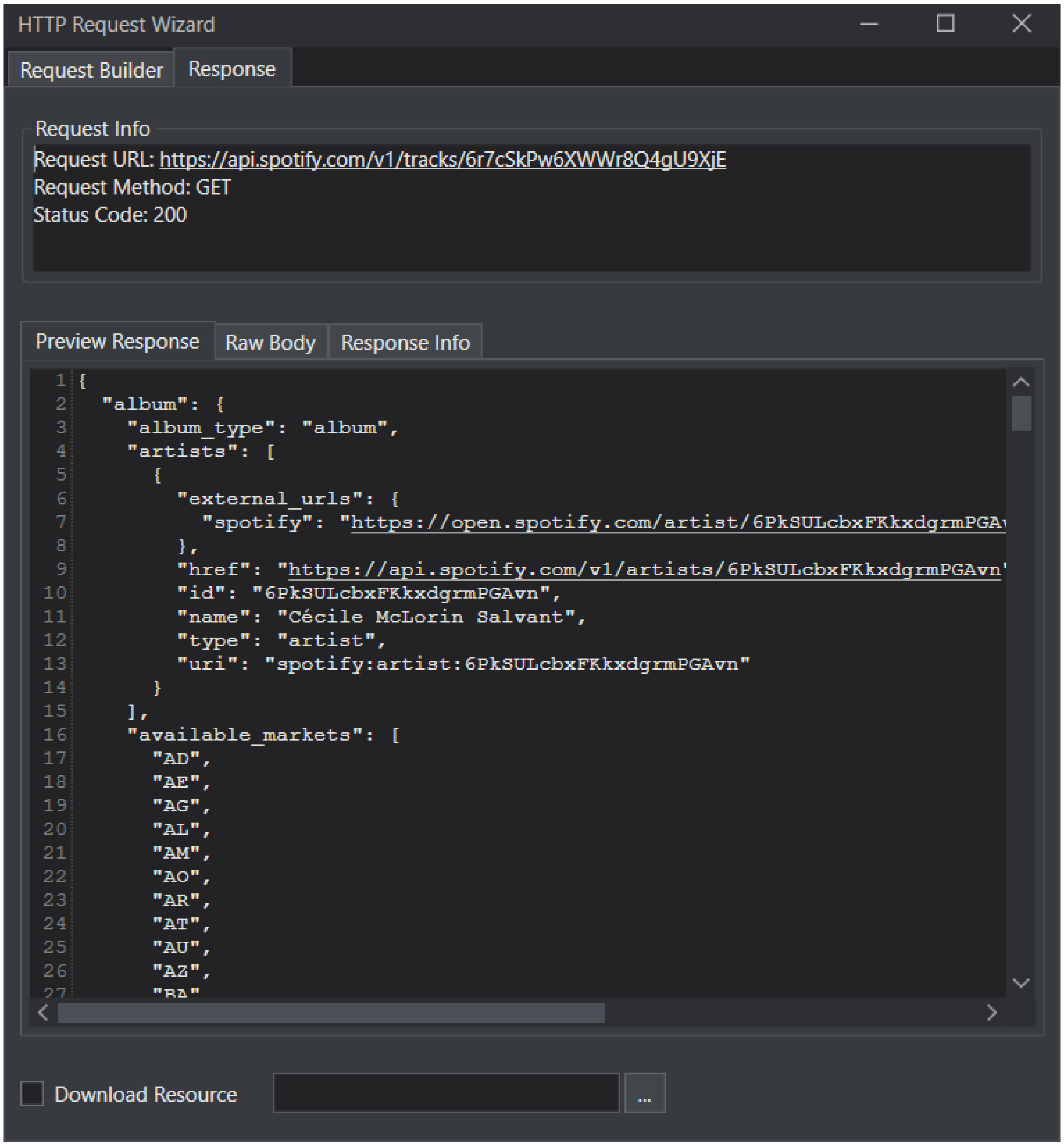
Request Method: GET

Accept response as: JSON

Authentication OAuth2

Access Token: [redacted]

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

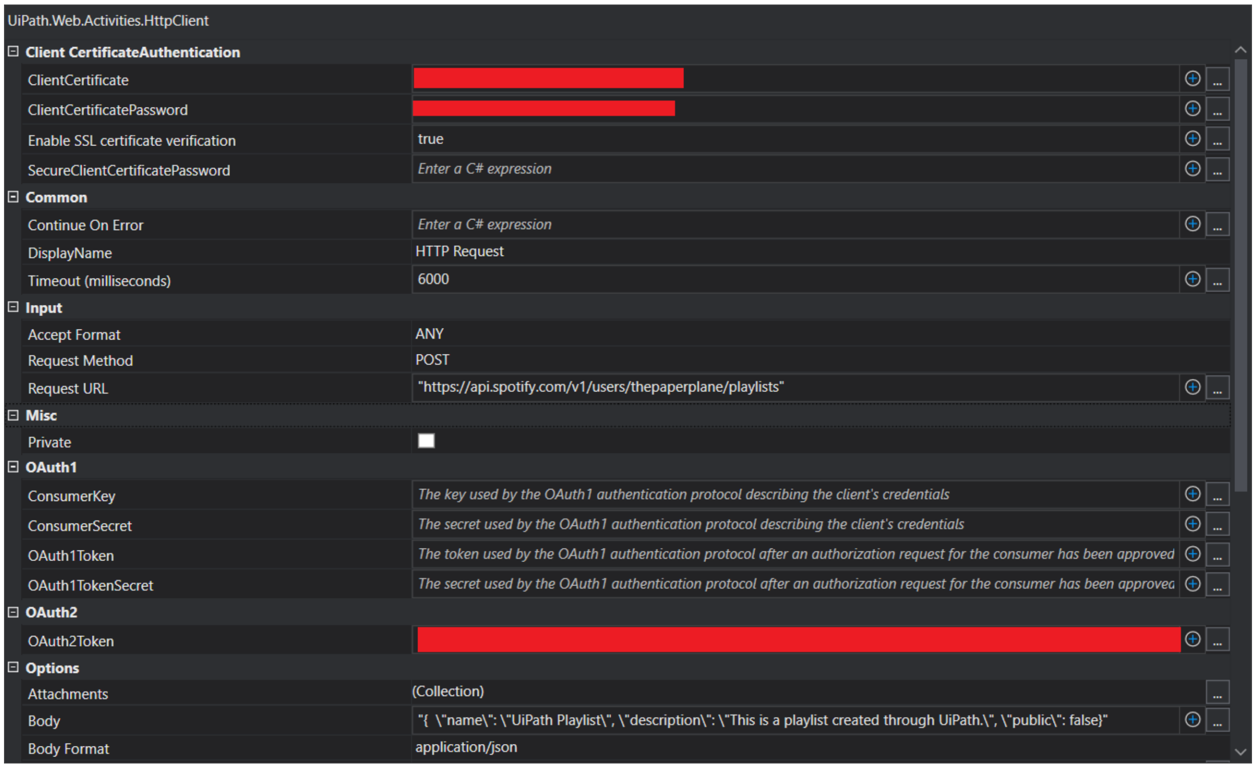


Request URL: https://api.spotify.com/v1/tracks/6r7cSkPw6XWWr8Q4gU9XjE
Request Method: GET
Status Code: 200

```
{
  "album": {
    "album_type": "album",
    "artists": [
      {
        "external_urls": {
          "spotify": "https://open.spotify.com/artist/6PkSULcbxFKkxdgrmPGA
        },
        "href": "https://api.spotify.com/v1/artists/6PkSULcbxFKkxdgrmPGAvn"
        "id": "6PkSULcbxFKkxdgrmPGAvn",
        "name": "Cécile McLorin Salvant",
        "type": "artist",
        "uri": "spotify:artist:6PkSULcbxFKkxdgrmPGAvn"
      }
    ],
    "available_markets": [
      "AD",
      "AE",
      "AG",
      "AL",
      "AM",
      "AO",
      "AR",
      "AT",
      "AU",
      "AZ",
      "BA"
```

GET request response preview.

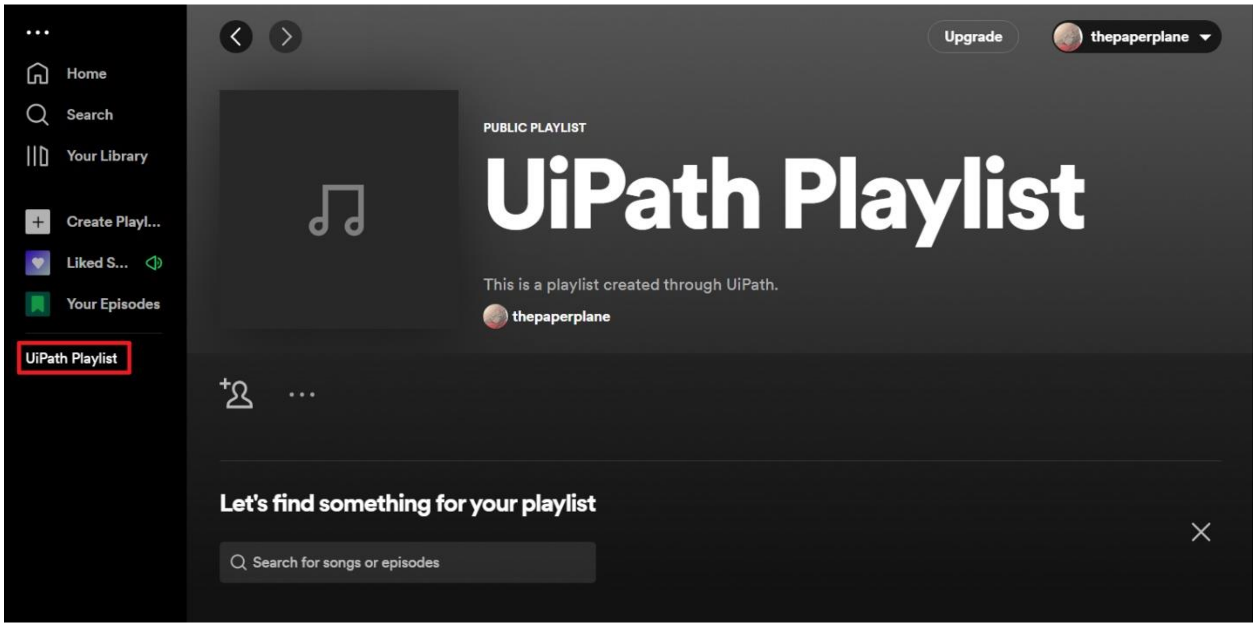
The following example uses the activity's **Properties** panel to create a new playlist called "UiPath Playlist" on an account called "thepaperplane". Note that the **Body** parameter takes in a string that is formatted like a JSON object.



UiPath.Web.Activities.HttpClient

Client CertificateAuthentication	
ClientCertificate	[redacted]
ClientCertificatePassword	[redacted]
Enable SSL certificate verification	true
SecureClientCertificatePassword	*Enter a C# expression*
Common	
Continue On Error	*Enter a C# expression*
DisplayName	HTTP Request
Timeout (milliseconds)	6000
Input	
Accept Format	ANY
Request Method	POST
Request URL	"https://api.spotify.com/v1/users/thepaperplane/playlists"
Misc	
Private	☐
OAuth1	
ConsumerKey	*The key used by the OAuth1 authentication protocol describing the client's credentials*
ConsumerSecret	*The secret used by the OAuth1 authentication protocol describing the client's credentials*
OAuth1Token	*The token used by the OAuth1 authentication protocol after an authorization request for the consumer has been approved*
OAuth1TokenSecret	*The secret used by the OAuth1 authentication protocol after an authorization request for the consumer has been approved*
OAuth2	
OAuth2Token	[redacted]
Options	
Attachments	(Collection)
Body	"{ \"name\": \"UiPath Playlist\", \"description\": \"This is a playlist created through UiPath.\", \"public\": false}"
Body Format	application/json

POST request to create a new playlist.

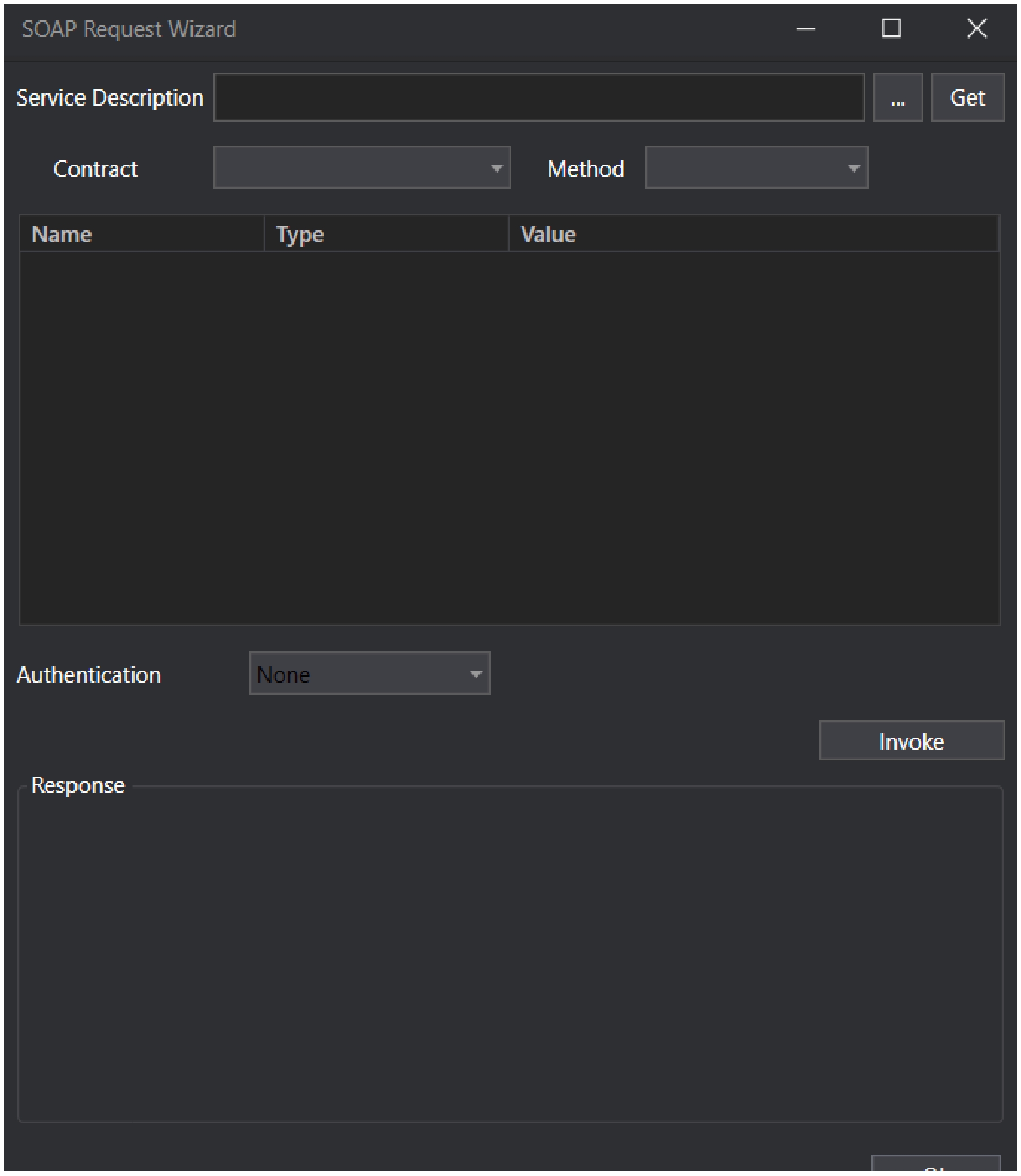


The request creates a new playlist called "UiPath Playlist" on thepaperplane's account.

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.

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.



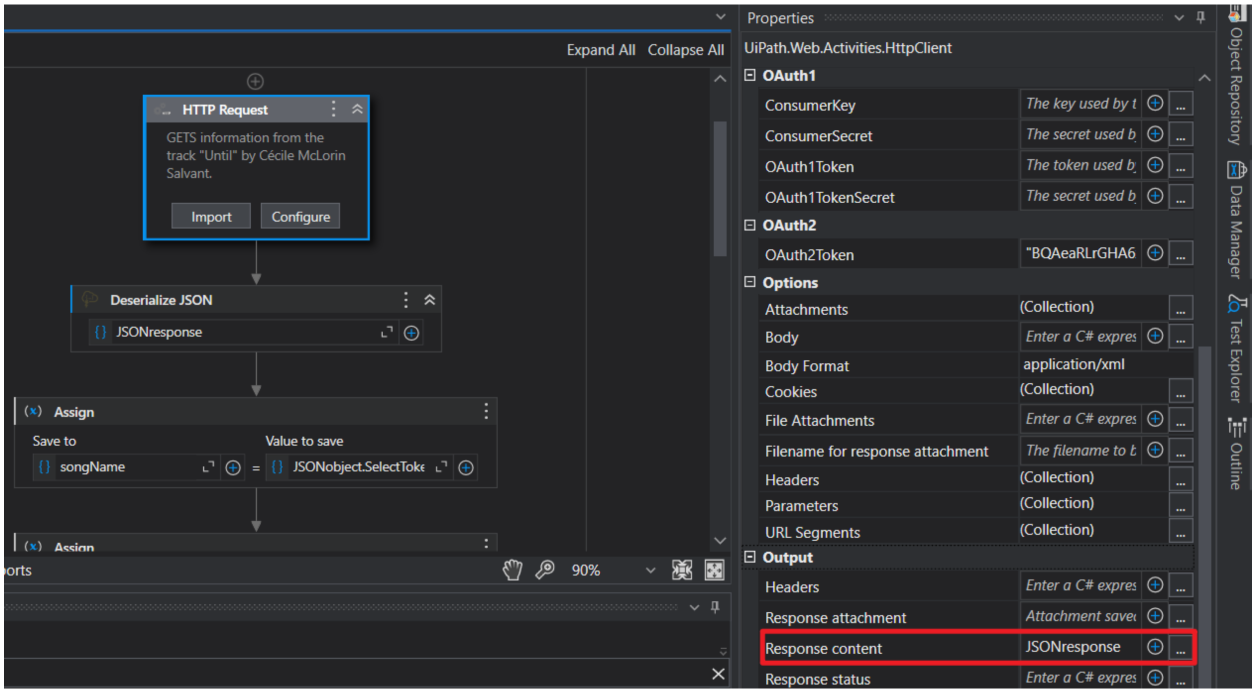
SOAP Request Wizard

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. In the **Properties** panel for the same **HTTP Request** activity as the Spotify GET example, create a new string variable in the **Response content** textbox.



GETS information from the track "Until" by Cécile McLorin Salvant.

JSONresponse

songName = JSONobject.SelectToke

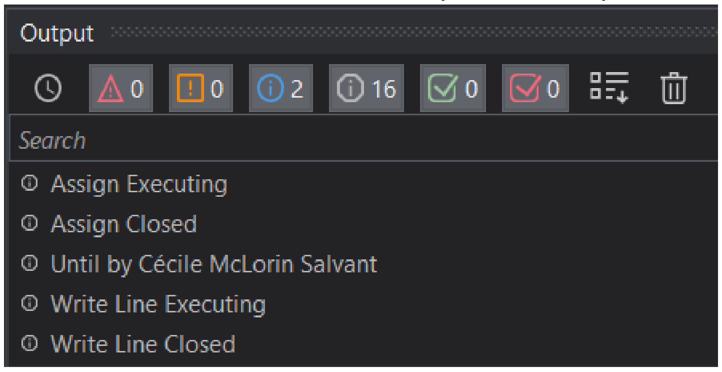
UiPath.Web.Activities.HttpClient

OAuth1	
ConsumerKey	The key used by t
ConsumerSecret	The secret used b
OAuth1Token	The token used b
OAuth1TokenSecret	The secret used b
OAuth2	
OAuth2Token	"BQAeaRLrGHA6
Options	
Attachments	(Collection)
Body	Enter a C# expres
Body Format	application/xml
Cookies	(Collection)
File Attachments	Enter a C# expres
Filename for response attachment	The filename to b
Headers	(Collection)
Parameters	(Collection)
URL Segments	(Collection)
Output	
Headers	Enter a C# expres
Response attachment	Attachment save
Response content	JSONresponse
Response status	Enter a C# expres

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."



```
Assign Executing
Assign Closed
Until by Cécile McLorin Salvant
Write Line Executing
Write Line Closed
```