

# Oracle® Cloud

## Rain or Shine, Oracle Integration Has You Covered



E96819-02  
February 2019



Oracle Cloud Rain or Shine, Oracle Integration Has You Covered,

E96819-02

Copyright © 2017, 2019, Oracle and/or its affiliates. All rights reserved.

Primary Author: Oracle Corporation

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

# Contents

<b>1</b>	<b>Module 1 Before You Begin</b>	
	About This Learning Path	1-1
	Get Access to Oracle Integration	1-3
<b>2</b>	<b>Module 2 Create Connections</b>	
	Create Connections: Create a Connection to the Integration	2-1
	Create a Connection to the API	2-3
<b>3</b>	<b>Module 3 Set Up an Integration</b>	
	Set Up Your Integration: Create an Integration	3-1
	Add Connections to the Integration	3-2
	Add the Trigger Connection	3-3
	Add the Invoke Connection	3-6
	Map the Data Between Connections	3-10
	Map Request Parameters Between Connections	3-10
	Map the Response Data Between Connections	3-11
	Add Business Identifier Fields for Tracking	3-13
	Activate the Integration	3-14
<b>4</b>	<b>Module 4 Create and Configure a Web Application with Visual Builder</b>	
	Build Your Visual Builder Application: Create the Application	4-1
	Create a Service Connection to the Integration	4-2
	Define the User Interface	4-3
	Configure the Action Chain to Invoke the Integration	4-8
	Run Your Application	4-14

# 1

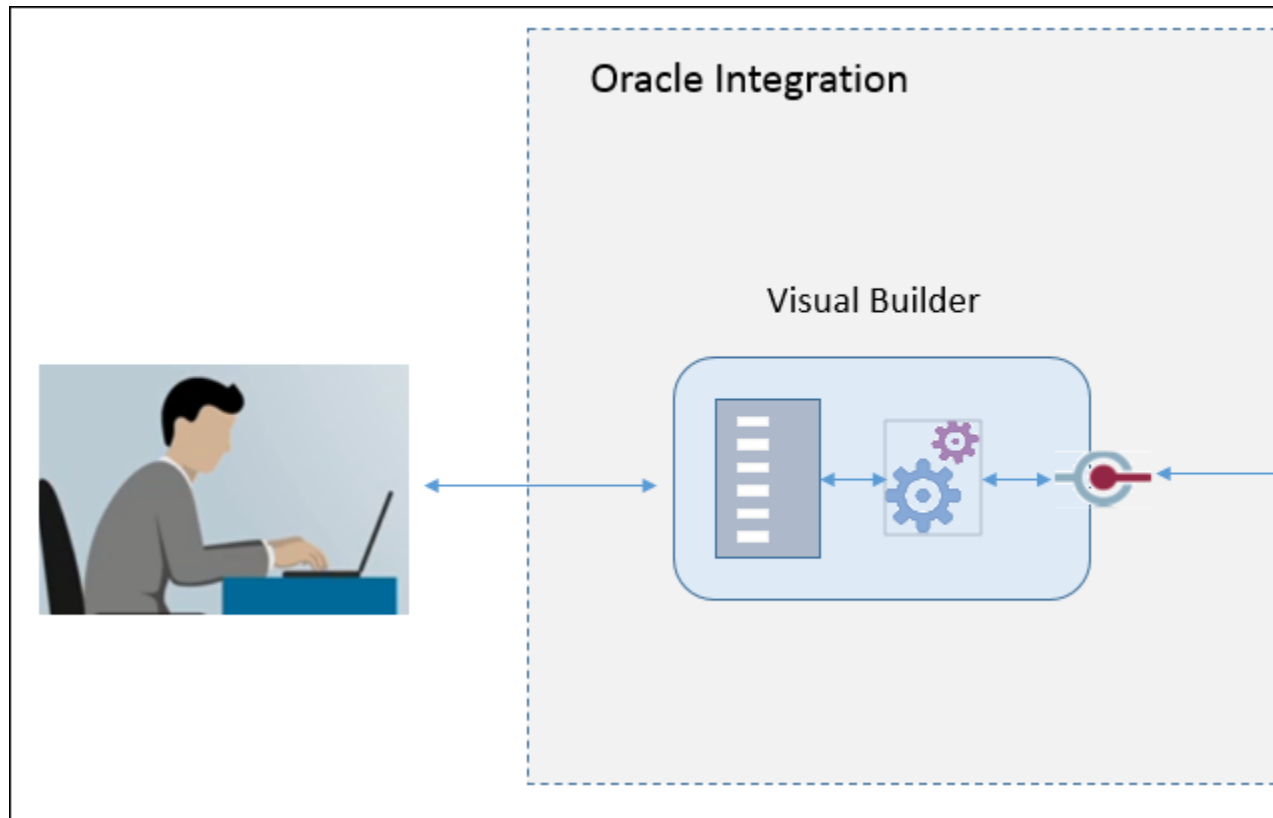
## Module 1 Before You Begin

### About This Learning Path

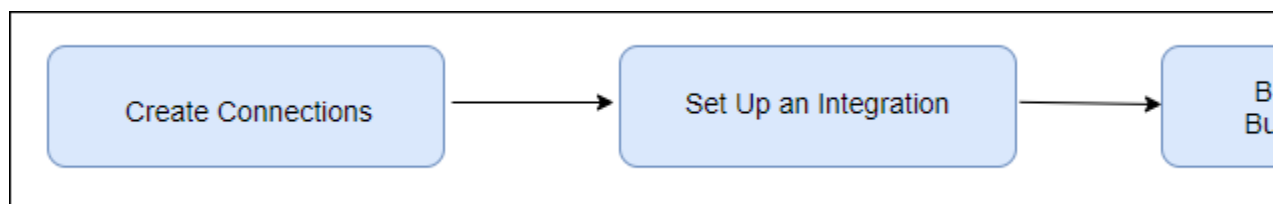
With Oracle Integration, you have the power to build applications and quickly design integrations that connect applications into a seamless unified solution. In this learning path, you'll build a solution to enable your users to view weather forecast for a given location. You want users to be able to enter the zip code or the name of a location and view details like what's the local time, temperature, wind speed, and humidity level in that location. Let's say you want to build something like this:

My Application		
Location *	<input type="text" value="90210"/>	<button>Get Weather Information</button>
Local Time	2018-06-28 23:44	
Temp F	64.0	
Temp C	17.8	
Weather Description	Clear	
Wind Speed Miles	0.0	
Wind Speed km	0.0	
Wind Direction	N	
Humidity	81	
Precipitation MM	0.0	

You'll use the Integrations and Visual Builder features of Oracle Integration to retrieve weather forecast from a REST weather API named APIXU. To complete the learning path, you'll need access to Oracle Integration. You'll also need to sign up with the APIXU weather API to obtain the API key.



You'll follow these three main steps to build the solution: create connections, set up an integration, and build your Visual Builder application.



You begin by creating connections. You create an integration connection and then a connection to APIXU. After the connections are in place, you create and configure your integration. You need to use the integration connection as the trigger and configure the endpoint by specifying request query parameters and the response data that the integration will send to the Visual Builder application. Next, you use the weather API connection as the invoke and configure the endpoint by specifying the request query parameters and the response data that the weather API returns to your integration.

Applications generally use different data structures. To enable information exchange, you need to map their parameters. In your integration, you define the mappings that transfer data between the trigger and invoke for each request and response. You need to map the request query parameters. You also need to map the response parameters. Then, define business identifiers for tracking payload fields in messages during runtime. After you activate your integration, it's ready to be invoked.

You can then create the front-end for the integration – your Visual Builder application. You configure your application to create a connection to the integration. Next, you design your application's user interface. You define the fields that display weather details, define variables corresponding to each field, and map the fields with the variables. When your application's user interface is ready, you need to define the event that will invoke the integration. Let's say that the integration is invoked when the user clicks the Get Weather Information button. In the action chain for the button, you need to map the location parameters of the integration and your application and specify the API key. Then, you map the integration's response parameters with the variables defined in the application.

When your application is ready, run the application, enter a location of your choice, and click the Get Weather Information button. This makes your application to invoke the integration. The integration then connects to the weather API, retrieves the weather forecast, and displays it to you. That's all there is to it. Let's begin!

## Get Access to Oracle Integration

You'll use Oracle Integration to perform the tasks in this learning path. It would be helpful if you have a working knowledge of the user interface of Oracle Integration.

To complete the learning path, you need the following:

- Access to Oracle Integration. You need your user name, password, data center/region, and identity domain to sign in to your instance.
- ServiceDeveloper and ServiceDeployer roles for the Integrations feature of Oracle Integration
- ServiceDeveloper role for the Visual Builder feature of Oracle Integration

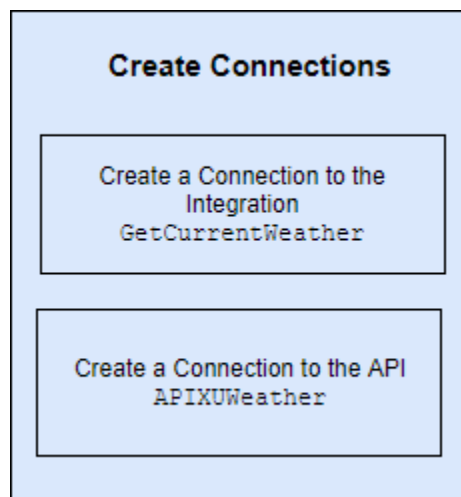
If you don't have an instance of Oracle Integration up and running, use free credits to try Oracle Integration. See [Sign Up for Free Oracle Cloud Promotion](#).

# 2

## Module 2 Create Connections

### Create Connections: Create a Connection to the Integration

In this module, you'll create connections to the applications with which you want to share data. You'll create the integration connection named `GetCurrentWeather` and the API connection named `APIXUWeather`.




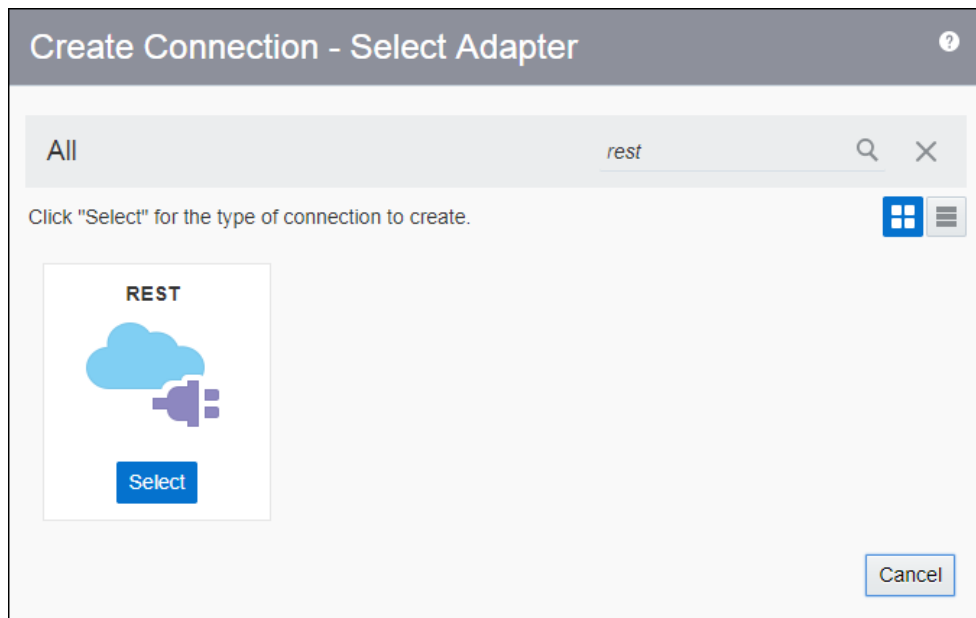
Let's begin by creating the `GetCurrentWeather` connection that will be used as the trigger to invoke the integration. Use the REST adapter to create your connection.

1. Sign in to Oracle Integration.

Use the service URL and credentials given to you either in an email or by your administrator.

2. In the Integration navigation pane, click **Integrations**. (If the Guide Me overlay appears, then click **Got It** to continue.)
3. In the Designer navigation pane, click **Connections**. (If the Guide Me overlay appears, then click **Got It** to continue.)
4. Click **Create**.
5. Select the REST adapter.

In the Create Connection - Select Adapter dialog, type `rest` in the **Search** field, then click **Search** . This displays the REST adapter. Click **Select** for the REST adapter.



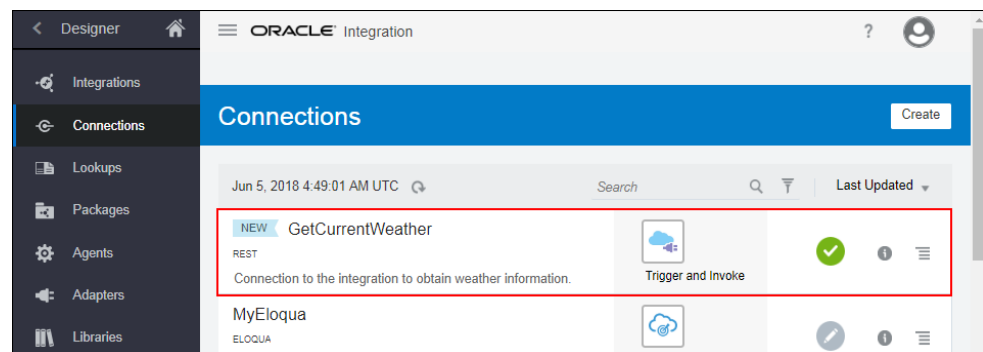
6. In the Create New Connection dialog, enter information about the connection.
  - a. In the **Name** field, enter `GetCurrentWeather`. Notice that the name is automatically added in capital letters to the **Identifier** field.
  - b. From the **Role** drop-down list, select **Trigger and Invoke**.  
 When you select a role, only the connection properties and security policies appropriate to that role can be configured. With the role as Trigger and Invoke, you will be able to use your connection either as an invoke connection or a trigger connection in the integration.
  - c. In the **Description** field, enter `Connection to the integration to obtain weather information`.
  - d. Click **Create**.  
 The details page for the `GetCurrentWeather` connection is displayed. You're now ready to configure connection details.
7. In the **Connection Administrator** section, enter your email address to receive email notifications related to the connection.
8. Configure connection properties.
  - a. Click **Configure Connectivity**.  
 The Connection Properties dialog is displayed.
  - b. From the **Connection Type** drop-down list, select **REST API Base URL**.
  - c. In the **Connection URL** field, enter the URL to access your Oracle Integration instance. For example `https://myicinstance.com`. You can copy the URL from the address bar in your browser where you've opened the instance.
  - d. Click **OK**.
9. Configure connection security.
  - a. Click **Configure Security**.  
 The Credentials dialog is displayed.

- b. From the **Security Policy** drop-down list, select **Basic Authentication**.
  - c. In the **Username**, **Password**, **Confirm Password** fields, specify the sign-in credentials for accessing Oracle Integration.
  - d. Click **OK**.
10. Click **Test** to check the connection.

If the connection is working fine, then the following message is displayed:  
Connection GetCurrentWeather was tested successfully.

11. Click **Save**.
12. Click **Close**.

The GetCurrentWeather connection is now listed on the Connections page. You will use this connection as the trigger connection in your integration.



## Create a Connection to the API

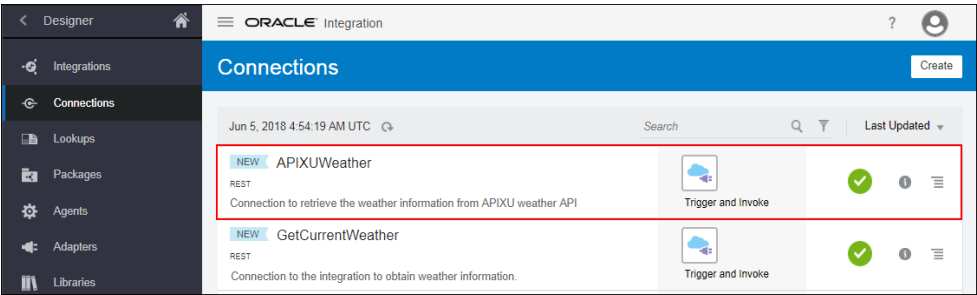
Similarly, now create a connection to the APIXU weather API. This connection will be used to invoke the weather API to get the weather information.

1. On the **Connections** page, click **Create**.
2. Select the REST adapter.  
In the Create Connection - Select Adapter dialog, for the REST adapter, click **Select**. If the REST adapter is not listed, type `rest` in the **Search** field, then click **Search**.
3. In the Create New Connection dialog, enter information about the connection.
  - a. In the **Name** field, enter `APIXUWeather`. Notice that the name is automatically added in capital letters to the **Identifier** field.
  - b. In the **Role** field, retain the default selection **Trigger and Invoke**.  
With the role as Trigger and Invoke, you will be able to use your connection either as an invoke connection or a trigger connection in the integration.
  - c. In the **Description** field, enter `Connection to retrieve the weather information from APIXU weather API`.
  - d. Click **Create**.

The details page for your APIXUWeather connection is displayed. You're now ready to configure the connection details.

4. In the **Connection Administrator** section, enter your email address to receive email notifications related to the connection.
5. Configure connection properties.
  - a. Click **Configure Connectivity**.  
The Connection Properties dialog is displayed.
  - b. From the **Connection Type** drop-down list, select **REST API Base URL**.
  - c. In the **Connection URL** field, enter the URL of the API: `http://api.apixu.com/v1`.
  - d. Click **OK**.
6. Configure connection security.
  - a. Click **Configure Security**.  
The Credentials dialog is displayed.
  - b. From the **Security Policy** drop-down list, select **No Security Policy**. The API manages the security for the connection.
  - c. Click **OK**.
7. Click **Test** to check the connection.  
If the connection is working fine, then the following message is displayed:  
Connection APIXUWeather was tested successfully.
8. Click **Save**.
9. Click **Close**.

The APIXUWeather connection is now listed on the Connections page. You will use this connection as the invoke connection in your integration.

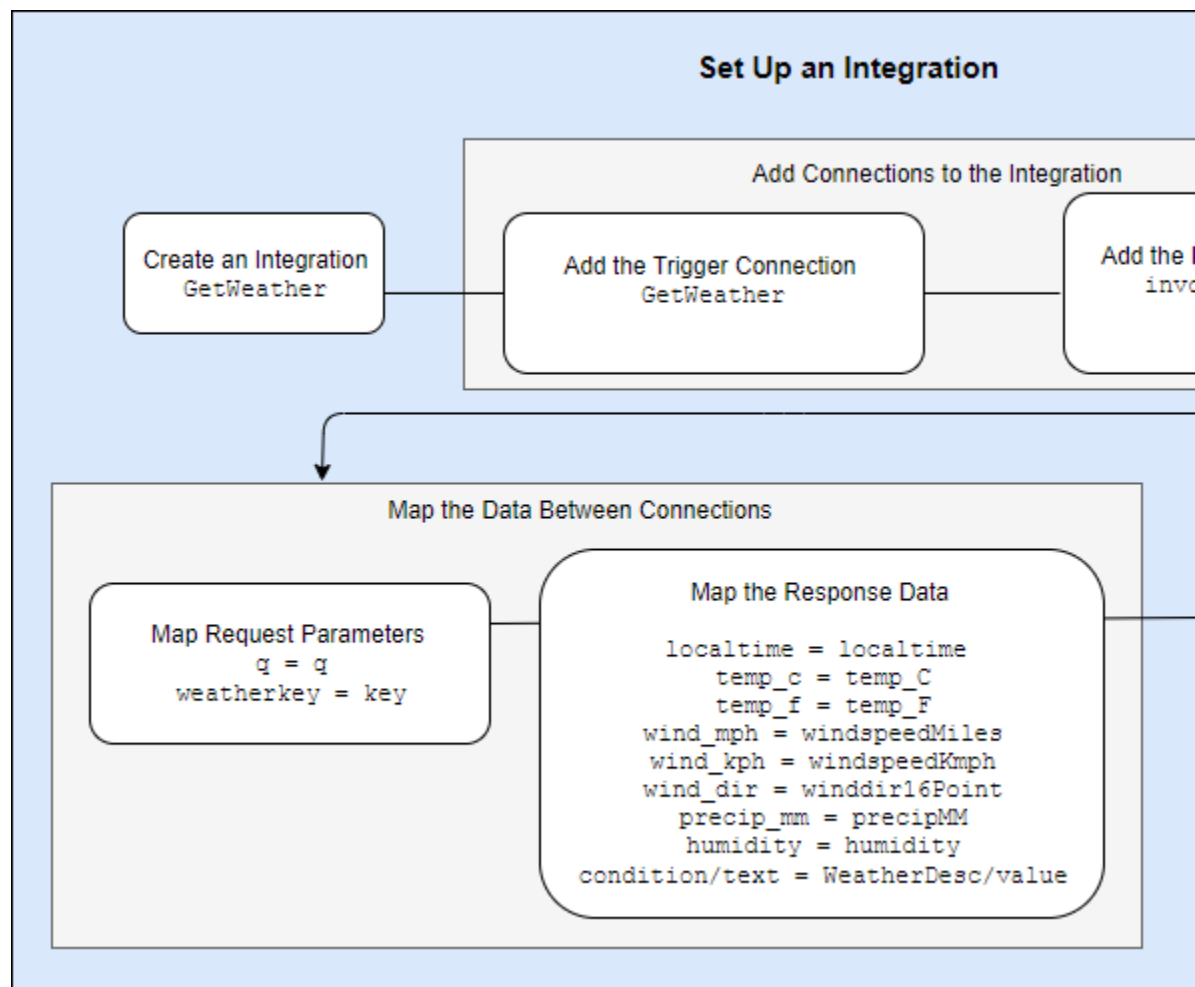


# 3

## Module 3 Set Up an Integration

### Set Up Your Integration: Create an Integration

In this module, you'll create and configure an integration named `GetWeather`. You'll add the trigger connection (`GetWeather`) and the invoke connection (`invokeWeather`) to the integration. Then, you'll map the request and response parameters between the connections, add business identifiers, and activate your integration.




Let's first create the integration.

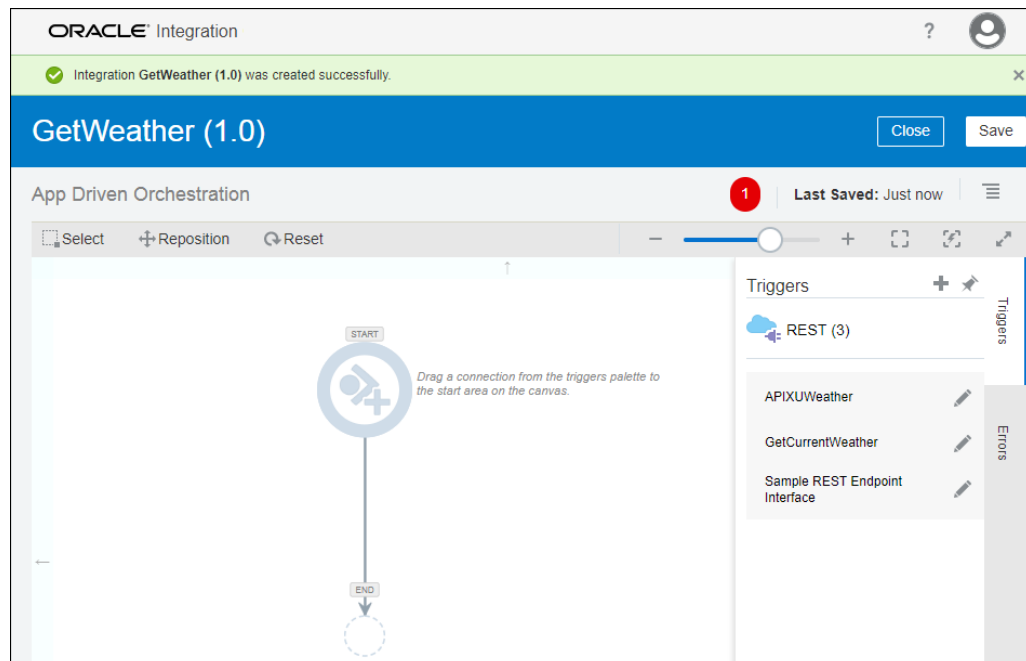
1. In the Designer navigation pane, click **Integrations**. (If the Guide Me overlay appears, then click **Got It** to continue.)

- On the Integrations page, click **Create**.  
The Create Integration - Select a Style dialog is displayed.
- Click **Select** for the **App Driven Orchestration** integration style.
- Enter the following information in the Create New Integration dialog:

Field	Enter
What do you want to call your integration?	GetWeather
Identifier	GETWEATHER (This value is entered for you automatically when you specify the integration name.)
Version	(Leave as is. Use the default value of 01.00.0000.)
What does this integration do?	Retrieve weather information from the APIXU weather API
Which package does this integration belong to?	com.oracle.weather (Packages enable you to group integrations into a single structure that makes them easy to import and export.)

- Click **Create**.

The integration **GetWeather** is created successfully, and the integration canvas page is displayed. The **Errors palette** icon  indicates that the configuration of the integration is not yet complete.



## Add Connections to the Integration

Next, let's add connections, which you created earlier, to the integration. In this learning path, the integration will be invoked when your Visual Builder application requests for weather information. The integration connection will send request to the

weather API connection to obtain weather forecasts. Let's first add the integration connection as the trigger, or source, connection. Then, add the weather API connection as the target, or invoke, connection.

- [Add the Trigger Connection](#)
- [Add the Invoke Connection](#)

## Add the Trigger Connection

When your Visual Builder application will invoke the integration, the integration will request the weather API for weather information by using two request query parameters, `q` and `weatherkey`. Also, the response payload that the integration will return to Visual Builder will comprise JSON input parameters. Let's use these details to add the integration connection as the trigger connection to the integration.


1. On the GetWeather integration canvas, in the **Triggers** pane, click **REST** if the REST adapters are not listed already.
2. Drag the **GetCurrentWeather** integration connection to the plus icon below **START** on the canvas.

This displays the Configure REST Endpoint configuration wizard.

3. On the Basic Info page of the wizard, specify the following information:

Field	Enter/Select
What do you want to call your endpoint?	GetWeather
What is the endpoint's relative resource URI?	/weather
What action do you want to perform on the endpoint?	GET
Add and review parameters for this endpoint	Select the check box
Configure this endpoint to receive the response	Select the check box

4. Click **Next**.
5. On the Request Parameters page, add the request query parameters for the endpoint of your integration.

Click **Add**  to add a parameter. The following table lists the parameters and their data type that you need to add for the trigger endpoint.

Name	Data Type
<code>q</code> (Represents the location, whether it is a zip code, city, or state, for which you want to retrieve weather information.)	String
<code>weatherkey</code> (Represents the security key issued by the weather API to enable you to invoke the API. You will define this parameter again when you create your Visual Builder application.)	String

6. Click **Next**.
7. On the Response page, select the type of payload the endpoint will return.
  - a. In the **Select the type of payload with which you want the endpoint to reply** section, select **JSON**. When you select JSON, response is returned based on the specified JSON input parameters.
  - b. From the drop-down list in the **Select the response payload format** section, select **JSON Sample** as the payload format.
  - c. Click the **<<<inline>>>** link to specify JSON payload input parameters that the integration will return as a response to your Visual Builder application.

The screenshot shows the 'Configure REST Endpoint' dialog box with the 'Response' tab selected. The 'Response' tab contains the following sections:

- Select the attachment processing options:**
  - ☐ Accept attachments from response
  - ☐ Response is HTML form
- Select the response payload format:**
  - JSON Sample (selected in the dropdown)
- Sample Location:**
  - Choose File | No file chosen
  - ..OR.. enter sample JSON <<< inline >>>
- Select the type of payload with which you want the endpoint to reply:**
  - ☐ XML
  - ☐ XML(text)
  - ☒ JSON
  - ☐ Other Media Type
- Media Type:**
  - For example, application/oracle.cloud+json, applic

- d. Copy the following sample entries and paste them in the **Enter Sample JSON** area in the Response Sample Json Payload page:

```
{ "localtime": "2018-05-01 17:01", "temp_C": "31", "temp_F": "88", "WeatherDesc":
{ "value": "Partly Cloudy" }, "windspeedMiles": "12", "windspeedKmph": "19",
"winddir16Point": "WNW", "humidity": "46", "percipMM": "0.0" }
```

**Configure REST Endpoint**

Response Sample Json Payload  
Enter the sample JSON that describes the HTTP message payload.

Enter Sample JSON

```
{
  "localtime": "2018-05-01 17:01",
  "temp_C": "31",
  "temp_F": "88",
  "WeatherDesc": {
    "value": "Partly Cloudy"
  },
  "windspeedMiles": "12",
  "windspeedKmph": "19",
  "winddir16Point": "WNW",
  "humidity": "46",
  "percipMM": "0.0"
}
```

Cancel OK

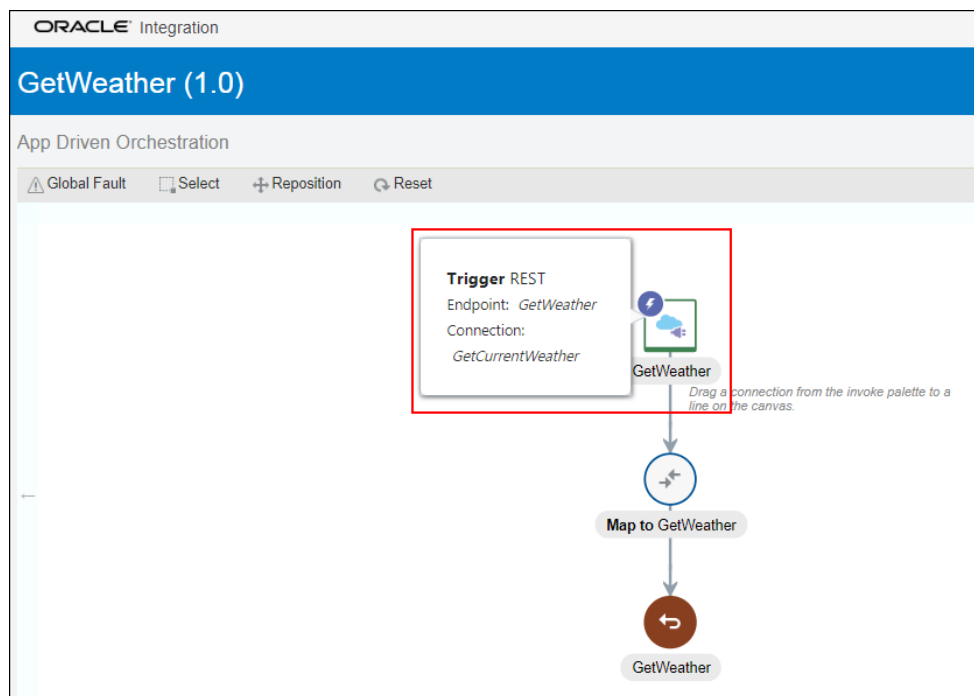
e. Click **Ok**.

If the JSON entries are valid, the wizard shows the Response page.

8. Click **Next**.

9. On the Summary page, verify the endpoint details, including the REST Service URI, query parameters, the method, and the response media type. Click **Done**.

The **GetWeather** trigger endpoint, configured to use the **GetCurrentWeather** trigger connection, is displayed on the canvas. Also, a mapper named **Map to GetWeather** is automatically created and displayed on the canvas.

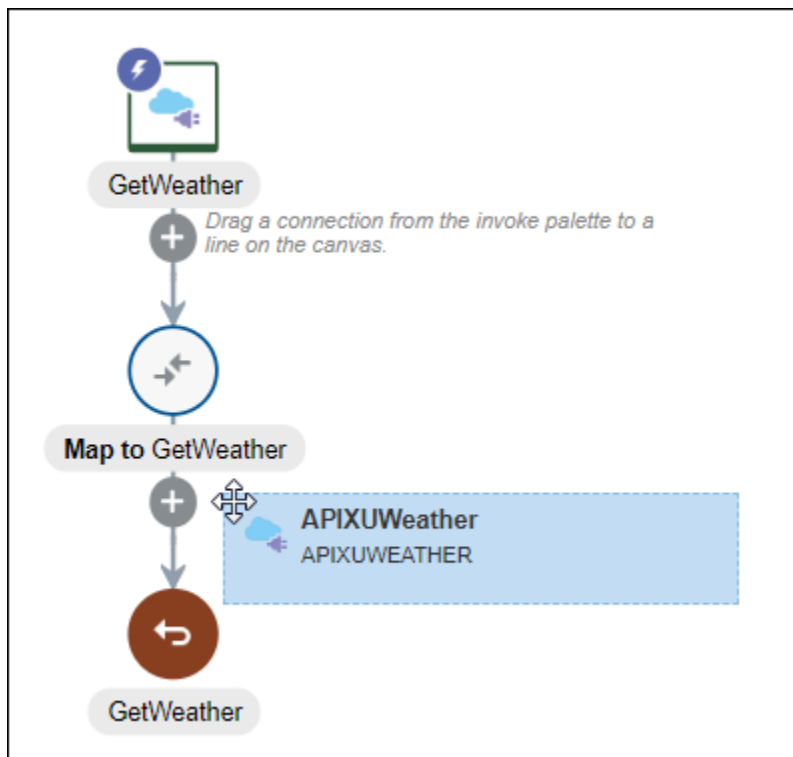


10. Click **Save**.

## Add the Invoke Connection

Similarly, add an invoke connection named `APIXUWeather` to the integration and configure it to define query parameters and response type. In your integration, the `GetWeather` trigger connection you just added will send requests to the `APIXUWeather` invoke connection to get information from the weather API. Keep your API key ready.

1. On the **GetWeather** integration canvas page, click the **Invokes** tab.
2. Click **REST** to list all the REST adapters, if not already listed.
3. To make invocation to the weather API, drag the **APIXUWeather** connection to the canvas and drop it on the plus icon below the mapper **Map to GetWeather**.




This displays the Configure REST Endpoint configuration wizard.

4. On the Basic Info page of the wizard, specify the following information:

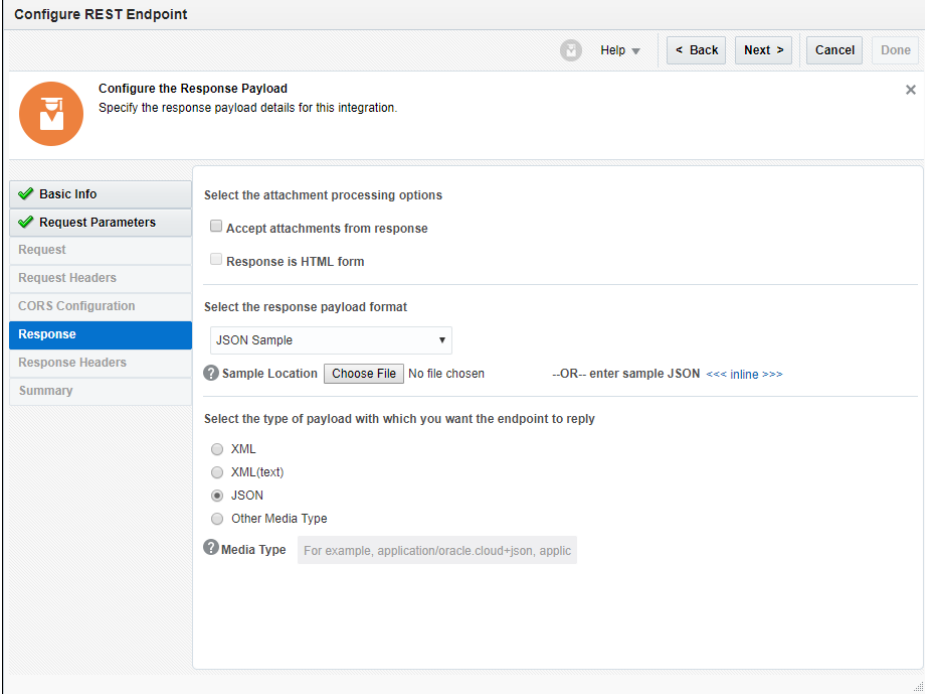
Field	Enter/Select
What do you want to call your endpoint?	<code>invokeWeather</code>
What is the endpoint's relative resource URI?	<code>/current.json</code> (You obtain this URI from the APIXU weather API.)
What action do you want to perform on the endpoint?	GET
Add and review parameters for this endpoint	Select the check box
Configure this endpoint to receive the response	Select the check box

5. Click **Next**.
6. On the Request Parameters page, add the request query parameters for the endpoint.

Click **Add**  to add a parameter. The following table lists the parameters and their data type that you need to add for the endpoint. The parameter names are defined by the weather API, and you shouldn't modify their definition.

Name	Data Type
q (Represents the location, whether it is a zip code, city, or state, for which you want to retrieve weather information.)	String
key (Represents the security key issued by the weather API to enable you to invoke the API.)	String

7. Click **Next**.
8. On the Response page, select the type of payload the API endpoint will return.
  - a. In the **Select the type of payload with which you want the endpoint to reply** section, select **JSON**.



**Configure REST Endpoint**

Configure the Response Payload  
Specify the response payload details for this integration.

**Basic Info**

**Request Parameters**

**Response**

**Select the attachment processing options**

☐ Accept attachments from response

☐ Response is HTML form

**Select the response payload format**

JSON Sample

**Sample Location**  No file chosen --OR-- enter sample JSON <<< inline >>>

**Select the type of payload with which you want the endpoint to reply**

☐ XML

☐ XML(text)

☒ JSON

☐ Other Media Type

**Media Type** For example, application/oracle.cloud+json, applic

- b. From the drop-down list in the **Select the response payload format** section, select **JSON Sample** as the payload format.
- c. Click the <<<inline>>> link to specify JSON payload input parameters that you want the API to return as a response to the integration.
- d. To obtain the JSON payload, open another tab in your browser and enter the API URL in the following format:

`http://api.apixu.com/v1/current.json?key=application_key&q=location`

Where:

*application\_key* refers to the key that you obtained from the weather API, and *location* refers to the location for which you want to retrieve weather information. For example:

`http://api.apixu.com/v1/current.json?key=492798775zj456794&q=34772`

This displays the JSON payload, including the fields and types of value.

- e. Copy the JSON message payload.
- f. Return to the Configure REST Endpoint configuration wizard. On the Response Sample Json Payload page, paste the JSON message payload in the **Enter Sample JSON** area and click **Ok**.

**Configure REST Endpoint**

Response Sample Json Payload  
Enter the sample JSON that describes the HTTP message payload.

Enter Sample JSON

```
{
  "location": {
    "name": "Round Rock",
    "region": "Texas",
    "country": "USA",
    "lat": 30.51,
    "lon": -97.71,
    "tz_id": "America/Chicago",
    "localtime_epoch": 1528951564,
    "localtime": "2018-06-13 23:46",
    "current": {
      "last_updated_epoch": 1528951512,
      "last_updated": "2018-06-13 23:45",
      "temp_c": 28.0,
      "temp_f": 82.4,
      "is_day": 0,
      "condition": {
        "text": "Clear",
        "icon": "http://cdn.apixu.com/weather/64x64/night/113.png",
        "code": 1000,
        "wind_mph": 9.4,
        "wind_kph": 15.1,
        "wind_degree": 170,
        "wind_dir": "S",
        "pressure_mb": 1017.0,
        "pressure_in": 30.5,
        "precip_mm": 0.0,
        "precip_in": 0.0,
        "humidity": 58,
        "cloud": 0,
        "feelslike_c": 29.5,
        "feelslike_f": 85.1,
        "vis_km": 16.0,
        "vis_miles": 9.0
      }
    }
  }
}
```

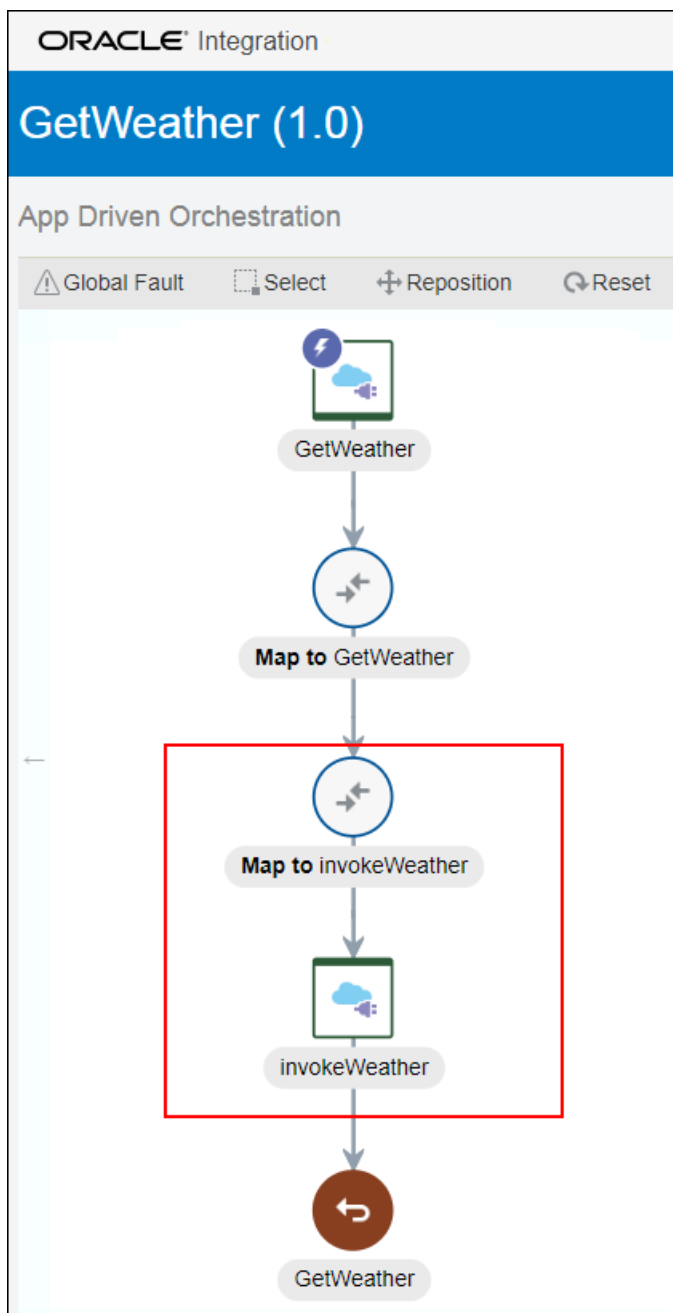
Cancel Ok


If the JSON entries are valid, the wizard shows the Response page.

9. Click **Next**.
10. On the Summary page, verify the REST Service URI, the method, query parameters, and the response media type. Click **Done**.

The endpoint **invokeWeather**, configured to use the **APIXUWeather** invoke connection, appears on the canvas. Also an invoke mapper named **Map to invokeWeather** is automatically created and displayed on the canvas.

Your integration now has two mappers, **Map to invokeWeather** and **Map to GetWeather**. We don't need two mappers, so let's delete **Map to GetWeather**.



11. Delete the **Map to GetWeather** map action that was automatically generated for the trigger endpoint.
  - a. Click **Map to GetWeather**.
  - b. Click **More Actions** , then click **Delete**.
  - c. Click **Delete** again to confirm.
12. Click **Save**.

## Map the Data Between Connections


Applications generally use different data structures. To enable information exchange, you need to map their parameters. Let's map the request parameters and then the response parameters between the trigger and invoke connections.

- [Map Request Parameters Between Connections](#)
- [Map the Response Data Between Connections](#)

### Map Request Parameters Between Connections

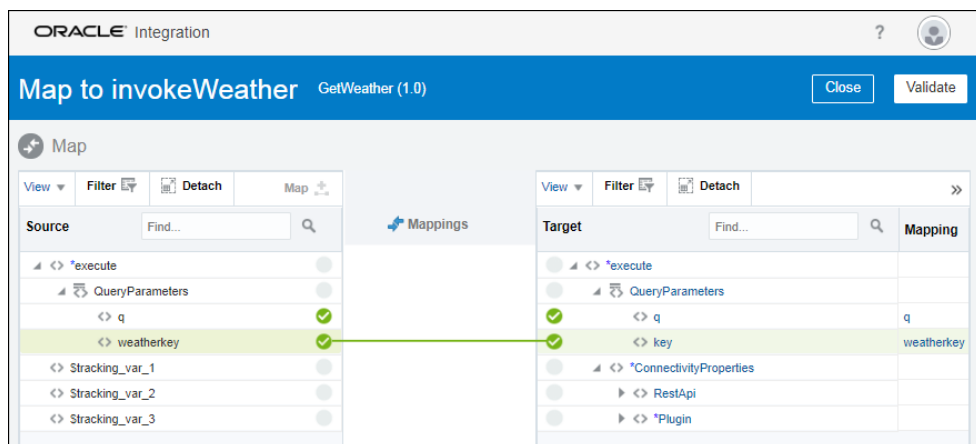
Map the parameters that you defined in the integration (trigger connection) with the parameters required by the API XU weather API (invoke connection).

The visual mapper enables you to map parameters between applications by dragging the trigger parameters onto the invoke parameters. The maps you create are called transformation maps.

1. On the **GetWeather** integration canvas, click **Map to invokeWeather**, then click **Edit** .

In the mapper, data structures are automatically populated with information pulled from the trigger and invoke connections. The **Source** pane displays parameters you defined in the integration connection (`GetCurrentWeather`), and the **Target** pane displays parameters that you defined in the weather API connection (`APIXUWeather`).

2. In the **Source** pane, under **QueryParameters**, drag the **q** parameter onto the **q** parameter in the **Target** pane. A green check mark icon appears next to both the parameters, indicating they are mapped.
3. Similarly, from the **Source** pane, drag the **weatherkey** parameter onto the **key** parameter in the **Target** pane. The green connecting line is displayed between the most recently mapped parameters.



4. Click **Validate** to verify that the mappings are valid.

If the mapping is valid, you'll get the following message: Mapping is valid and ready to use.

5. Click **Close**.

## Map the Response Data Between Connections

The APIXU weather API can return a whole lot of parameters. In our integration, we want only a subset of the available parameters returned by the API. Let's add a transformation to map the parameters that we want in our integration to the parameters returned by the weather API.

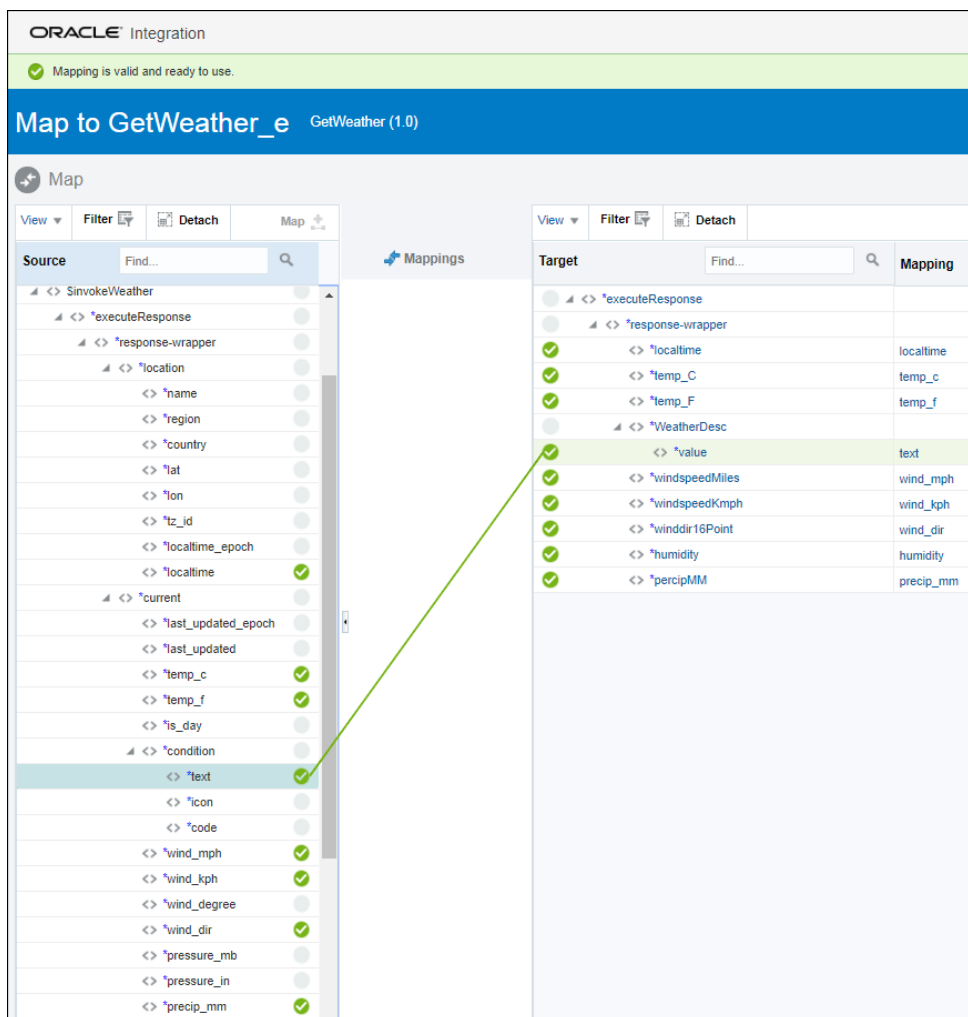
1. On the **GetWeather** integration canvas, click the **Actions** tab.
2. In the **Actions** pane, from the **Data** section, drag the **Map** action and drop it on the plus icon below the **invokeWeather** object on the canvas.

The Map to GetWeather dialog is displayed. The **Source** pane displays all the parameters that the weather API can return. The **Target** pane shows all the parameters that you want the API to return as a response when your integration is invoked. You'll now map the source parameters to the target parameters.

3. In the **Source** pane, under **\$invokeWeather**, expand **executeResponse**, then **response-wrapper**, then **location**.
4. In the **Target** pane, under **executeResponse**, expand **response-wrapper**, if not already expanded.
5. From the **Source** pane, drag the **localtime** parameter onto the **localtime** parameter in the **Target** pane.
6. In the **Source** pane, under **response-wrapper**, expand **current**. Map the source parameters to the target parameters as listed in the following table:

Source Parameter (Weather API)	Target Parameter (Integrations)
temp_c	temp_C
temp_f	temp_F
wind_mph	windspeedMiles
wind_kph	windspeedKmph
wind_dir	winddir16Point
precip_mm	precipMM
humidity	humidity

7. In the **Source** pane, under **current**, expand **condition**.
8. In the **Target** pane, expand **WeatherDesc**.
9. From the **Source** pane, drag the **text** parameter onto the **value** parameter under **WeatherDesc** in the **Target** pane.
10. Click **Validate** to verify that the parameters are mapped correctly. If the mapping is valid, the following message is displayed: Mapping is valid and ready to use.



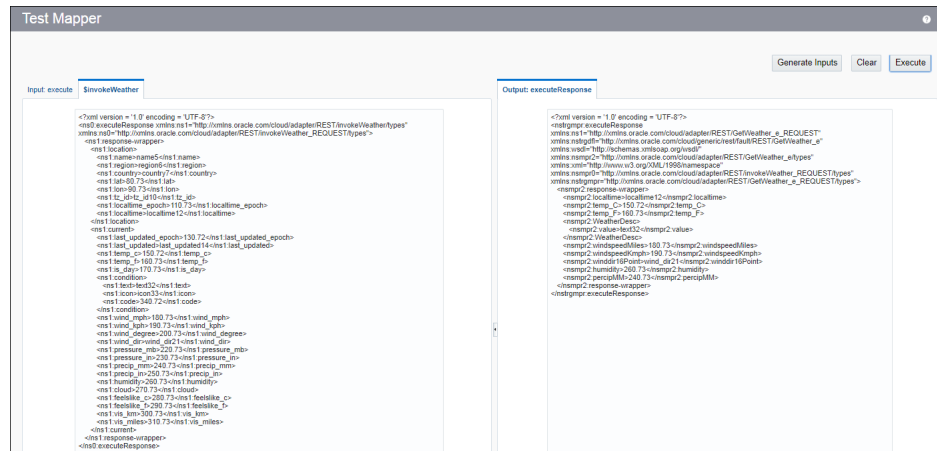
11. Test your mapper.

- a. In the **Target** pane, click **Test**.
- b. In the Test Mapper dialog, click the **invokeWeather** tab, then click **Generate Inputs** to generate the payload.

The **invokeWeather** tab displays all the input parameters that the API can return.

- c. Click **Execute**.


The **Output: executeResponse** tab shows only those parameters that the API will return when your integration is invoked.

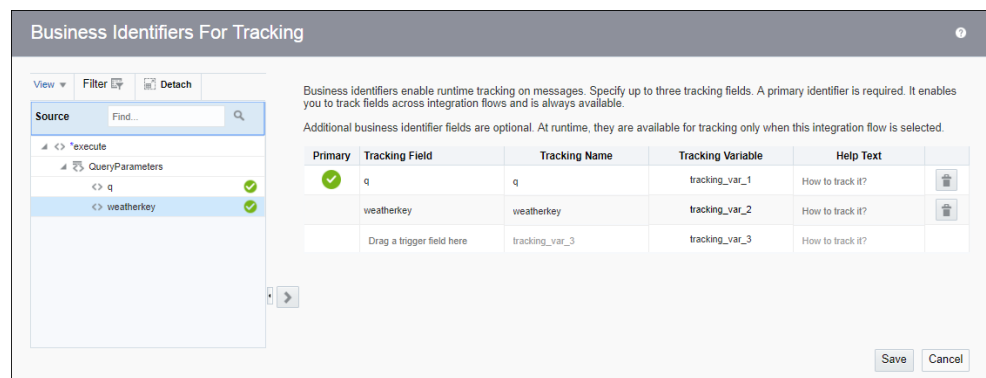


- d. Click **Close** to close the Test Mapper dialog.
12. Click **Close** to close the Map to GetWeather dialog.
13. Click **Save** to save the integration.


## Add Business Identifier Fields for Tracking

You can track payload fields during runtime by using business identifiers. Tracking helps with troubleshooting your integration flows. You can specify up to three business identifier fields for tracking, and select one of the fields as the primary business identifier. Let's add identifiers to track fields across integration flows.

1. On the integration canvas page, click **Actions** , then select **Tracking**.  
The Business Identifiers For Tracking dialog is displayed.
2. From the **Source** pane, drag the **q** parameter to the **Tracking Field** column in the first row.  
The **q** parameters is marked as the Primary parameter.
3. Drag the **weatherkey** parameter to the **Tracking Field** column in the second row. It is optional to add a second parameter.
4. Click **Save**.




5. On the integration canvas page, click **Save** to save your integration.

Notice that the **Errors palette** icon  has disappeared, indicating that your integration is now complete.

6. Click **Close**.


## Activate the Integration

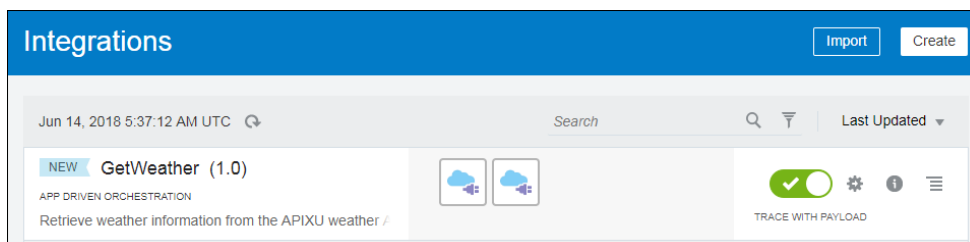
Your integration is now ready to go live. Let's activate the integration.


1. On the Integrations page, in the row for the **GetWeather** integration, click **Configured. Switch to activate** .

This displays the Activate Integration dialog.

2. Enable tracking to verify that the integration is working as expected.
  - a. Select the **Enable Tracing** check box.
  - b. Select the **Include payload** check box so that you can verify the logs. This option is not recommended for a production environment.
3. Click **Activate**.

It might take a few moments before the integration is activated. Click **Refresh**  if the status is in progress. The icon turns green when the integration is active and ready to be invoked.



4. Next, obtain the endpoint URL of your integration. For your integration, click **How to run** , then click the link displayed for **Endpoint URL**.

This opens a new browser tab and displays endpoint connectivity details like the endpoint URL, swagger link to access the API specification, query parameters, response sample, and so on.

5. Under **Endpoint URL**, copy the following part of the URL:

```
https://example.com:443/ic/api/integration/v1/flows/rest/
GETWEATHER/1.0/weather
```

Where:

*example.com* refers to the host name of your Oracle Integration instance. Ensure that you do not copy the query parameters from the URL: `?q=[q-value]&weatherkey=[weatherkey-value]`.

You'll need this endpoint URL when you create the connection to the integration from your Visual Builder application.

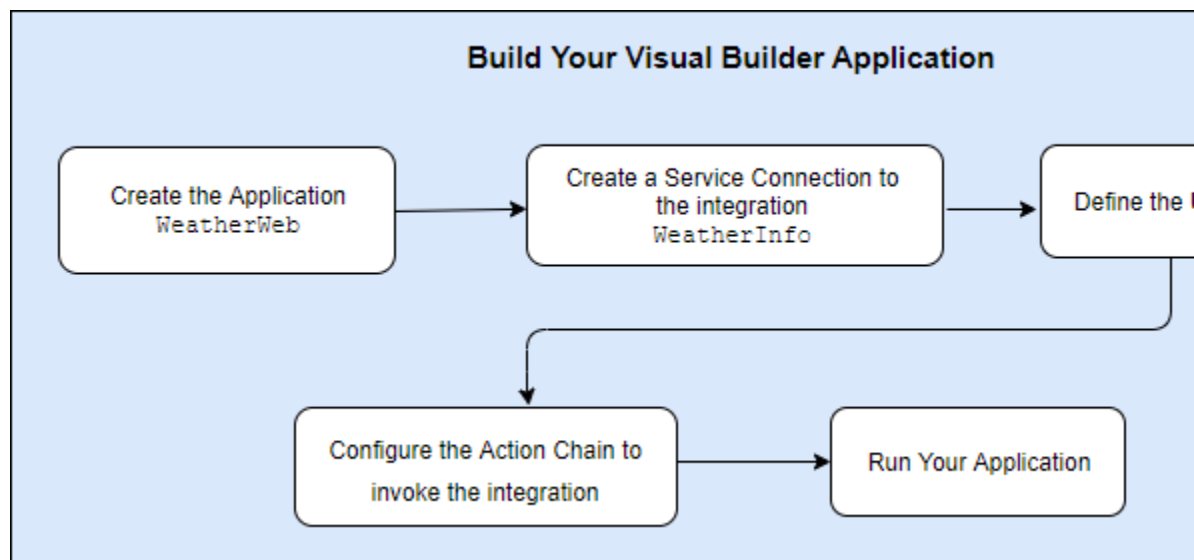
You can keep the browser window open in case you need to copy the endpoint URL again.

# 4

## Module 4 Create and Configure a Web Application with Visual Builder

### Build Your Visual Builder Application: Create the Application

You're now ready to create and configure the front-end for your integration. In this module, you will create a Visual Builder web application named `WeatherWeb` to retrieve weather information. You'll create a service connection (`WeatherInfo`) to the integration. Then, using the visual development tool, you'll design your application page, create and map objects to store and display weather data, and run the application.



First, let's create the web application that your users will use to retrieve weather information.

1. Go to the Integration navigation pane, and click **Visual Builder**.
2. On the **Visual Applications** page, click **New Application**.
3. Provide application details in the Create Application dialog.
  - **Application Name:** Enter the application name as `WeatherWeb`.
  - **Application ID:** The application name is automatically added as the application ID.
  - **Description:** Enter a description of your application.
4. Click **Finish**.

The application is created, and the Welcome page is displayed.

## Create a Service Connection to the Integration

Next, create a connection to the integration that your Visual Builder application will use to invoke the integration for getting weather information.

1. On the Welcome page of your application, click **Service Connections**.  
This displays the **Service Connection** button in the **Services** pane.
2. Click the **Service Connection** button.
3. In the **Create Service Connection** dialog, click **Define by Endpoint** as the source of your connection.
4. In the **URL** field, paste the endpoint URL of your integration that you copied earlier after activating the integration. For example:

```
https://example.com:443/ic/api/integration/v1/flows/rest/  
GETWEATHER/1.0/weather
```

Where:

*example.com* refers to the host name of your Oracle Integration instance.

5. Click **Next**.
6. On the **Service** tab, verify the connection details, which include service base URL, service name, and service ID.
7. Click the **Authentication** tab, and configure the user name and password.
  - From the **Authentication Mechanism** list, select **Basic**.
  - In the **Username** and **Password** fields, provide the credentials to access your Oracle Integration instance.
8. Click the **Request** tab, and add the request query parameters.
  - a. Click the **URL Parameters** tab.
  - b. In the **Query Parameters** section, click **Add**, then select **Dynamic Parameter**.
  - c. In the **Name** field, enter *q*. By default, **Type** is selected as **String**.
  - d. Select the **Required** check box.
  - e. Similarly, add the *weatherkey* parameter. Notice that these are the same parameters that you defined in the trigger connection while creating your integration.
9. Click the **Test** tab to test your connection.
  - a. In the **Request** section, enter values for query parameters. For example, in the **Value** field for the *q* parameter, enter *90210* and in the **Value** field for the *weatherkey* parameter, specify your API application key.
  - b. Click **Send**.

The **Response** section displays the response data returned by the integration.

**Create Service Connection**

Method: GET Path: /weather Action Hint: Retrieve Many

Service Authentication Request Response **Test**

**Request**

Body Headers **URL Parameters**

Parameter	Name	Type	Value	Required
query	q	string	90201	<input checked="" type="checkbox"/>
query	weatherkey	string		<input checked="" type="checkbox"/>

Reset to Defaults

Send Status: 200 OK

**Response**

Body Headers

```
{
  "localtime": "2018-06-27 21:39",
  "temp_C": "19.4",
  "temp_F": "66.9",
  "WeatherDesc": {
    "value": "Clear"
  },
  "windspeedMiles": "0.0",
  "windspeedKmph": "0.0",
  "winddir16Point": "N",
  "winddir16Point": "N"
}
```

Copy to Response Body

< Back Cancel Create

- c. In the Response section, click **Copy to Response Body**. This copies the response data as a sample payload to the **Response** tab.

10. Click the **Response** tab.

Notice that the sample payload you copied is displayed.

11. Click **Create**.

The WeatherInfo service connection is now created.

12. Close the service connection page.

## Define the User Interface

Now create a web application and design its page by adding fields that will display the weather information. Also, create variables to store information and then map them to fields.

Visual Builder provides a simple visual development tool that lets you drag and drop components to pages.

### Create a Web Application

1. On the Welcome page of your application, click **Web Apps**.

This displays the **Web Apps** button in the **Web Apps** pane.


2. Click the **Web Application** button.

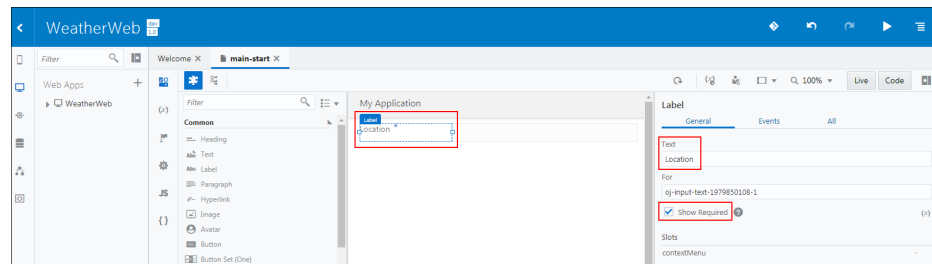
3. In the Create Web Application dialog, in the **Id** field, enter `WeatherWeb` and click **Create**.

This displays the **main-start** page of your web application.

## Define the Fields

Your application will comprise various fields to display weather information, and a button to invoke the integration. So, add Input Text elements and the button to the **main-start** page.

1. Add a field named Location.
  - a. From the **Field** section in Component Palette, drag **Input Text** to the canvas.
  - b. Click the **Input Text** label on the canvas.
  - c. Click **Expand Property Inspector**  to display the Property Inspector pane.
  - d. In the **Label** pane in Property Inspector, in the **Text** field, enter `Location`.
  - e. Select the **Show Required** check box to mark the **Location** field as mandatory.



2. Add the **Get Weather Information** button that will be used to invoke the integration.
  - a. From the **Common** section in Component Palette, drag **Button** to the canvas.
  - b. In the **Button** pane in Property Inspector, in the **Text** field, enter `Get Weather Information`.
3. From the **Layout** section in Component Palette, drag **Horizontal Rule** to the canvas to divide the page.
4. Next, add all the input fields listed in the table.

To add a field, drag the **Input Text** component to the canvas and select **Readonly** in the **Input Text** pane in Property Inspector. Then, select the **Input Text** label on the canvas and enter the label in the **Text** field in the **Label** pane in Property Inspector.

Component in Component Palette	Label in Property Inspector	Input Text Property in Property Inspector
Input Text	Local Time	Readonly
Input Text	Temp F	Readonly
Input Text	Temp C	Readonly
Input Text	Weather Description	Readonly
Input Text	Wind Speed Miles	Readonly
Input Text	Wind Speed km	Readonly
Input Text	Wind Direction	Readonly
Input Text	Humidity	Readonly

Component in Component Palette	Label in Property Inspector	Input Text Property in Property Inspector
Input Text	Precipitation MM	Readonly

Your page would like the following sample page.

My Application

Location \*

Local Time

Temp F

Temp C

Weather Description

Wind Speed Miles

Wind Speed km

Wind Direction

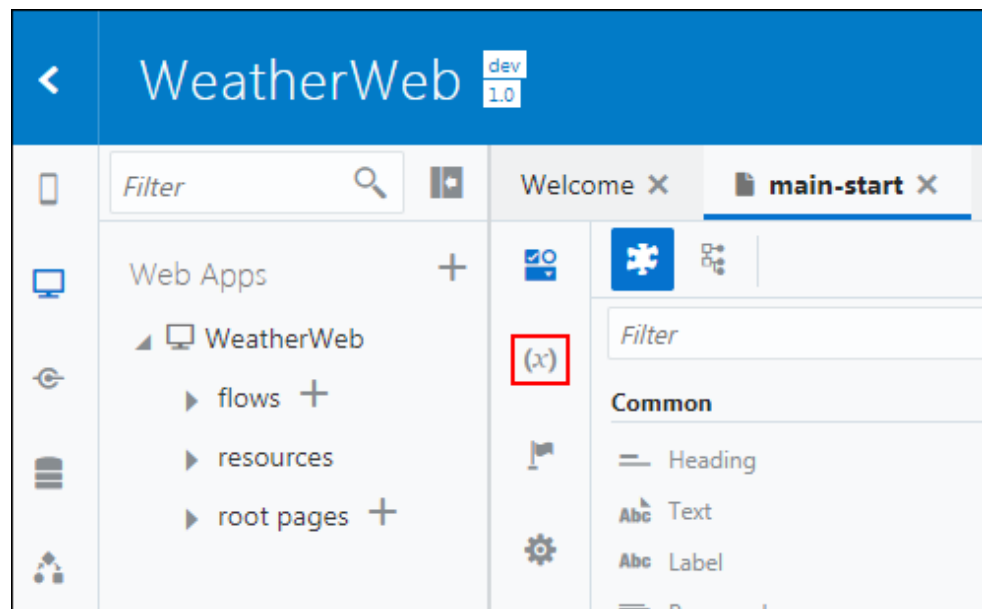
Humidity

Precipitation MM

### Define the Variables

Now, corresponding to each component you added, add variables to store weather information.

1. On the **main-start** page, click **Variables** (x).

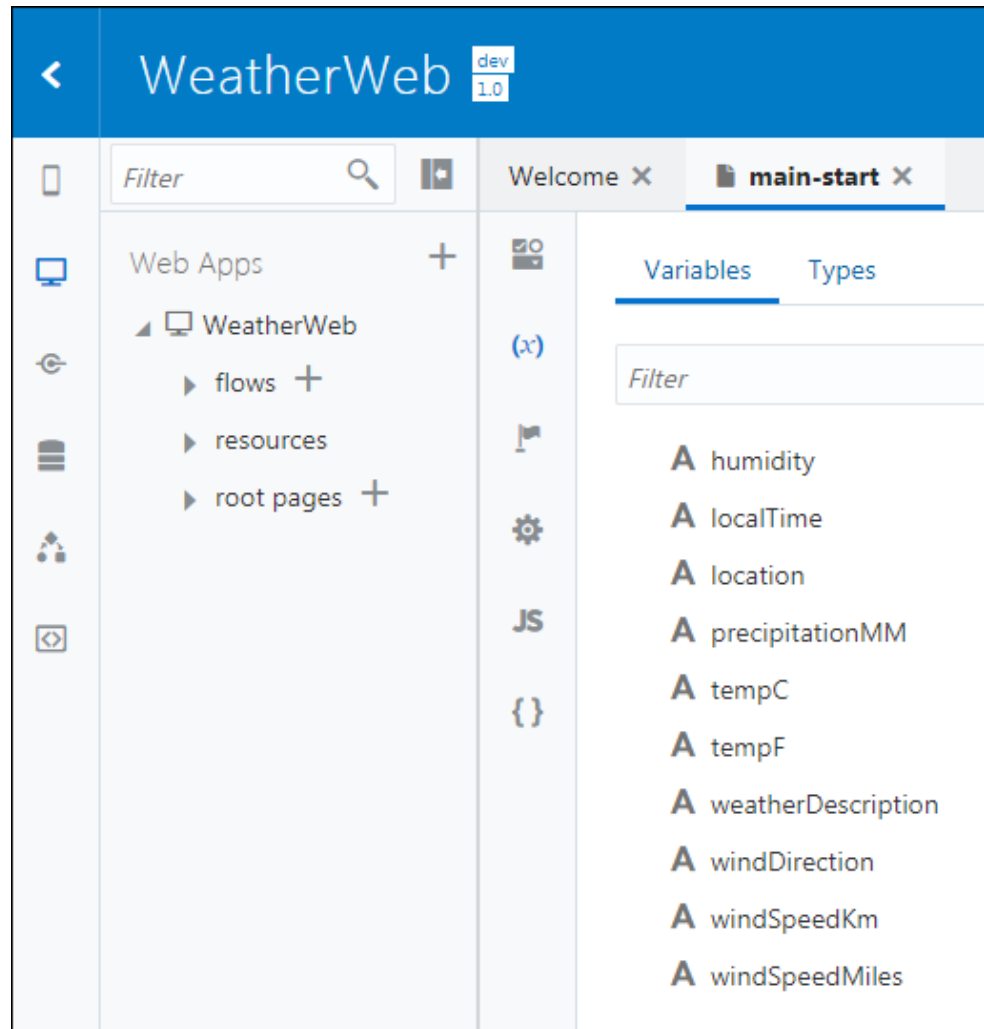


2. Click the **+Variable** button.

3. In the New Variable dialog, select the **Create Another** check box.
4. In the **Id** field, enter the name of the variable and click **Create**. By default, the variable type **String** is selected.




Create the following variables:

- humidity
  - localTime
  - location
  - precipitationMM
  - tempC
  - tempF
  - weatherDescription
  - windDirection
  - windSpeedKm
  - windSpeedMiles
5. When you are done, click **Cancel** in the New Variable dialog to close the dialog.
- The newly created variables are listed on your application page.

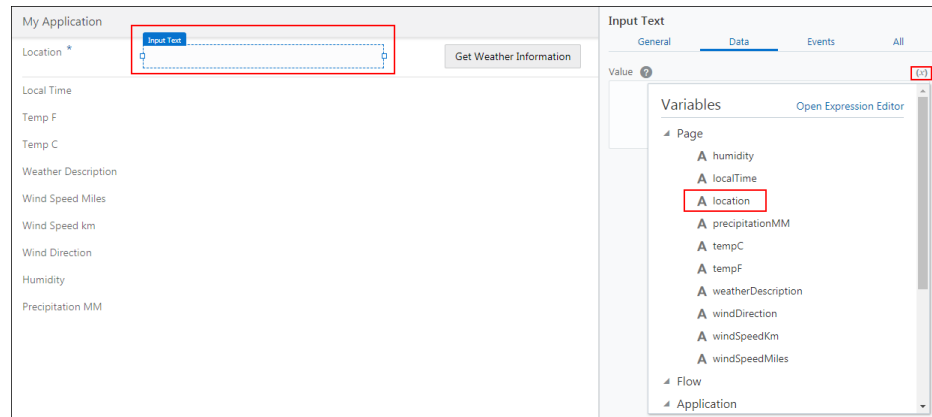


### Map Fields to Variables

Next, bind each field to the corresponding variable.

1. Click **Designer**  to go back to the canvas.
2. Map the **Location** field to the **location** variable:
  - a. Click the **Location** field on the canvas. Ensure that you click the input text and not the label of the Location field.
  - b. In the **Input Text** pane, click the **Data** tab.
  - c. Move the mouse pointer to the **Value** box. This displays the **Variables**  icon. Click **Variables** .
 

The list of variables is displayed.
  - d. From the **Variables** list, select the **location** variable to bind it to the **Location** input text field.



3. Similarly, bind the following fields to the corresponding variables. Ensure that you select the Input Text, and not Label of a field.

Field	Variable
Local Time	localTime
Temp F	tempF
Temp C	tempC
Weather Description	weatherDescription
Wind Speed Miles	windSpeedMiles
Wind Speed km	windSpeedKm
Wind Direction	windDirection
Humidity	humidity
Precipitation MM	precipitationMM

## Configure the Action Chain to Invoke the Integration

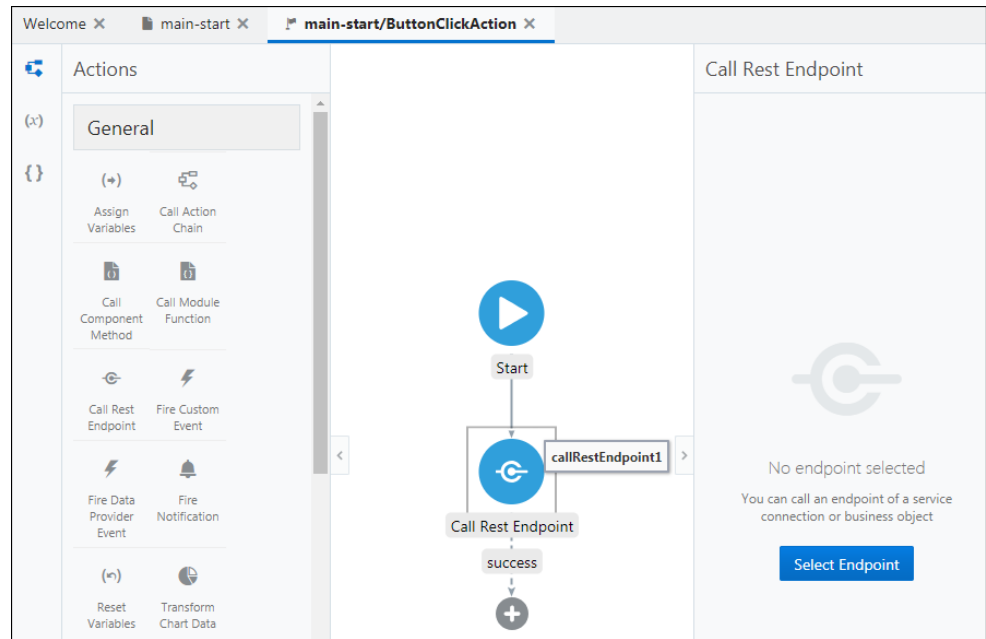
Now that your page is ready, let's define how the integration will be invoked. In our case, the integration will be invoked when the **GetWeatherInformation** button is clicked in the application.

### Create an Event to Invoke the Integration

First, let's create the event to invoke the integration.

1. On the main-start page, click the **GetWeatherInformation** button on the canvas.
2. In the **Button** pane, click the **Events** tab.
3. Click **New Event**, then **Quick Start: 'click'**.
4. On the main-start/ButtonClickAction page, from the **General** category in the **Actions** pane, drag the **Call REST Endpoint** action and drop it below the **Start** icon on the canvas.

The **callRestEndpoint1** object is added to the page.



5. In the **Call Rest Endpoint** pane, click **Select Endpoint**.
6. In the Select Endpoint dialog, select your integration as the endpoint. Expand **Service Connections**, then your service connection, then select **/weather**.
7. Click **Select**.

The service connection shows the input parameters from the getWeather integration.

### Call Rest Endpoint

**Id \***

**Label**

**Description**

**Endpoint \*** [Select](#)

icApiIntegrationV1FlowsRestGETWEAT...

**Input Parameters** [Assign](#)

**A** q \*

NOT MAPPED

**A** weatherkey \*

NOT MAPPED

**Parameters** [Assign](#)

**A** filePath

NOT MAPPED

**Request Type**

☐ json

☐ form

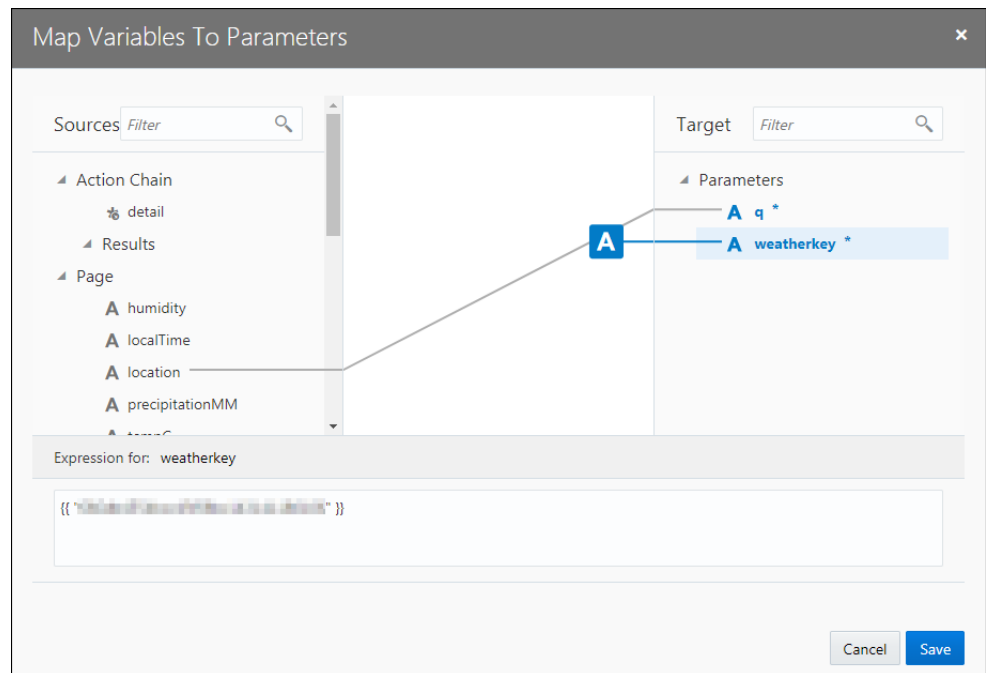
☐ url

- In the **Call Rest Endpoint** pane, click **Not Mapped** for either the **q** or **weatherkey** parameter.

The Map Variables to Parameters dialog is displayed. The **Source** pane shows the variables you added to your page, and the **Target** pane shows the input parameters defined in the integration. You'll now map the integration parameters to the corresponding page variables.

- From the **Source** pane, drag the **location** variable to the **q** parameter in the **Target** pane.

10. In the **Target** pane, select the **weatherkey** variable. Then in the **Expression for** box at the bottom of the screen, enter the API application key as a constant within quotation marks.
11. Click **Save**.

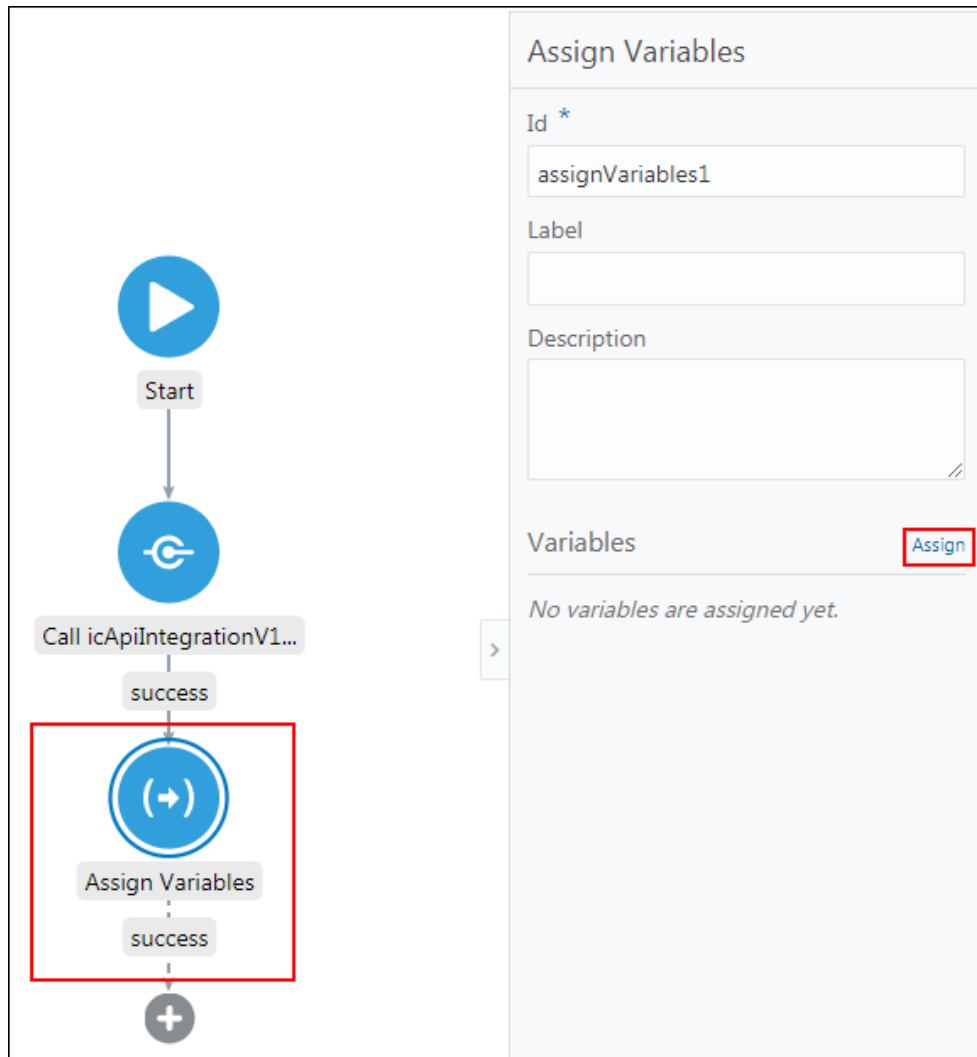


12. The input parameters now appear mapped. Select **json** as the **Request Type**.

### Map Response Data to Page Variables

Your integration will receive weather forecast from the weather API and pass on the information to the Visual Builder application. So, now map the response data from the weather API to your page variables.

1. From the **Actions** pane, drag the **Assign Variables** action to the plus icon below the call REST endpoint on the action chain.
2. In the **Assign Variables** pane, click **Assign**.



The Map Variables To Parameters dialog is displayed. The **Target** pane displays the page variables defined in your Visual Builder application.

3. In the **Source** pane, under **Action Chain**, under **Results**, expand **callRestEndpoint1**, then expand **body**.

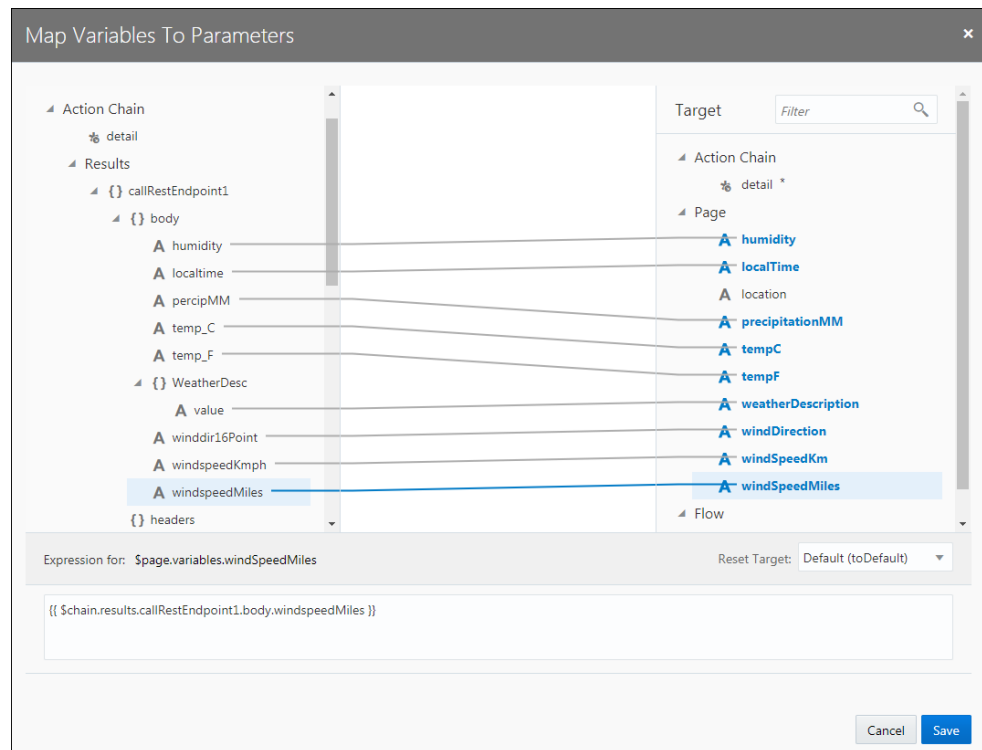
The **Source** pane now displays the parameters defined in your integration. You had defined these parameters in the sample payload of the trigger connection of your integration.

4. From the **Source** pane, drag the **humidity** attribute and drop it onto the **humidity** parameter in the **Target** pane. A connecting line between the two parameters represents the mapping.
5. Similarly, map all the variables as listed in the following table.

Source Pane Variables	Target Pane Variables
humidity	humidity
localtime	localTime
precipMM	precipitationMM

Source Pane Variables	Target Pane Variables
temp_C	tempC
temp_F	tempF
WeatherDesc, value	weatherDescription
winddir16Point	WindDirection
windspeedKmph	windSpeedKm
windspeedMiles	windSpeedMiles

The parameters are now mapped.



- Click **Save** to save the mapping.

In the **Assign Variables** pane, all the variables are now listed as **Mapped**.

### Assign Variables

Id \*

assignVariables1

Label

Description

Variables


Assign

humidity	MAPPED
localTime	MAPPED
precipitationMM	MAPPED
tempC	MAPPED
tempF	MAPPED
weatherDescription	MAPPED
windSpeedMiles	MAPPED
windSpeedKm	MAPPED
windDirection	MAPPED

Your application is now ready to invoke the integration to retrieve weather information.

## Run Your Application

It's time to see it all working. Let's run the application and verify that it displays the current weather information for the specified location.

1. In your application, click **Run** .  
Your application is displayed at runtime.
2. In the **Location** field, enter a zip code, city, or state name. For example, enter 90210.
3. Click **Get Weather Information**.

Congratulations! Your application shows the current weather information for the location you entered.

My Application	
Location *	<input type="text" value="90210"/> <input type="button" value="Get Weather Information"/>
Local Time	2018-06-28 23:44
Temp F	64.0
Temp C	17.8
Weather Description	Clear
Wind Speed Miles	0.0
Wind Speed km	0.0
Wind Direction	N
Humidity	81
Precipitation MM	0.0