Define the parameters

In the previous topic, you added an operation to your basic OpenAPI Specification structure to retrieve information about an airport. Now you will add a parameter to pass the airport code of the airport that you want to retrieve information on. This parameter appears as a configuration field for the action in the Workflow designer.

Summary

A parameter is a piece of information required by the API in order to complete the operation. Parameters can be passed to operations in several locations:

- Appended to the URL as a **query**
- Included in the URL as part of the **path**
- Passed in the **header** of the HTML
- Passed in the **body** of the HTML
- Passed as the content of a form

In this example, you will define one parameter to be appended to the URL as a query. For instructions on defining multiple parameters, see Defining more than one parameter. For examples on how to pass parameters by other methods, see Alternative parameter locations.

Parameters in the OpenAPI Specification

Parameters for each operation are defined in the **parameters** array within the HTTP method object. Each parameter must have a unique name within the location it is passed in for the OpenAPI Specification to be valid. For example, you could have two parameters with the name of ID if one were passed in the query, and the other in the body, but not if both parameters were passed in the query.

Parameters must be defined as they appear in the API. If the API requires the parameter in a query, your OpenAPI Specification must define the parameter as being passed in the query.

Define the parameters

To define a parameter, you define:

- The name of the parameter
- The data type of the parameter
- The location in which the parameter is passed

Step 1: Add the parameters array

Each operation in an OpenAPI Specification defines its own array of parameters within the HTTP method object. The **parameters** array is usually added after the **produces** array, before the closing brace of the HTTP method object.

Copy

```
"get": {
  "summary": "Get airport info",
  "description": "Get airport info",
  "operationId": "getAirportInfo",
  "produces": [
    "application/json"
  ],
  "parameters": [

  ],
```

Step 2: Add a parameter object

Each parameter is defined as an object within the parameters array, with each parameter object having multiple key : value pairs. Add an open and closing brace to create an empty object within the **parameters** array to hold the parameter.

Copy

```
"parameters": [
  {

  }
],
```

Defining more than one parameter

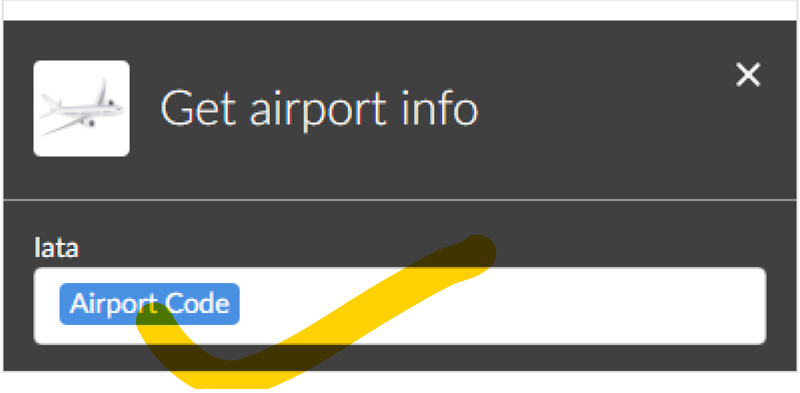
Step 3: Add the parameter name and type

Parameters must have a **name** and a data **type**, such as **string** (text) or **number**.

Copy

```
"parameters": [
  {
    "name": "iata",
    "type": "string",
  }
],
```

Nintex Workflow Cloud uses the **name** and **type** to create the configuration field for this action in the Workflow designer.



Get airport info

Iata

Airport Code

Step 4: Add the parameter location

Parameters must have a location, which specifies how they are passed to the API. The location is defined with the key **in**. Passing the parameter in the query appends it to the API URL.

Copy

```
    "parameters": [
      {
        "name": "iata",
        "type": "string",
        "in": "query"
      }
    ],
```

Alternative parameter locations

Passing the parameter in the header

Passing the parameter in the path

Passing the parameter in the body

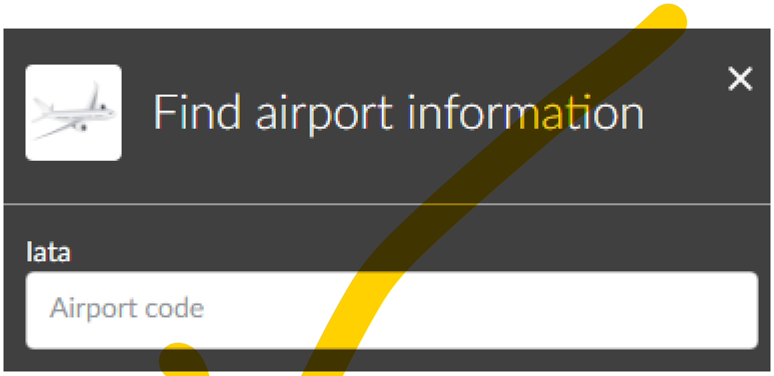
Step 5: Add watermark text

You can guide designers with watermark or placeholder text when they configure your custom actions. Nintex Workflow Cloud uses the **description** key in the parameter object to create watermark text in the configuration field for the action.

Note: Watermark text is not displayed for enum parameters, or other drop-down list configuration fields.

Copy

```
    "parameters": [
      {
        "name": "iata",
        "type": "string",
        "in": "query",
        "description": "Airport code"
      }
    ],
```



Find airport information

Iata

Airport code

The OpenAPI Specification

The OpenAPI Specification now defines the operations you can perform, and the parameters needed for those operations. Next you will describe the response the API will return when the operation is called.

Copy

```
{
  "swagger": "2.0",

  "info": {
    "version": "1.0.0",
    "title": "Airport Data"
    "description": "Retrieves location information based on airport codes"
  },

  "host": "www.airport-data.com",
  "basePath": "/api",
  "schemes": [
    "http"
  ],

  "produces": [
    "application/json"
  ],

  "paths" : {
    "/ap_info.json": {
      "get": {
        "summary": "Get airport info",
        "description": "Get airport info",
        "operationId": "getAirportInfo",
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "iata",
            "type": "string",
            "in": "query",
            "description": "Airport code"
          }
        ],

      }
    }
  }
}
```

Next steps

Define the responses

Resources

Search Resources

SDKs

Learning Center

Product Help

Status

Connect

Community

Partner Central

Nintex Xchange

Blog