Build the basic OpenAPI Specification structure

Summary
OpenAPI Specification in JSON
Create an empty OpenAPI structure
Step 1: Create an empty file
Step 2: Add the outer braces
Step 3: Add the OpenAPI Specification version
Step 4: Add the API information
Step 5: Add the Connection information
Step 6: Add the data type the API produces
The OpenAPI Specification
Next steps

In the following topics, you will build a simple OpenAPI Specification for an API that retrieves the name, city and country of an airport when given the airport's three-letter IATA code, such as JFK.

The first step is to create the basic structure of the OpenAPI Specification.

Note: These topics are based on the OpenAPI 2.0 Specification.

The complete OpenAPI Specification and icon for this example are available here.

Summary

All OpenAPI Specifications have the same basic structure that controls how the information is stored in the file. It is important to follow the structure to ensure you are creating a valid OpenAPI Specification. Nintex Xtensions only accepts valid OpenAPI Specifications.

OpenAPI Specification in JSON

OpenAPI Specifications for Nintex Xtensions are written in JSON, a data format based on JavaScript. If you're unfamiliar with JSON, you can build your own OpenAPI Specification by copying from the examples, and remembering a few key rules:

- Information is structured in key : value pairs separated by a colon
- A value may be:

  A single value

  An array of multiple values of the same type, separated by commas and enclosed in square brackets []

  A object made of different key : value pairs, separated by commas and enclosed in braces {}
- All text is enclosed in double quotation marks

Create an empty OpenAPI structure

All OpenAPI Specifications need some basic information:

- The OpenAPI Specification version the file is using
- The API information, which includes the version, title and a short description
- Basic connection information for the API: how to find the API, and what connection types are accepted
- The formats the API uses to return data

Step 1: Create an empty file

You can use any text editor to create an OpenAPI Specification. An online editor such as the Swagger.io editor can provide helpful features such as automatic indenting and syntax highlighting, as well as the ability to interactively test your API calls as you build them.

If you decide to create the file using your own text editing software, make sure you save it as a JSON file (.json) instead of a text file (.txt).

Step 2: Add the outer braces

The entire OpenAPI Specification is written between an open and closed brace.

Copy

```
{

}
```

Step 3: Add the OpenAPI Specification version

The OpenAPI Specification version goes at the top of every OpenAPI Specification under the open brace, to identify which version of the OpenAPI Specification has been used. The OpenAPI Specification version is identified by its former name of **Swagger**. Nintex Xtensions supports version 2.0 of the OpenAPI Specification.

Copy

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

}
```

Step 4: Add the API information

The API information is an object that contains:

- The version of the API that this OpenAPI Specification describes
- The title of the API
- The brief description of the API's purpose

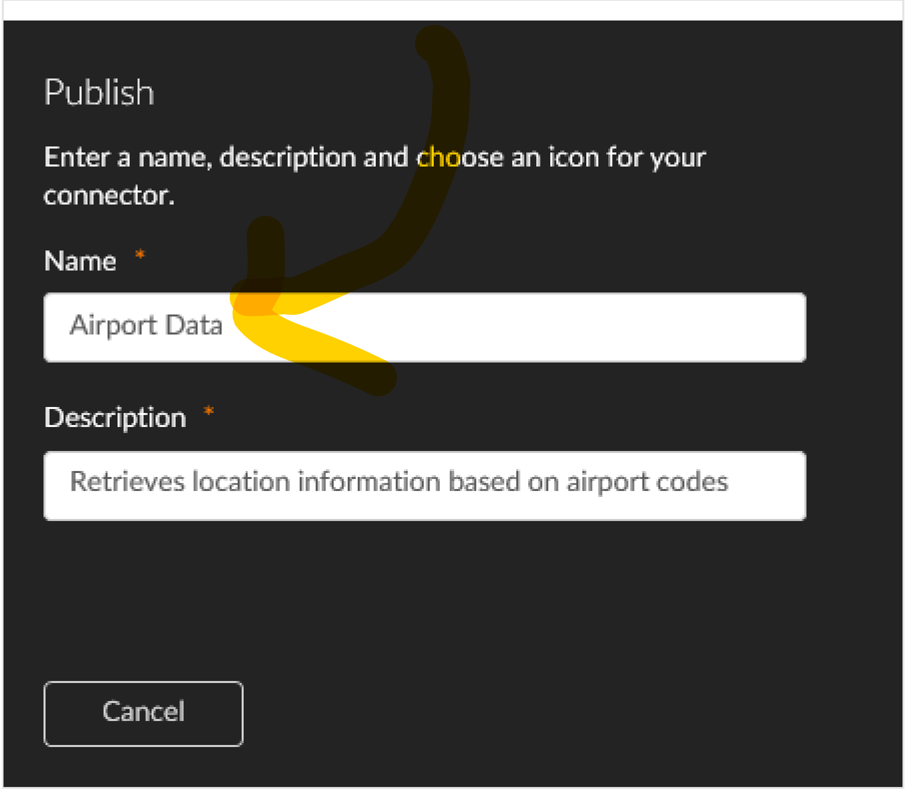
The API information is usually written below the OpenAPI Specification version.

Copy

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

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

When the OpenAPI Specification is imported into Nintex Workflow Cloud, the **title** value is used to populate the **Name** of the **action** group.



Publish

Enter a name, description and choose an icon for your connector.

Name *

Airport Data

Description *

Retrieves location information based on airport codes

Step 5: Add the Connection information

The connection information details how to connect to the API:

- The host URL where the API exists.
- The base path of the API that indicates where the API exists within the host. All API calls are made with the host and base path. For example, www.airport-data.com/api/ap-info.json.
- An array of the types of connections, called **schemes**, that the API allows.

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"
  ],

}
```

If your API allows more than one connection type, separate each type with a comma in the **scheme** array:

Copy

```
"schemes": [
  "http", "https"
],
```

Nintex Xtensions supports HTTP and HTTPS connections.

Step 6: Add the data type the API produces

The data type, or MIME type, is an array that lists the formats in which the API returns information. This is defined for the entire API, but can be overridden for individual operations.

```
{
  "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"
  ],

}
```

Nintex Workflow Cloud supports application/json.

The OpenAPI Specification

This is the basic OpenAPI Specification structure. It declares what the API is called, and how to find and connect to it, but it does not yet include any of the operations that can be called, or the parameters or responses. The following topics add each of these components to create a complete OpenAPI Specification.

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"
   ],


}
```

Next steps

Define the operations

Resources

Search Resources

SDKs

Learning Center

Product Help

Status

Connect

Community

Partner Central

Nintex Xchange

Blog