SpecGen – <OpenAPI> Elements

Contents

[Overview 1](#_Toc73345483)

[DataObject – <OpenAPI> Elements 1](#_Toc73345484)

[“Exclude Operations”-Element: <ExcludeOperations> 2](#_Toc73345485)

[Configure behaviour of a specific endpoint operation 2](#_Toc73345486)

[Add custom Paths or Service Paths 4](#_Toc73345487)

[Group – <OpenAPI> Elements 6](#_Toc73345488)

## Overview

This document lists all available <OpenAPI> elements that can be used in the SpecGen source files. These <OpenAPI> elements allow controlling and/or customising certain aspects of the final OpenAPI definition (yaml files) output of the SpecGen process.

The <OpenAPI> elements are consumed by the XSLT process that produces Open API yaml/json files. The XSLT files that utilise the <OpenAPI> elements are:

* dmToOpenAPI.xsl: Produces the top level Open API yaml file
* dmToExamples.xsl: Produces the XM/JSON example yaml file

The <OpenAPI> elements can be added in the SpecGen source XML file(s). They have to be in a specific spot/node in the source file and all available <OpenAPI> elements are wrapped in a node called <OpenAPI>.

**Example:**

<DataObject name="StudentPersonal" ...>

**<OpenAPI>**

...

**</OpenAPI>**

<Key>@id</Key>

<EventsReported>false</EventsReported>

...

</DataObject>

## DataObject – <OpenAPI> Elements

As most objects listed in the SpecGen source files have an identical behaviour, meaning having the same API functionality, it is much easier to just list the exceptional behaviour. Based on this approach most <OpenAPI> elements work on the ‘exclude’ concept. Only functionality that falls outside the standard behaviour need to be configured in the SpecGen source files to exclude certain functionality. This ensures that minimal amount of Open API configuration is required in the SpecGen source files. For example in the SIF AU data model SpecGen source files there are no <OpenAPI> elements at all since all objects follow the standard SIF API functionality. However the SIF Infrastructure data model SpecGen files have a number of <OpenAPI> elements because almost each object in the infrastructure specification has some API functionality disabled/not supported. So, the infrastructure SpecGen source files are a good source for many examples on how to use the <OpenAPI> elements.

To control/configure some behaviour of a data object endpoint, appropriate <OpenAPI> elements can be added a specific <**DataObject**> element:

**Example:**

<**DataObject** name="environment" ...>

**<OpenAPI>**

...

**</OpenAPI>**

...

</**DataObject**>

### “Exclude Operations”-Element: <ExcludeOperations>

This element can be used to exclude operations for an endpoint. Any number of operations can be excluded by simply provide a comma separated list of operations to be excluded. The following are valid values (case-sensitive):

* **ALL**: Will exclude all operation of an endpoint. Basically the endpoint will no longer be listed in the Open API yaml file.
* **all-batch**: Will exclude all batch operation of an endpoint including admin directives as these are ‘batch’ style operations.
* **all-single**: Will exclude all single object operations of an endpoint.
* **get-single**
* **get-batch**
* **post-single**
* **post-batch**
* **put-single**
* **put-batch**
* **delete-single**
* **delete-batch**
* **head-single**
* **head-batch**
* **adminDirective:** Will exclude the admin directive endpoints (GET & HEAD).

**Examples**

Example 1: Exclude all batch operations and the HTTP head operations:

<OpenAPI>

**<ExcludeOperations>all-batch, head-single</ExcludeOperations>**

</OpenAPI>

Example 2: Exclude HTTP PUT (single & batch) and HTTP DELETE (single & batch) operations:

<OpenAPI>

**<ExcludeOperations>**

**put-single, put-batch, delete-single ,delete-batch**

**</ExcludeOperations>**

</OpenAPI>

### Configure behaviour of a specific endpoint operation

It is possible to configure various aspects of each available operation of an endpoint. For example the HTTP PUT for a single object may exclude specific HTTP Error Status and/or HTTP Headers while the batch version of the HTTP PUT supports the standard error status and HTTP headers. Such operation specific configuration is supported with an appropriate OpenAPI node. Each operation has its specific node name. The following node names are available:

* GetBatch
* GetSingle
* PostBatch
* PostSingle
* PutBatch
* PutSingle
* DeleteBatch
* DeleteSingle

Example: Configure behaviour of the HTTP GET for a batch operation

<OpenAPI>

**<GetBatch>**

**...**

**<GetBatch>**

</OpenAPI>

The aspects/behaviour below can be configured by using the specific element name under the specific operation node.

* **ExcludeQueryParams**: Comma separated list of URL Query parameters to exclude. ALL will exclude all URL Query parameters. This node is only applicable for the <GetSingle> and <GetBatch> operation.
* **ExcludeRequestHTTPHeaders**: Comma separated list of HTTP Headers to exclude in request. ALL will exclude all.
* **ExcludeResponseHTTPHeaders**: Comma separated list of HTTP Headers to exclude in response. ALL will exclude all.
* **ExcludeErrorCodes**: Comma separated list of HTTP Error Status codes to exclude in response. ALL will exclude all.
* **DelayedSupported**: Indicator if the endpoint supports DELAYED request/response for batch operations.
  + **true**: operation does support delayed requests. Only applicable for 'batch' operations.
  + **false**: operation does **NOT** support delayed requests. Only applicable for 'batch' operations.
  + If not provided then 'true' is assumed for 'batch' operations
* **QBESupported**: Indicator if the endpoint supports "Query by Example (QBE)". Only applicable for the <PostBatch> node.
  + **true**: QBE is support.
  + **false**: QBE is **NOT** supported.
  + If not provided then 'true' is assumed for '<PostBatch>

**Example 1:**

A certain object does not support any URL query parameters and doesn’t support delayed queries (HTTP GET):

<OpenAPI>

<GetBatch>

<ExcludeQueryParams>ALL</ExcludeQueryParams>

<DelayedSupported>false</DelayedSupported>

<GetBatch>

</OpenAPI>

**Example 2:**

Exclude the HTTP Error Code 501 and 503 for the single object HTTP PUT.

<OpenAPI>

<PutSingle>

<ExcludeErrorCodes>501, 503</ExcludeErrorCodes>

</PutSingle>

</OpenAPI>

**Example 3:**

An object doesn’t support the ‘order’ and ‘where’ URL query parameter for the HTTP GET, doesn’t support “changes since” (controlled by specific URL query parameters in the request and HTTP header in the response), doesn’t support delayed queries, doesn’t support any batch operations except the HTTP GET and the requestor cannot ask for accepting ‘mustUseAdvisory’ values for UUIDs.

<OpenAPI>

<ExcludeOperations>

post-batch, put-batch, delete-batch, head-batch

</ExcludeOperations>

<GetBatch>

<ExcludeQueryParams>

where, order, changesSinceMarker

</ExcludeQueryParams>

<ExcludeResponseHTTPHeaders>

changesSinceMarkerGet

</ExcludeResponseHTTPHeaders>

<DelayedSupported>false</DelayedSupported>

</GetBatch>

<PostSingle>

<ExcludeRequestHTTPHeaders>

mustUseAdvisory

</ExcludeRequestHTTPHeaders>

</PostSingle>

</OpenAPI>

### Add custom Paths or Service Paths

Some data model objects may support additional operations that are specific to an implementation or a specification. Since the SIF API supports all common HTTP predicates for an object the ‘additional’ operations are typically service paths. They do not form part of an object definition and hence are not configured or listed in the SpecGen’s source files. However an implementation may have the need to add a certain set of service paths to their Open API definition. This can be achieved by adding the list of <Path> elements of the desired set of service paths to an <OpenAPI> element.

**Example:**

<OpenAPI>

<Paths>

<**Path>**

**...**

**</Path>**

**<Path>**

**...**

**</Path>**

</Paths>

</OpenAPI>

For each path there are two bits of information required:

* **URI**: The Open API notation of the path (the service path).
* **Ref**: The Open API reference of the external yaml file with the path of the Open API definition of the operation (input, payloads etc.). This is most likely a static but manually maintained yaml file.

**Example:**

A good example is the ‘Job’ object in the infrastructure specification. The job phases as well as the states of each phase are ‘service paths’ that need to be added to the Open API definition of a ‘Job’ object. The details of these service paths are defined in a static yaml file called infraPaths.yaml.

<DataObject name="job" ...>

<OpenAPI>

<Paths>

<Path>

**<URI>/job/{id}/{phaseName}</URI> <-- serve path -->**

**<Ref>infraPaths.yaml#/infraPaths/jobPhasePaths</Ref>**

</Path>

<Path>

**<URI>/job/{id}/{phaseName}/states</URI>**

**<Ref>infraPaths.yaml#/infraPaths/jobPhaseStatesGet</Ref>**

</Path>

<Path>

**<URI>/job/{id}/{phaseName}/states/state</URI>**

**<Ref>**

**infraPaths.yaml#/infraPaths/jobPhaseStatesPost**

**</Ref>**

</Path>

</Paths>

</OpenAPI>

...

</DataObject>

## Group – <OpenAPI> Elements

The SpecGen source files may have some objects grouped within a certain area; however these objects are not top-level objects that should be appearing in the final Open API definition as an endpoint. One such object is the ‘deleteRequest’ in the infrastructure data model that is listed in the ‘request’ group. This object is an object that is used as payload for a standard data model delete request but it is not an endpoint in its own right, so that object as well as the group it belongs should not form part of the Open API endpoint definition.

How to exclude a specific <DataObject> from the output has been described in the previous section. Just excluding the unnecessary objects would still produce the group name in the final OpenAPI yaml file without any objects. To also exclude the group name/heading from the final Open API yaml the following <OpenAPI> element can be added to the <Group> element:

**Example:**

<Group name="...">

<OpenAPI>

**<ExcludeGroupName>true</ExcludeGroupName>**

</OpenAPI>

...

</Group>