Web Services API Reference

Return Values

4/13/2025

Return Values

This is part of the API Basics section of the Web Services API.

Contents

Overview

All Web Services API methods return a result for each operation in addition to the HTTP status code. The results are different depending on the type of operation.

All Methods

All methods always return the `statusCode` attribute . If an error occurs, that is, if the `statusCode` is not `0`, the response includes error details in the `statusMessage` attribute.

The following status codes are supported:

Code	Description
0	The operation is successful. No `statusMessage` is provided.
1	A required parameter is missing in the request.
2	A specified parameter is not valid for the current state.
3	The operation is forbidden.
4	An internal error occurred. This could occur if an internal error occurred with Web Services or with one of the servers working with Web Services (for example: Cassandra or a Genesys Framework component).
5	The user does not have permission to perform this operation.
6	The requested resource could not be found.
7	The operation was partially successful. Returned if at least one action in a bulk operation succeeded. More information is available in the Partial Success section.
8	Change password demanded. Web Services requested a password change for the user.
9	Processing incomplete
10	Input validation error - the provided value is not within the range of valid values
11	User requested to change read-only property
12	Unable to retrieve resource error
13	Unable to create resource error
14	Unable to delete resource error
15	Unable to update resource error

Code	Description
16	Unable to assign resource error
17	Unable to unassign resource error
18	Resource already exists
19	Resource already in use
20	User is not authenticated. Any subsequent request should provide credentials

If an error occurs during an operation, the response includes `statusCode` and `statusMessage` to clarify the error. No other attributes are included.

Note that if an error occurs during a request, you can assume that the request failed to modify the data of the contact center.

GET

GET requests are used to retrieve a variety of information and the response body will depend on what is being requested as well as the request parameters.

These are the possible scenarios:

1. If retrieving a collection of URIs, the response will include the array attribute `uris` which will hold the requested collection and collection or relative uris with array attribute `paths`.
2. If retrieving a collection of resources, the response will include an array attribute named after the requested resource (for example: `GET /users?fields=*` will contain `"users":[ {..user1..}, {...user2...}, and so forth]`
3. If the URI is a singular resource (for example: `GET /users/{id}`) the response will include an attribute named after the singular of the requested resource which will contain the requested value. (for example: `GET /users/{id} will return "user":{...user..}`)

Example

If retrieving a collection of URIs, the response will include the array attribute `uris` which will hold the requested collection.

```
GET /skills
{
        "statusCode": 0,
        "uris": [
                "http://../api/v2/skills/123",
                "http://../api/v2/skills/456",
                ...
        ],
        "paths":[
                "/skills/123",
                "/skills/456",
                ...
        ]
```

```
}
```

Example

If retrieving a collection of resources, the response will include an array attribute named after the requested resource. For example, `GET /users?fields=*` will contain `"users":[ {..user1..}, {...user2...}, etc]`.

```
GET .../users?fields=*
{
    "statusCode":0,
    "users":[
               {
                "userName":"..",
                "firstName":"...",
                 etc},
               }
               {
                "userName":"..",
                "firstName":"...",
                 etc
               }
              ]
}
```

Example

If the URI is a "singular" resource such as `GET /users/{id}`, the response includes an attribute named after the singular form of the requested resource. This attribute contains the requested value. For example, `GET /users/{id}` will return `"user":{...user..}`.

```
GET /devices/{id}
{
    "statusCode": 0,
    "device": {
                "vendor": "...",
                "phoneNumber": "...",
                 ...
               }
}
```

POST to Create Resource

When a POST request is successful, the following extra attributes will be included:

1. `id`—the ID of the newly-created object.
2. `uri`—The URI to access the newly-created object.
3. `path`—The relative URI to access the newly created object.

Example

Request:

```
POST /users
{
 ... some user data
}
```

Response:

```
{
    "statusCode":0
    "id":"12345",
    "uri":"http://...api/v2/users/12345"
    "path": "/users/12345"
}
```

POST to Assign Resource

POST can also be used to assign one resource to another's collection, such as when assigning a skill to a user. When this is the case, no extra attributes are returned and only `statusCode:0` will be returned on success.

DELETE

The DELETE operation does not have any extra attributes. Only `statusCode:0` will be returned on success.

DELETE to Unassign Resource

DELETE can also be used to unassign one resource from another's collection, such as when unassigning a skill from a user. No extra attributes are returned and only `statusCode:0` will be returned on success.

PUT

The PUT operation does not have any extra attributes. Only `statusCode:0` will be returned on success.

Asynchronous Operations

Web Services supports many operations that are performed using POST on an existing resource and the response for which is sent via CometD. When POST is used to perform one of these operations, `statusCode:0` will be returned on success.

Hybrid Operations

In order to increase API usability and minimize network traffic, multi-step operations are occasionally implemented. For instance, it is possible to create a device and assign it to a user with one operation. When hybrid operations are implemented, the methods will return all of the values required for each operation being performed. For example, POST to create a resource requires a return value of `"uri"` and `"id"` whereas POST to assign does not have any extra return values. Implementing a multi-step "create and assign" POST returns `"uri"`, `"id"`, and `"statusCode"` on successful completion.

Partial Success

Some operations may be considered successful if they are able to perform some of their work. These operations are considered "bulk" operations and are different from "transactions", which involve multiple steps that possibly use multiple servers. An example of a transaction is "create user" which involves creating some data in Cassandra as well as Configuration Server. If one of these actions fails, Web Services considers the whole operation a failure. In contrast, an operation such as "assign multiple skills to user" is a bulk operation which consists of a series of transactions (for example, each individual skill assignment is a transaction). The general rule is that if a step of a transaction fails, Web Services considers the whole operation a failure. If at least one transaction in a bulk operation succeeds, Web Services considers this a "partial success." Note that for bulk GETs (for example, GET /users) if the result is a partial list, the response includes statusCode:7 instead of 0. The rest of the result looks the same. For POST, PUT, and DELETE, the partial success returns have the following attributes:

Attribute	Value
statusCode	Always 7
succeeded	An array of resource descriptors (see below). Each represents a resource for which the transaction was successful.
failed	An array of failure descriptors (see below). Each represents a resource for which the transaction failed.

Attribute	Value
uri	The URI of a resource from request parameters for which the transaction succeeded. For example, if assigning multiple skills to a user, this is the URI of a skill).
path	The relative path of the resource
id	The unique identifier of the resource above.

Attribute	Value
<uids>	The attributes which uniquely identify the resource for which this transaction failed. For example, if assigning skill uris, this will be "uri." If creating a user this will be "userName." If a resource has more than one identifying attribute all should be present.

Attribute	Value
statusCode	The status code describing the reason for failure.
statusMessage	The message describing the reason for failure.

Examples

Assign:

```
POST /users/{id}/skills
{
        "uris":["uri1", "uri2"], "paths": ["uri3"]
}

{
        "statusCode":7 (partial success)
        "succeeded":[
                {
                        "id":<id1>,
                        "uri":"uri1",
                        "path":"path1",
                },
                {
                        "id":<id2>,
                        "uri":"uri2",
                        "path":"path2"
                }]
        "failed":[
                {
                        "statusCode":X,
                        "statusMessage":"msg",
                        "uri":"uri3",
                        "path": "path2"
                }
        ]
}
```

Create:

```
POST /users
{
        "users":[
                {
                        "firstName":"..",
                        "lastName":"..",
                        "userName":"u1", etc
                },
                {
                        "firstName":"..",
                        "lastName":"..",
                        "userName":"u2", etc
                },
                {
                        "firstName":"..",
                        "lastName":"..",
                        "userName":"u3", etc
                }

        ]
}
```

```
{
        "statusCode":7 (partial success)
        "succeeded":[
                {
                        "id":<id>,
                        "uri":"uri1",
                        "path":"path"
                },
                {
                        "id":<id>,
                        "uri":"uri2",
                        "path":"path2"
                }]

        "failed":[
                {
                        "statusCode":3,
                        "statusMessage":"Operation forbidden, username already exists",
                        "userName":"u3"
                }]
}
```

Delete:

```
DELETE /users
{
        "uris":["uri1", "uri2"],"paths": ["uri3"]
}

{
        "statusCode":7 (partial success)
        "succeeded":[
                {
                        "id":<id1>,
                        "uri":"uri1",
                        "path":"path1"
                },
                {
                        "id":<id2>,
                        "uri":"uri2",
                        "path": "path2"
                }]

        "failed":[
                {
                        "statusCode":X,
                        "statusMessage":"...",
                        "uri":"uri3",
                        "path": "path2"
                }]
}
```