# Web Services API Reference

## Request Parameters

4/21/2025

# Request Parameters

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

## Contents

# Overview

This outlines the request parameters for the Web Services API.

# Object Fields

## Requesting Devices

When making `list` requests for any kind of object, Web Services returns a list of the corresponding object URIs.

### [+] Requesting Devices - Example

**Request:**

```
GET .../api/v2/me/devices
```

**Response:**

```
{
   "statusCode" : 0,
   "uris" : [
             "http://127.0.0.1:8080/api/v2/devices/ba0f987f-15b4-42c7-bed0-5f302259f9db"
            ]
}
```

## Requesting a list of objects with their actual devices

In order to receive a list of objects with their actual fields, you will need to provide the `fields` request parameter.

### [+] Requesting a list of objects with their actual devices - Example

**Request:**

```
GET .../api/v2/me/devices?fields=*
```

**Response:**

```
{ "devices" : [ { "capabilities" : [ "ForwardCallsOn",
            "DoNotDisturbOn"
          ],
        "deviceState" : "Active",
        "doNotDisturb" : "Off",
        "e164Number" : "5001",
        "id" : "ba0f987f-15b4-42c7-bed0-5f302259f9db",
        "phoneNumber" : "5001",
        "telephonyNetwork" : "Private",
        "userState" : { "displayName" : "Ready",
            "id" : "9430250E-0A1B-421F-B372-F29E69366DED",
            "state" : "Ready"
          },
        "voiceEnvironmentUri" : "http://127.0.0.1:8080/api/v2/voice-environments/a481cd8e-7b6a-4466-af88-db3471ac909e"
      } ],
  "statusCode" : 0
}
```

## Specify data fields when requesting an object

When requesting an object from the Web Services server, it is possible to specify which data fields you receive by providing the `fields` request parameter.

### [+] Specify data fields when requesting an object - Example

**Request:**

```
GET .../api/v2/queues/<queue_id>?fields=id,name
```

**Response:**

```
{
        "id":<queue_id>,
        "name":<queue_name>
}
```

## Requesting all field of an object

To request all fields of an object, set the `fields` property to `*`.

### [+] Requesting all field of an object - Example

**Request:**

```
GET .../api/v2/queues/<queue_id>?fields=*
```

**Response:**

```
{
        "id":<queue_id>,
        "name":<queue_name>,
        "description":<queue_description>,
        ...
}
```

## Requesting Queues

Note that when making "list" requests for any kind of object, Web Services returns a list of the corresponding object URIs.

### [+] Requesting Queues - Example

**Request:**

```
GET .../api/v2/queues
```

**Response:**

```
{
        "statusCode":0,
        "uris":[
```

```
                "http://.../api/v2/queues/<queue_1_id>",
                ...
                "http://.../api/v2/queues/<queue_N_id>"
        ]
}
```

### Request a list of objects with their actual fields

In order to receive a list of objects with their actual fields, you need to provide the `fields` request parameter and have it set either to `*`, or to a list of data fields of interest.

### [+] Request a list of objects with their actual fields - Example

**Request:**

```
GET .../api/v2/queues?fields=id,name
```

**Response:**

```
{
        "statusCode":0,
        "queues":[{
                "id":<queue_1_id>,
                "name":<queue_1_name>
        },
        ...
        {
                "id":<queue_N_id>,
                "name":<queue_N_name>
        }]
}
```

## Object Filtering

It is possible to filter objects using request parameters when doing "list" requests.

For example:

**Request:**

```
GET .../api/v2/queues?fields=id,name,channel&channel=voice
```

**Response:**

```
{
        "statusCode":0,
        "queues":[{
                "id":<queue_1_id>,
                "name":<queue_1_name>,
                "channel":"voice"
        },
        ...
        {
                "id":<queue_N_id>,
                "name":<queue_N_name>,
```

```
                "channel":"voice"
        }]
}
```

> **Important**
> Note that the filtering parameter must be exactly the same as the name of the corresponding object field.

You can also combine several filtering parameters to make even more constraints:

**Request:**

```
GET .../api/v2/system/routing-templates?fields=*&channel=voice&version=1.0.0
```

**Response:**

```
{
        "statusCode":0,
        "routingTemplates":[{
                "id":"00_RouteToSpecDestination",
                "name":"Route Call to Specified Destination",
                "description":"Routes calls to a skill or queue",
                "version":"1.0.0",
                "channel":"voice",
                "dependencies":["media", "destination"],
                "enabled":true,
                "schema": [...]
        },
        ...
        {
                "id":"07_SegmentCallerRouteToSpecDestination",
                "name":"Play Greeting, Segment Caller, and Route To Specified Destination",
                "description":"Plays a user-configured greeting, ...",
                "version":"1.0.0",
                "channel":"voice",
                "dependencies":["media", "destination", "data_record_type"],
                "enabled":false,
                "schema": [...]
        }]
}
```

> **Important**
> Note that some "list" requests may make some of the filtering parameters mandatory.

## Pagination

The following pagination-related request parameters can be used with REST API requests.

> **Important**
> Pagination and sorting functionality is only enabled if Elastic Search indexing is enabled.

| Name | Description | Request | Resources | Example |
|---|---|---|---|---|
| offset | Specifies the index of the first record to be returned.<br>• Defaults to 0. | GET | All "plural" resources | The following request will return the first 100 users in the contact center:<br>`GET /api/v2/users?offset=0&limit=100` |
| limit | Specifies the number of records to be returned.<br>• Maximum allowed value is 100.<br>• Default value is 10. | GET | All "plural" resources | The following request will return the second page of 25 users in the contact center:<br>`GET /api/v2/users?offset=25&limit=25` |

Read requests with pagination return an extra field called `totalCount` containing the total count of objects satisfying the request criteria.

```
{
        "statusCode": 0,
        "users": [...],
        "totalCount": 2
}
```

The following API resources support sorting and pagination:

- **users**
- **groups/<id>/users**
- **contacts**

## Sorting

The following sorting-related request parameters can be used with REST API requests.

| Name | Description | Request | Resources | Example |
|---|---|---|---|---|
| sortBy | Specifies a comma | GET | `/api/v2/` | |

| Name | Description | Request | Resources | Example |
|---|---|---|---|---|
| | separated list of object properties to be used for sorting. GET All "plural" resources The following request will sort users by their last names first and then by their first names: | | | `users?sortBy=lastName,firstName&limit=100` |
| order | Specifies sorting order to be used, can be either "Ascending" or "Descending", defaults to "Ascending". | GET | All "plural" resources | The following request will return users sorted by their last names in a descending order: `GET /api/v2/users?sortBy=lastName&order=Descending&li` |

## Subresources

The subresources feature allows you to read subresources of an object together with the object itself. If you have a user object that has one or more skills and one or more devices, you can read all skills and devices of that user with the following request:

**Request:**

```
GET .../api/v2/users/<user_id>?subresources=*
```

**Response:**

```
{
        "id":<user_id>,
        "firstName":<first_name>,
        ...
        "skills":[{
                "id":<skill_1_id>,
                ...
        },
        ...
        {
                "id":<skill_N_id>,
                ...
        }],
        "devices":[{
                "id":<device_1_id>,
                ...
        },
        ...
        {
                "id":<device_M_id>,
```

```
                ...
        }]
}
```

If you do not include the `subresources` parameter in the request, you will get everything except the "skills" collection and "devices" collection.

> **Important**
>
> It is also possible to apply the subresources feature to object settings and request both an object and its settings in one request.

## Selecting Subresources

In the example above, "`subresources=*`" was specified in order to get all available subresources. If the object you are interested in has several types of subresources, it is possible to choose whether you want all subresources to be returned or just some of them. This can be achieved by specifying a comma-separated list of subresources.

### Example 1

To receive a list of skills and devices associated with an agent, use the following.

**Request:**

```
GET .../api/v2/users/<user_id>?subresources=skills,devices
```

**Response:**

```
{
        "id":<user_id>,
        "firstName":<first_name>,
        ...
        "skills":[{
                "id":<skill_1_id>,
                ...
        },
        ...
        {
                "id":<skill_N_id>,
                ...
        }],
        "devices":[{
                "id":<device_1_id>,
                ...
        },
        ...
        {
                "id":<device_M_id>,
                ...
        }]
}
```

### Example 2

To receive a list of skills associated with an agent, use the following.

**Request:**

```
GET .../api/v2/users/<user_id>?subresources=skills
```

**Response:**

```
{
        "id":<user_id>,
        "firstName":<first_name>,
        ...
        "skills":[{
                "id":<skill_1_id>,
                ...
        },
        ...
        {
                "id":<skill_N_id>,
                ...
        }]
}
```

## Resolving URIs

### Introduction

This feature is called "resource link resolution", which allows you to read an object and all other objects it is associated with in one request. For example, if we have a device object associated with a phone number object and we want to read both of them in one request, we need to do the following:

**Request:**

```
GET .../api/v2/devices/<device_id>?resolveUris=*
```

**Response:**

```
{
        "id":<device_id>,
        "phoneNumberUri":"http://...",
        ...
        "phoneNumber":{
                "id":<phone_number_id>,
                ...
        }
}
```

In comparison, if you do not include the "resolveUris" parameter in the request, you will get everything except the "phoneNumber" object. In the example above, we specify "resolveUris=*" to resolve all URIs. It is possible to choose whether you want all URIs to be resolved or just some of them. This can be achieved by specifying a comma-separated list of property names referring to URIs.

## Examples

### Example 1

To resolve all URIs, use "`resolveUris=*`" as shown below.

**Request:**

```
GET .../api/v2/queues/<queue_id>?resolveUris=*
```

**Response:**

```
{
        "id":<queue_id>,
        "name":<queue_name>,
        ...
        "routingTemplateUri":"http://...",
        "phoneNumberUri":"http://...",
        ...
        "phoneNumber":{
                "id":<phone_number_id>,
                ...
        },
        "routingTemplate":{
                "id":<routing_template_id>,
                ...
        }
}
```

### Example 2

To resolve a specific URI, use "`resolveUris=<uri>`" as shown below

**Request:**

```
GET .../api/v2/queues/<queue_id>?resolveUris=phoneNumberUri
```

**Response:**

```
{
        "id":<queue_id>,
        "name":<queue_name>,
        ...
        "routingTemplateUri":"http://...",
        "phoneNumberUri":"http://...",
        ...
        "phoneNumber":{
                "id":<phone_number_id>,
                ...
        }
}
```

### Example 3

**Request:**

```
GET .../api/v2/queues/<queue_id>?resolveUris=phoneNumberUri,routingTemplateUri
```

**Response:**

```
{
        "id":<queue_id>,
        "name":<queue_name>,
        ...
        "routingTemplateUri":"http://...",
        "phoneNumberUri":"http://...",
        ...
        "phoneNumber":{
                "id":<phone_number_id>,
                ...
        },
        "routingTemplate":{
                "id":<routing_template_id>,
                ...
        }
}
```

## User Authentication

Basic HTTP Authentication is used. Please see RFC 2617 Section 2 for reference.

## Supported Requests

The following requests are supported at this time:

- /devices: fields=*
- /features: fields=*
- /me: subresources=*
- /me/calls: fields=*
- /me/devices: fields=*
- /me/skils: fields=*
- /skills: fields=*
- /system/features: fields=*
- /system/routing-templates: channel, version (these are query parameters), fields=*
- /users: fields=*, subresources=*
- /users/{id}: subresources=*
- /users/{id}/devices: fields=*
- /recordings: startTime, endTime, callerPhoneNumber, dialedPhoneNumber, userName, offset, limit (query parameters)