Web Services API Reference

Making a Request

12/13/2025

Making a Request

Contents

You can use the Web Services API to send and receive JSON-based data over HTTP. We are using cURL, which is command-line based, so you will want to open your favorite command line, terminal, or shell program, after making sure that it supports cURL. And of course, you should plug in the URL for your own Web Services server, as well as other site-specific information, when you issue the following cURL commands.

A simple request

As you might expect, your HTTP requests require a URL that contains the address of your server and the path to your Web Services API library.

> **Important**
> Ensure that the entire URL request does not exceed 2000 characters.

The rest of the URL indicates what kind of operation you would like to perform. Web Services operations are asynchronous. When a request returns `"statusCode":0`, this doesn't indicate a successful change of state — only that the request was successfully sent to T-Server.

In most cases, when you send a request you will also need to provide authentication. But you don't need authentication to ask for your current version of Web Services. To do this, type in the following cURL command:

```
curl http://000.111.222.333/api/v2/diagnostics/version
```

The above request will return something like this:

```
{"statusCode":0,"version":"8.5.200.50"}
```

[+] Click here to see other ways you can retrieve the Web Services version.

Instead of using cURL, you can also get the version using JavaScript, a REST client, or a web browser.

JavaScript

```
<!doctype html>
<html>
        <head>
                <script src='//ajax.googleapis.com/ajax/libs/jquery/1.11.1/
jquery.min.js'></script>
                <script>
                        $(document).ready(function() {

                                // Add a click handler to the getVersion button.
                                $('#getVersion')
                                .click(function() {

                                    // Create and configure the request.
```

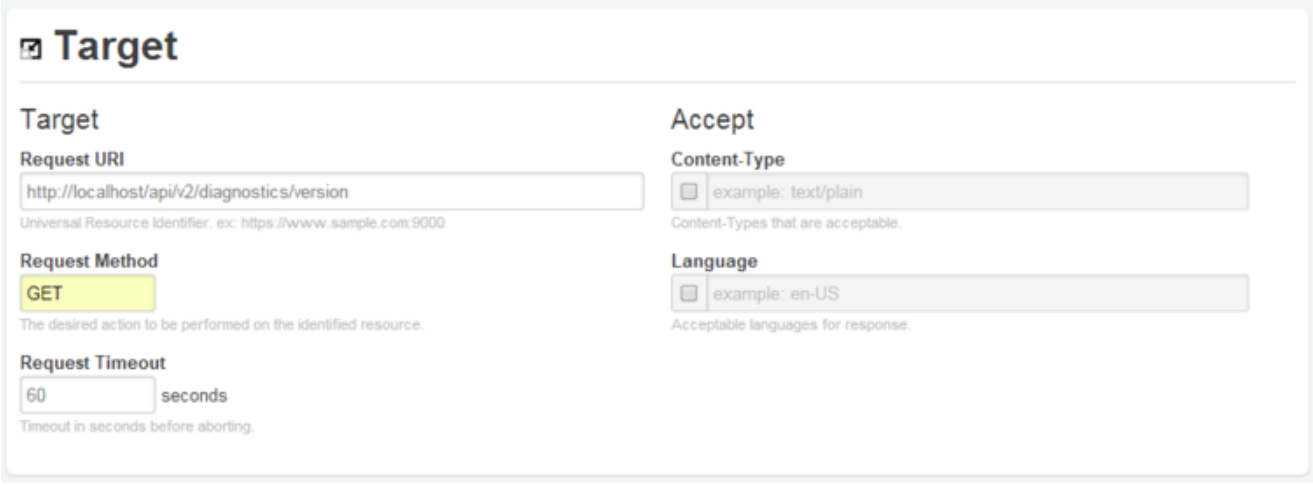
```
                                    var request = {
                                        url: 'http://localhost:8080/api/v2/diagnostics/
version',
                                        type: 'GET', crossDomain: true, success: function
(result) {
                                                // Update the label with the result.
                                            $('#version').text(result.version);
                                        },
                                        error: function (result) {
                                        alert('Failed to get version.');
                                        }
                                    };

                                $.ajax(request);
                                });
                        });
                </script>
        </head>
        <body>
             <div>
                <button id='getVersion'>Get Version</button>
                <br/>
                <label id='version'>-</label>
            </div>
        </body>
</html>
```

Response

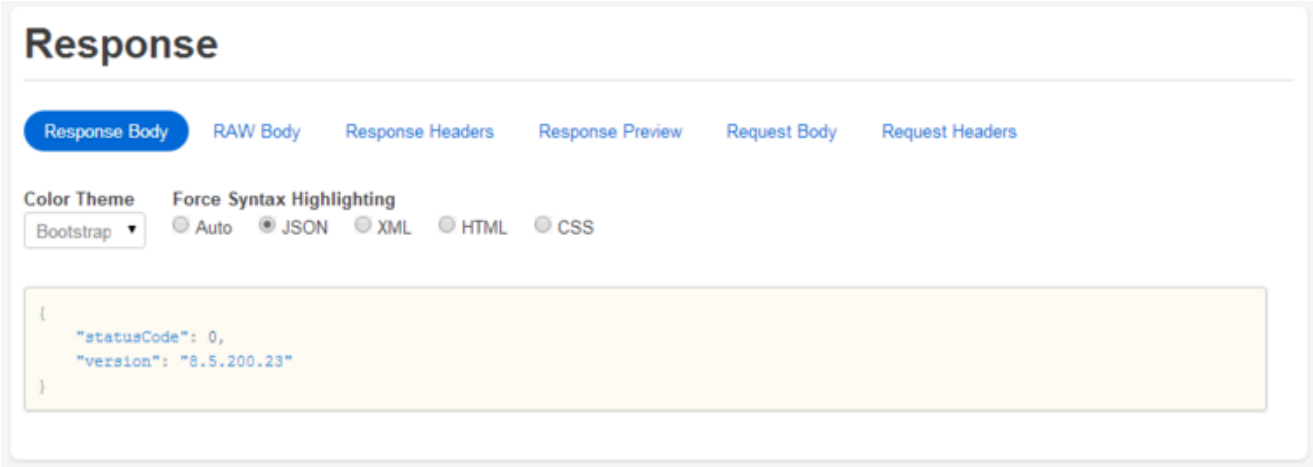
```
{
    "statusCode":0,
    "version":"8.5.200.23"
}
```

REST client

Instead of writing a client application to test your API calls, you can use a REST client embedded in your web browser.



Response



Web browser

This call is the only REST API call you can make in a web browser because it doesn't require authentication. All you need to do is navigate to the following URL: `http://WS_Server:WS_Port/api/v2/diagnostics/version`

Where `WS_Server` is the IP of your Web Services node and `WS_Port` is its port.

Response

```
{"statusCode":0,"version":"8.5.200.23"}
```

Authentication

The following request asks for information about user `ksippo`. Like most Web Services requests, this one requires authentication. cURL allows us to specify the user name and password by using the format `-u` **`username:password`**.

The user mentioned in the following request does not have a password, so we have left the password field empty:

```
curl -u ksippo: http://000.111.222.333/api/v2/me
```

The response from the Web Services server should look something like this:

```
{
        "statusCode":0,
        "user":{
                "id":"63630bbebf4840d7a0bffd6312bc29ff",
                "userName":"ksippo",
                "firstName":"Kristi",
                "lastName":"Sippola",
                "roles":["ROLE_AGENT"],
                "enabled":true,
                "changePasswordOnFirstLogin":false,
                "uri":"http://127.0.0.1/cloud-web/api/v2/users/
```

```
63630bbebf4840d7a0bffd6312bc29ff",
                "path":"/users/63630bbebf4840d7a0bffd6312bc29ff"
        }
}
```

Sending data

Sending data is a bit more complex. We use a POST request and indicate to cURL that we are sending data in JSON format. We also use a URL that tells the Web Services server to carry out an operation for the current user, `ksippo`.

Finally, the following request uses the cURL data parameter, `-d`, to carry the JSON payload, which lets the server know that we want to set ksippo's status to `NotReady`.

```
curl -X POST -H "Content-Type: application/json" -d '{"operationName":"NotReady"}' -u ksippo: http://000.111.222.333/api/v2/me/channels/
voice
```

If we did everything right, we will get confirmation from the server by way of a status code of `0`:

```
{"statusCode":0}
```

Filtering a request

You may also want to get specific information associated with an agent or other user, such as a list of their skills or devices. To do this, you can filter your request, as shown in the Subresources topic.

What's next?

Now that we have an idea of how to send requests, let's take a look at how to interpret responses from the Web Services server.