Web Services API Reference

Interpreting a response

5/4/2025

Interpreting a response

Contents

On the previous page, we showed you how to use cURL to make some basic requests. Now we will show you how to interpret the response from the Web Services server.

Determining your Web Services version

The first request we sent asked for the current version of Web Services. It looked like this:

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

The following response starts with a status code of `0`, which indicates that our request was successful. The `version` parameter, surprisingly enough, tells you what your current version of Web Services is:

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

Web Services status codes

When you are trying to figure out what happened with your request, you may find it helpful to understand the status codes that have been returned by the Web Services server. These codes are described in a table on the Return Values page.

Note in particular that a status code of `20` means that you have failed authentication, as we will show in the next section.

Getting user information

The second request we sent on the previous page asked for information about user `ksippo`. This request included authentication information, as shown here:

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

This request should receive a status code of `0`, followed by user information about ksippo:

```
{
        "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"
        }
}
```

Don't forget...

Note that if we send that request without including authentication information, we will receive an error message. That is, if we send something like this:

```
curl http://000.111.222.333/api/v2/me
```

...the Web Services server won't let us in, and will send a response like this:

```
{"statusCode":20,"statusMessage":"Access denied"}
```

As mentioned above, the status code of 20 indicates that we have failed authentication and are therefore denied access, as clarified by the status message.

[+] Click here to see other ways you can retrieve user information.

Instead of using cURL, you can also get user information using JavaScript or a REST client.

JavaScript

```
<!--
        This sample improves on the version.html sample by making the server location
configurable
        and also allowing credentials to be entered. To establish a session with Genesys Web
Services,
        a Basic Authorization header must be included. Subsequent requests can also include
this header
        or they can rely on the cookie established by the first request.

        The sample request this time is to return basic information about 'me' (the user
making the request as
        identified by the credentials).
-->
<!doctype html>
<html>
        <head>
                <script src='//ajax.googleapis.com/ajax/libs/jquery/1.11.1/
jquery.min.js'></script>
                <script>
                        $(document).ready(function() {

                                $('#getMe')
                                .click(function() {

                                        // Read the values from the input boxes
                                        var username = $('#username').val();
                                        var pw = $('#password').val();

                                        var uri = $('#baseUri').val();
                                        uri += 'api/v2/me';

                                        // Create and configure the request
                                        var request = {
                                   url: uri,
                                   type: 'GET',
                                                crossDomain: true,
                                   success: function (result) {
                                       // Update the textarea with a string version of the
resulting JSON
```

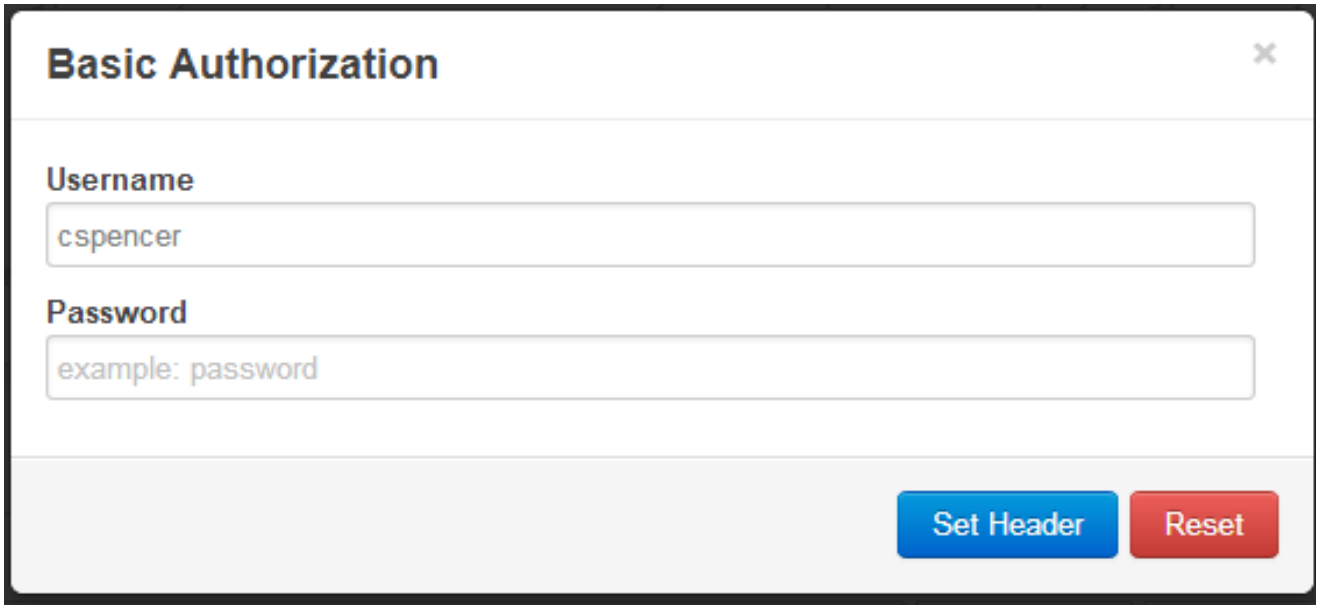
```
                                        $('#result').text(JSON.stringify(result.user, null,
4));
                                    },
                                    error: function (result) {
                                        alert('Failed to get my user info.');
                                    }
                                };

                                // This adds the Authorization header. The call to btoa base
64 encodes the username and
                                // password separated by a ':'. For more info on Basic
authentication check the RFC.
                                request.beforeSend = function (xhr) {
                                        xhr.setRequestHeader('Authorization', 'Basic
' + window.btoa(username + ':' + pw));
                                 };

                                $.ajax(request);
                                 });
                        });
                </script>
        </head>
        <body>
                <div>
            <input id='baseUri' type='text' style='margin-bottom: 5px; width: 200px;'
placeholder='GWS Base Uri' value='http://localhost:8080/'>
            <br/>
            <input id='username' type='text' style='margin-bottom: 5px; width: 200px;'
placeholder='Username' value='paveld@redwings.com'>
            <br/>
            <input id='password' type='password' placeholder='Password' value='password'>
            <br/>
                        <button id='getMe'>Get Me</button>
                        <br/>
                        <br/>
                        <textarea id='result' rows='20' cols='100'></textarea>
                </div>
        </body>
</html>
```

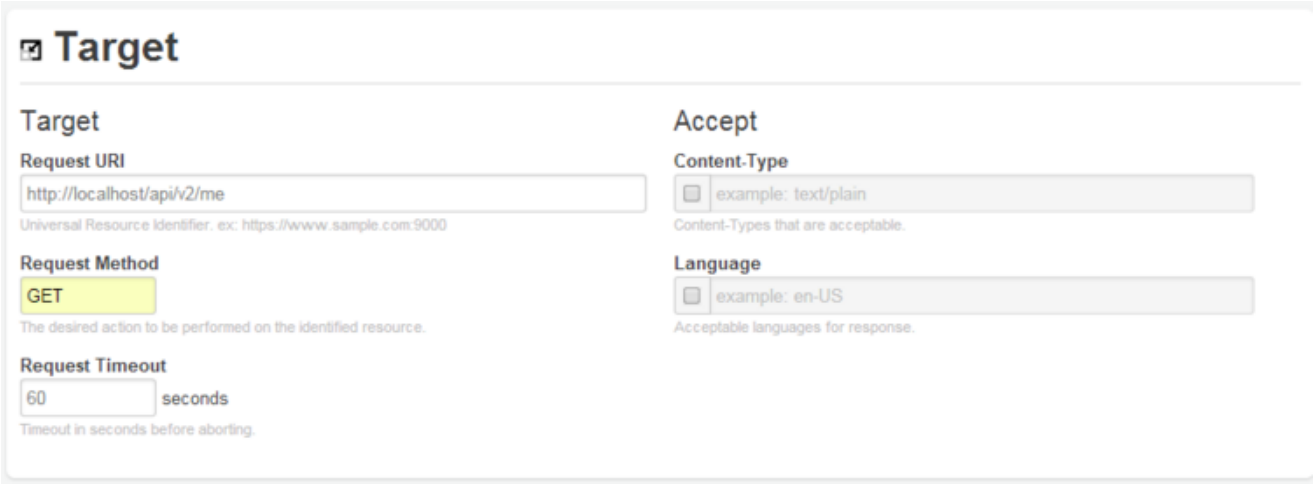
REST client

To retrieve agent information, we must authenticate, as shown here (in our case, `cspencer` is the user and there is no password associated with her account):

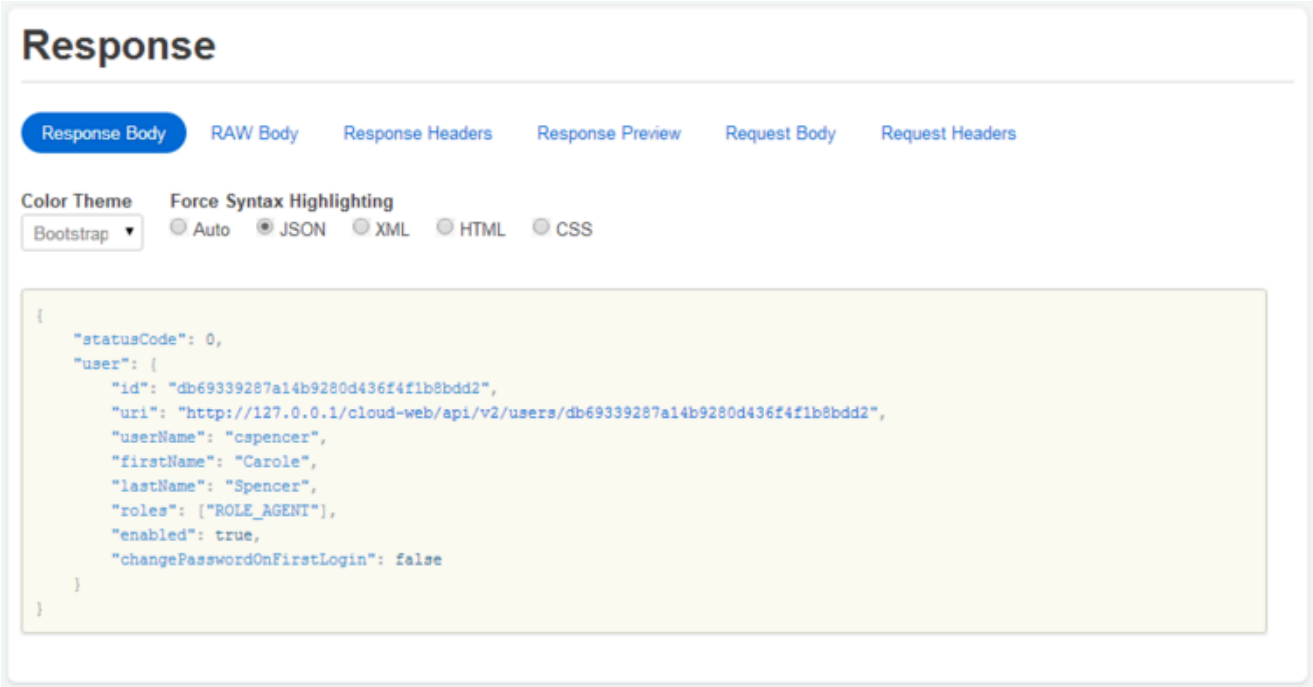


Choose `Set Header`, and you should see something similar to the following:

Request Now that you've been authenticated, you can make the request:



Response



What's next?

Next up, let's learn how to work with agents.