Using Custom JS to Make an HTTP Request

Last Modified on 02/01/2021 4:41 pm EST

On this page

When to use HTTP request from your common object

Before we jump into when to use HTTP requests within your common object, we should keep in mind that common resources work dynamically, and therefore, every single call going through a common object will perform the extra HTTP call.

That being said, an important factor to keep in mind is performance. You should ideally use HTTP requests within common resources when you are okay with the extra added time to the resource call. You are likely not going to notice a huge difference during calls performed via the common object with the extra HTTP call, however, it is important to be aware of the possibility prior to its implementation.

Now, let's address when you should perform a HTTP request from a common object. Ideally, you will perform an additional HTTP call when you have a unique identifier available to be used in the subsequent call. Let's look into a real world example:

Let's say I am calling a resource (for example, Vendors) that returns a payload like the example below:

```
{

 "vendorId": 1,

 "vendorName": "Vendor Demo",

 "VendorAddress": "001 1st street, Denver, Colorado - 80022",

 "contactInformation": {

  "contactId": 100

 }

}
```

In addition to the vendor's information, now you want to retrieve their contact info as well (such as, phone numbers, fax, etc..). As you can see from our example above, the "Vendors" payload only returns a `contactId`. The great news here is that we can use that `contactId` to perform a subsequent call to retrieve the contact information and add it to the final payload that will be returned when calling that common object object.

How to make use of HTTP requests from Common Objects

To be able to perform HTTP calls from a VDR, we will need to first *require('https')*. If you are not familiar with node's native HTTP server/client, please feel free to visit Node's documentation page here.

```
let https = require('https')
```

Here is what the final example JS Script would look like:

```
let https = require('https');


let options = {
  host: "snapshot.cloud-elements.com",
  port: 443,
  path: "/elements/api-v2/ping",
  method: "GET",
  "headers": {
  "accept": "application/json",
  "Authorization": `User ${configuration.userSecret}, Organization ${configuration.organizationSecr
et}, Element ${configuration.elementInstanceToken}`
 }
};

let data = "";
let request = https.request(options, function(res) {
  res.on('data', function(d) {
    data += d;
  }).on('end', function(d) {
    if (res.statusCode === 200) {
      transformedObject.ping = JSON.parse(data);
      done(transformedObject);
    } else {
          transformedObject.ping = { message: "https call to ping failed", error: res.statusMessage
 };
          done(transformedObject);
    }
  });
});

request.on('error', function(e) {
  transformedObject.ping = { message: "https call to ping failed", error: JSON.stringify(e) };
  done(transformedObject);
});

request.end();
```

How to pass in your Org, User, and Connector API keys

As you can see in the final example provided above, we are passing within the "headers" our "Authorization" tokens (i.e. User Token, Organization Token, and **Connector** Token):

```
   "Authorization": `User ${configuration.userSecret}, Organization ${configuration.organizationSe
 cret}, Element ${configuration.elementInstanceToken}`
```

We have setup a common object to support dynamic resource calls, meaning, you do not have to hard code the Organization, User, or **Connector** token. All these items will be retrieved from the **"configuration"** dynamically when the a

call is performed to the common object object.

General Concept and Knowledge

Going back to our original example (Vendors), now that you have the HTTP library all set up and you are able to perform additional API calls within your common object, you can use the "transformedObject" concept (more information here) to replace the "contactInformation" object original returned (below):

```
"contactInformation": {
  "contactId": 100
 }
```

For something more useful such as this example below:

```
 "contactInformation": {

   "phone": "+1 (303) 333-1122",

   "fax": "+1 (303) 333-2211"

}
```