
regoUniversity
NASHVILLE · 2022
Using REST APIs
Your Guides:
Ben Rimmasch & James Gille

Introductions

- Take 5 Minutes
- Turn to a Person Near You
- Introduce Yourself
- Business Cards

Agenda

- Overview
  - Why REST now?
- Clarity REST Endpoints
  - Create Project with java.net.URL
  - Create Task with HttpClient
  - Update Project with java.net.URL
  - Update Task with HttpClient
  - Create Task with RestApiJavaClient
- External REST Endpoints
  - Connect with java.net.url
  - Connect with Apache Commons HTTP/S Client
- Additional Tools

Part I: Overview

Overview

- Choosing the correct approach can cut down on development, maintenance, and debugging time.
- The java.net.url needs workarounds for modern HTTP verbs.
- The Apache Commons HTTP and HTTPS clients will potentially be the best upgrade path for an internal java client that Broadcom may provide.

Why REST Now?

- Broadcom finally officially supports customer and partner use of the Clarity REST API
- The REST API is built out sufficiently that most of the data that needs to be manipulated can be done without needing to fallback on the SOAP API
- Some data can only be updated via the REST API (primarily related to the new UI)
  - To determine what your version of Clarity supports you can check Broadcom's support website or Clarity's self-documenting REST client
  - https://[hostname]/niku/rest/describe/index.html

Part II:
Clarity REST Endpoints

Create Project java.net.URL

Clarity REST Endpoints – Base URL

- Programmatically retrieving the base URL can be done in multiple ways. Start with a ConfigurationManager.

```
<core:invokeStatic var="config"
  className="com.niku.union.config.ConfigurationManager"
  method="getInstance" />
```

- The scheduler URL can be used

```
<core:set var="baseUrl" className="${config.getProperties().getWebServer().getSchedulerUrl()}" />
```

- Or the Rest API Entry URL

```
<core:set var="baseUrl" className="${config.getRestAPIEntryURL()}" />
```

Clarity REST Endpoints - Create with java.net.URL

- Create a project using a java.net.URL

```
<core:set var="restResource" value="https://clarity.bendy.me/ppm/rest/v1/projects" />
<core:new var="restUrl" className="java.net.URL" >
  <core:arg type="java.lang.String" value="${restResource}" />
</core:new>
<core:invoke var="restConnection" on="${restUrl}" method="openConnection" />
```

Clarity REST Endpoints - Create with java.net.URL

- For this example, we are connecting to the same Clarity instance where the GEL script is running so we can utilize authorization without a password.

```
<core:set var="userName" value="admin" />
<core:invokeStatic
  var="restSessionController"
  className="com.niku.union.security.UserSessionControllerFactory"
  method="getInstance" />
<core:invoke var="restSession" on="${restSessionController}" method="init">
  <core:arg type="java.lang.String" value="${userName}" />
</core:invoke>
```

Clarity REST Endpoints - Create with java.net.URL

- Set headers with the authorization data

```
<core:expr value="${restConnection.setRequestProperty('Authorization', 'authToken ' +
restSession.getSessionId())}" />

<core:expr value="${restConnection.setRequestProperty('Cookie', 'sessionId=' +
restSession.getSessionId())}" />
```

Clarity REST Endpoints - Create with java.net.URL

- Set the REST verb to POST since that is what project creation requires

```
<core:expr value="${restConnection.setRequestMethod('POST')}" />
<core:expr value="${restConnection.setRequestProperty('Content-Type',
'application/json')}" />
<core:expr value="${restConnection.setDoOutput(true)}" />
```

Clarity REST Endpoints - Create with java.net.URL

- Create the JSON request body

```
<core:set var="restBody" encode="false">
  <![CDATA[{
  "name": "A Rego University Project",
  "isActive": "true"
}]]>
</core:set>
```

Clarity REST Endpoints - Create with java.net.URL

- Convert the request String to bytes and write it to the connection

```
<core:invoke var="restOs" on="${restConnection}" method="getOutputStream" />
<core:invoke var="requestBytes" on="${restBody}" method="getBytes">
  <core:arg type="java.lang.String" value="utf-8" />
</core:invoke>

<core:expr value="${restOs.write(requestBytes, 0, size(requestBytes))}" />
```

Clarity REST Endpoints - Create with java.net.URL

- It’s time to check the response. One of two streams will have data for us.

```
<core:choose>
  <core:when test="${restConnection.getResponseCode() == 200}" >
    <core:invoke var="is" on="${restConnection}" method="getInputStream" />
  </core:when>
  <core:otherwise>
    <core:invoke var="is" on="${restConnection}" method="getErrorStream" />
  </core:otherwise>
</core:choose>
```

Clarity REST Endpoints - Create with java.net.URL

- Let's read it into a StringBuffer.

```
<core:new var="isr" className="java.io.InputStreamReader">
  <core:arg type="java.io.InputStream" value="${is}" />
</core:new>
<core:new var="br" className="java.io.BufferedReader">
  <core:arg type="java.io.Reader" value="${isr}" />
</core:new>
<core:new var="response" className="java.lang.StringBuffer" />
<core:set var="nextLine" value="${br.readLine()}" />
<core:while test="${nextLine != null}">
  <core:expr value="${response.append(nextLine)}" />
  <core:set var="nextLine" value="${br.readLine()}" />
</core:while>
<core:expr value="${br.close()}" />
```

Clarity REST Endpoints - Create with java.net.URL

- Now let's convert that to a Google GSON object. This is a really powerful way to manipulate the response.

```
<core:new var="jsonParser" className="com.google.gson.JsonParser" />
<core:set var="jsonElement" value="${jsonParser.parse(response.toString())}" />
<core:set var="prjGson" value="${jsonElement.getAsJsonObject()}" />
```

Clarity REST Endpoints - Create with java.net.URL

- Once the data is in a GSON object you can reference into it really easy. For example, consider the following JSON stored in the variable “gson”:

```
{
  "_internalId": 5025045,
  "_self": "https://clarity.bendy.me/ppm/rest/v1/projects/5025045",
  "name": "A Rego University Project",
  "isActive": "true"
}
```

Clarity REST Endpoints - Create with java.net.URL

- The output of ${gson._internalId}:

```
<gel:log level="INFO">Project created with id: ${gson._internalId}</gel:log>
```

Start	Script		Result: 200 OK
Start	Script		Project created with id: 5025046
Start	Script		Total Elapsed time: 625 (ms)

Clarity REST Endpoints - Create with java.net.URL

- If you need to work with the data additionally in Java and a specific data type is required methods are available to convert the value.
  - `${gson._internalId.getAsInt()}`
  - `${gson._internalId.getAsLong()}`
  - `${gson._internalId.getAsString()}`
- For a full list see the documentation.
  - https://javadoc.io/doc/com.google.code.gson/gson/2.8.5/com/google/gson/JsonElement.html

Create Task HttpClient

Clarity REST Endpoints - Create with HttpClient

- Create a task using an Apache Commons HTTPS Client

```
<core:invokeStatic var="httpClient" className="org.apache.http.impl.client.HttpClients"
method="createDefault" />

<core:new var="httpRequest" className="org.apache.http.client.methods.HttpPost" />

<core:expr value="${httpRequest.addHeader('Content-type', 'application/json')}"/>
```

Clarity REST Endpoints - Create with HttpClient

- Add the authorization headers

```
<core:expr value="${httpRequest.addHeader('Authorization', 'authToken ' + restSession.getSessionId())}"/>
<core:expr value="${httpRequest.addHeader('Cookie', 'sessionId=' + restSession.getSessionId())}"/>
```

Clarity REST Endpoints - Create with HttpClient

- Configure the create task endpoint

```
<core:new var="uriBuilder" className="org.apache.http.client.utils.URIBuilder" >
  <core:arg type="java.lang.String" value="${baseUrl}"/>
</core:new>
<core:invoke method="setPath" on="${uriBuilder}">
  <core:arg type="java.lang.String" value="/ppm/rest/v1/projects/${prjGson._internalId}/tasks"/>
</core:invoke>
<core:expr value="${httpRequest.setURI(uriBuilder.build())}"/>
```

Clarity REST Endpoints - Create with HttpClient

- Create the request body and add it to the request

```
<core:set var="taskJson" encode="false">
  <![CDATA[{
  "name": "A New Task",
  "startDate": "2021-05-28T08:00:00",
  "finishDate": "2021-05-28T17:00:00"
}]]>
</core:set>
<core:new var="jsonEntity" className="org.apache.http.entity.StringEntity" >
  <core:arg type="java.lang.String" value="${taskJson}"/>
</core:new>
<core:expr value="${jsonEntity.setContentType('application/json')}"/>
<core:expr value="${httpRequest.setEntity(jsonEntity)}"/>
```

Clarity REST Endpoints - Create with HttpClient

- Execute the call

```
<core:invoke var="httpResponse" on="${httpClient}" method="execute"  >
  <core:arg type="org.apache.http.client.methods.HttpPost" value="${httpRequest}"/>
</core:invoke>
```

Clarity REST Endpoints - Create with HttpClient

- Check the response

```
<core:choose>
  <core:when test="${httpResponse.getStatusLine().getStatusCode() == 200}" >
    <gel:log level="DEBUG">Result: 200 OK</gel:log>
  </core:when>
  <core:otherwise>
    <gel:log level="WARN">Result: ${httpResponse.getStatusLine().getStatusCode()}
${httpResponse.getStatusLine().getReasonPhrase()}</gel:log>
  </core:otherwise>
</core:choose>
```

Clarity REST Endpoints - Create with HttpClient

- Read the response into a String

```
<core:new var="isr" className="java.io.InputStreamReader" >
  <core:arg type="java.io.InputStream" value="${httpResponse.getEntity().getContent()}"/>
</core:new>
<core:new var="br" className="java.io.BufferedReader">
  <core:arg type="java.io.Reader" value="${isr}" />
</core:new>
<core:new var="response" className="java.lang.StringBuffer" />
<core:set var="nextLine" value="${br.readLine()}" />
<core:while test="${nextLine != null}">
  <core:expr value="${response.append(nextLine)}" />
  <core:set var="nextLine" value="${br.readLine()}" />
</core:while>
<core:expr value="${br.close()}" />
```

Clarity REST Endpoints - Create with HttpClient

- Convert the String into a Google GSON object

```
<core:new var="jsonParser" className="com.google.gson.JsonParser" />
<core:set var="jsonElement" value="${jsonParser.parse(response.toString())}" />
<core:set var="taskGson" value="${jsonElement.getAsJsonObject()}" />
```

Update Project java.net.URL

Clarity REST Endpoints - Update with java.net.URL

- To update a project, change the endpoint

```
<core:set var="restResource" value="/ppm/rest/v1/projects/${prjGson._internalId}" />
```

- Since java.net.URL does not support the PATCH HTTP verb Clarity has a workaround

```
<core:expr value="${restConnection.setRequestMethod('PUT')}" />
<core:expr value="${restConnection.setRequestProperty('x-api-force-patch', 'true')}" />
```

Clarity REST Endpoints - Update with java.net.URL

- Change the name and active flag

```
<core:set var="restBody" encode="false">
  <![CDATA[{
  "name": "A Rego University Project (OLD)",
  "isActive": "false"
}]]>
</core:set>
```

Update Task HttpClient

Clarity REST Endpoints - Update with HttpClient

- To update a task, change the endpoint

```
<core:invoke method="setPath" on="${uriBuilder}">
  <core:arg type="java.lang.String" value="/ppm/rest/v1/projects/${prjGson._internalId}/tasks/${taskGson._internalId}"/>
</core:invoke>
<core:expr value="${httpRequest.setURI(uriBuilder.build())}"/>
```

- Modern versions of HttpClients have a PATCH verb implementation

```
<core:new var="httpRequest" className="org.apache.http.client.methods.HttpPatch" />
```

Clarity REST Endpoints - Update with HttpClient

- Change the name and task dates

```
<core:set var="taskJson" encode="false">
  <![CDATA[{
  "name": "An Old Task",
  "startDate": "2021-08-28T08:00:00",
  "finishDate": "2021-08-28T17:00:00"
}]]>
</core:set>
```

Create Task RestApiJavaClient

Clarity REST Endpoints - RestApiJavaClient

DISCLAIMER

- Broadcom does not officially support the use of its Java classes for any given purpose. However, they accept that some of their Java classes are widely-used (getting sessions, configuration manager, etc.) and greatly reduce complexity and development effort in some implementations.
- **The Rest API Java Client has known issues and limitations**. Rego has made Broadcom aware that it is using the RestApiJavaClient and has also made Broadcom aware of many bugs that affect it. Most of the bugs have no workaround and make its use impossible when affected by those bugs.
- In general, the stability of this approach is not good enough that we recommend its use. However, over time we hope to see it improve and become the de facto approach given its extreme ease of use and concise setup.

Clarity REST Endpoints - RestApiJavaClient

- Create the Rest API Java Client

```
<core:new var="javaRestClient" className="com.ca.ppm.rest.client.RestApiJavaClient">
  <core:arg type="java.lang.String" value="${baseUrl}/ppm" />
  <core:arg type="java.lang.String" value="${userName}" />
</core:new>
```

Clarity REST Endpoints - RestApiJavaClient

- Create the task JSON

```
<core:set var="taskJson" encode="false">
  <![CDATA[{
  "name": "A Different New Task",
  "startDate": "2021-06-01T08:00:00",
  "finishDate": "2021-06-01T17:00:00"
}]]>
</core:set>
```

Clarity REST Endpoints - RestApiJavaClient

- Execute the POST

```
<core:invoke var="createTaskResponse" on="${javaRestClient}" method="doPost">
  <core:arg type="java.lang.String" value="/projects/${prjGson._internalId}/tasks" />
  <core:arg type="java.lang.String" value="${taskJson}" />
</core:invoke>
```

Clarity REST Endpoints - RestApiJavaClient

- Check the response

```
<gel:log level="INFO">Success?: ${createTaskResponse.wasSuccessful()}</gel:log>
<core:set var="taskDbid" value="${createTaskResponse.getObject()._internalId.getAsInt()}" />
<gel:log level="INFO">Task internal ID: ${taskDbid}</gel:log>
```

Part III:
External REST Endpoints

External REST Endpoints

- In most situations the same approach that was used for Clarity (internal) REST endpoints can be used for external endpoints e.g., the java.net.URL and Apache Commons HTTP Client.
- One deviation will be the way Clarity handles the PATCH HTTP verb compared to the many other applications. A common implementation for many applications is to use a PUT HTTP verb with the following header.

```
<core:expr value="${restConnection.setRequestProperty('x-http-method-override', 'PATCH')}" />
```

External REST Endpoints

- Another probable difference will be authentication. Clarity has Java classes to generate a session with only a username. When calling external applications if the calls are authenticated you must use an endpoint to generate a session or a token with a username and password or API key.

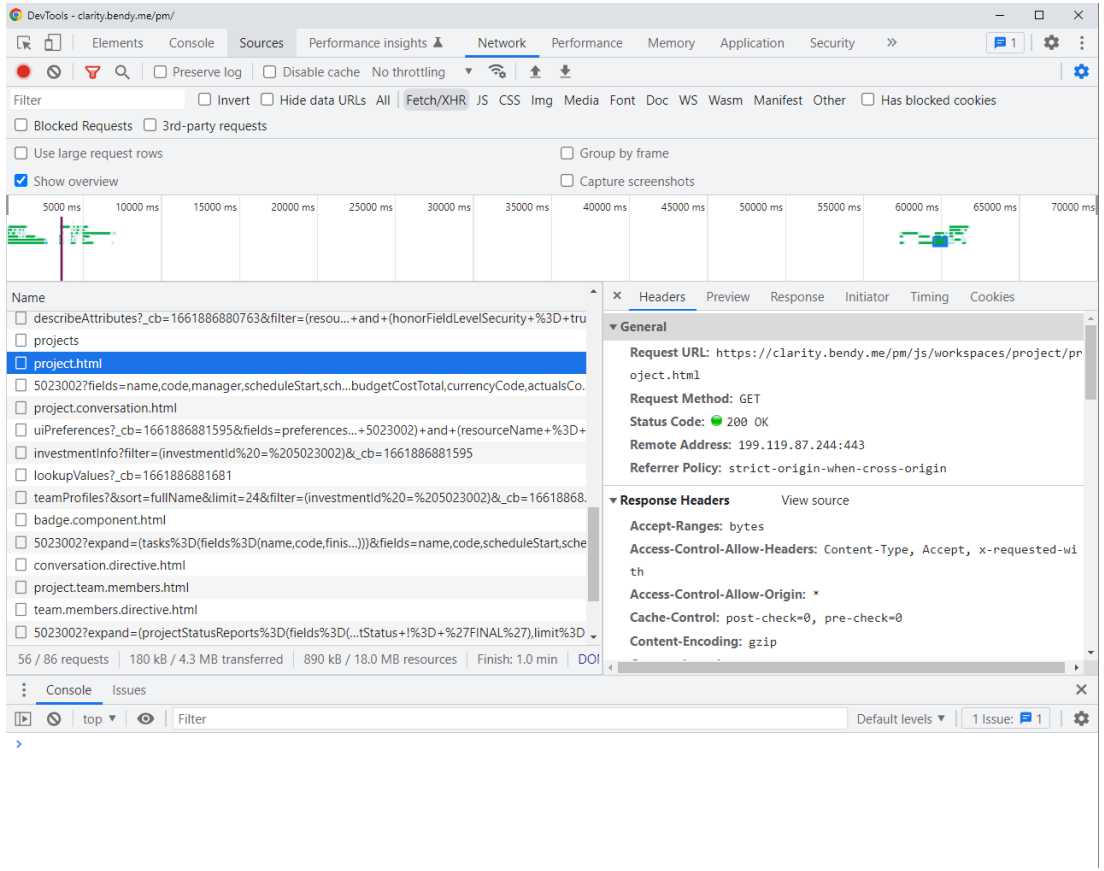
Part IV: Additional Tools

Rego Data Processor

Rego's Data Processor is capable of REST interaction. There is a separate slide deck and class to go over this functionality.

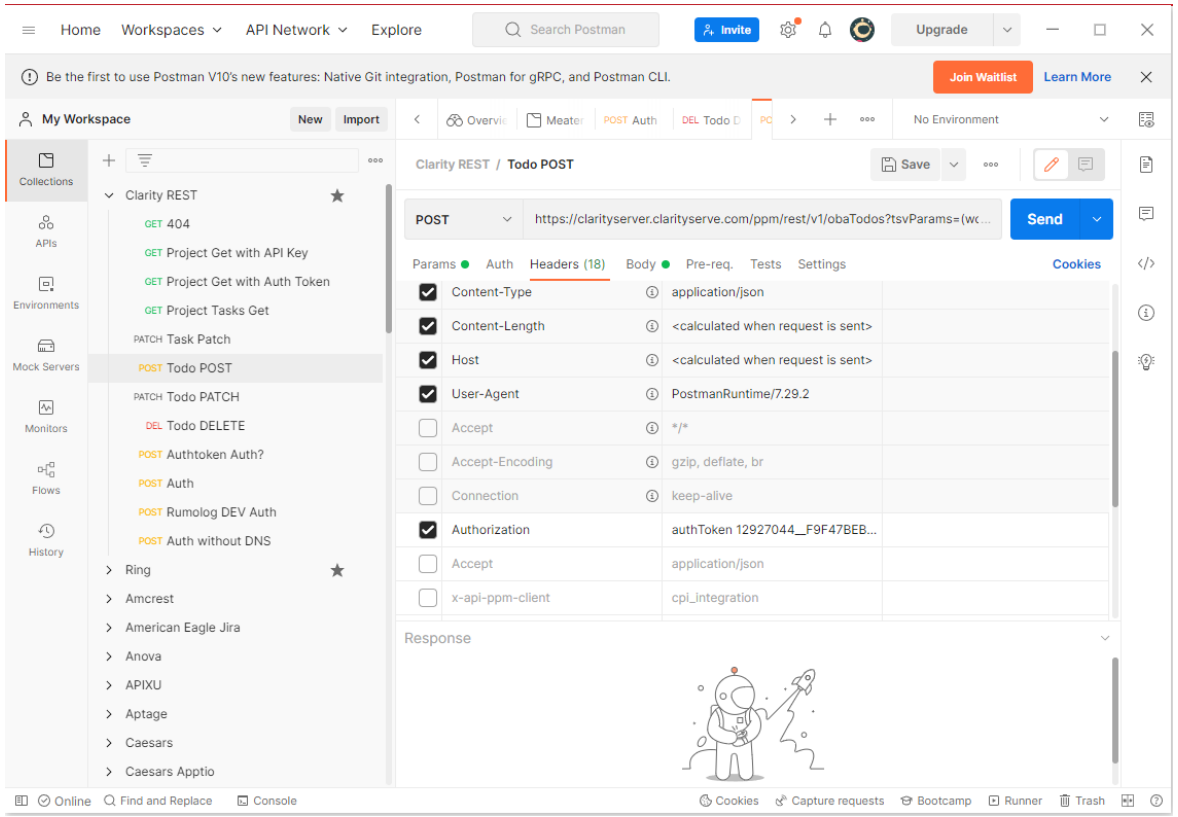
Chrome Developer Tools

- Allows you to view all REST API calls that the MUX performs when navigating and saving



Postman

- Allows you to view mock-up and test API calls



Questions?

Thank You For Attending regoUniversity

Instructions for PMI credits

- Access your account at pmi.org
- Click on **Certifications**
- Click on **Maintain My Certification**
- Click on **Visit CCR's** button under the **Report PDU's**
- Click on **Report PDU's**
- Click on **Course or Training**
- Class Provider = **Rego Consulting**
- Class Name = **regoUniversity**
- Course **Description**
- Date Started = **Today's Date**
- Date Completed = **Today's Date**
- Hours Completed = **1 PDU per hour of class time**
- Training classes = **Technical**
- Click on **I agree** and **Submit**

Let us know how we can improve!
Don't forget to fill out the class survey.

Phone
888.813.0444

Email
info@regoconsulting.com

Website
www.regouniversity.com