
regoUniversity
KANSAS CITY • 2024
Sponsored by
ValueOps by Broadcom
Clarity by Broadcom
Rally by Broadcom
ConnectALL by Broadcom
Insights by Broadcom
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
- External Tools
- Best Practices
- Jam Tags
- Clarity REST Examples

Part I: Overview

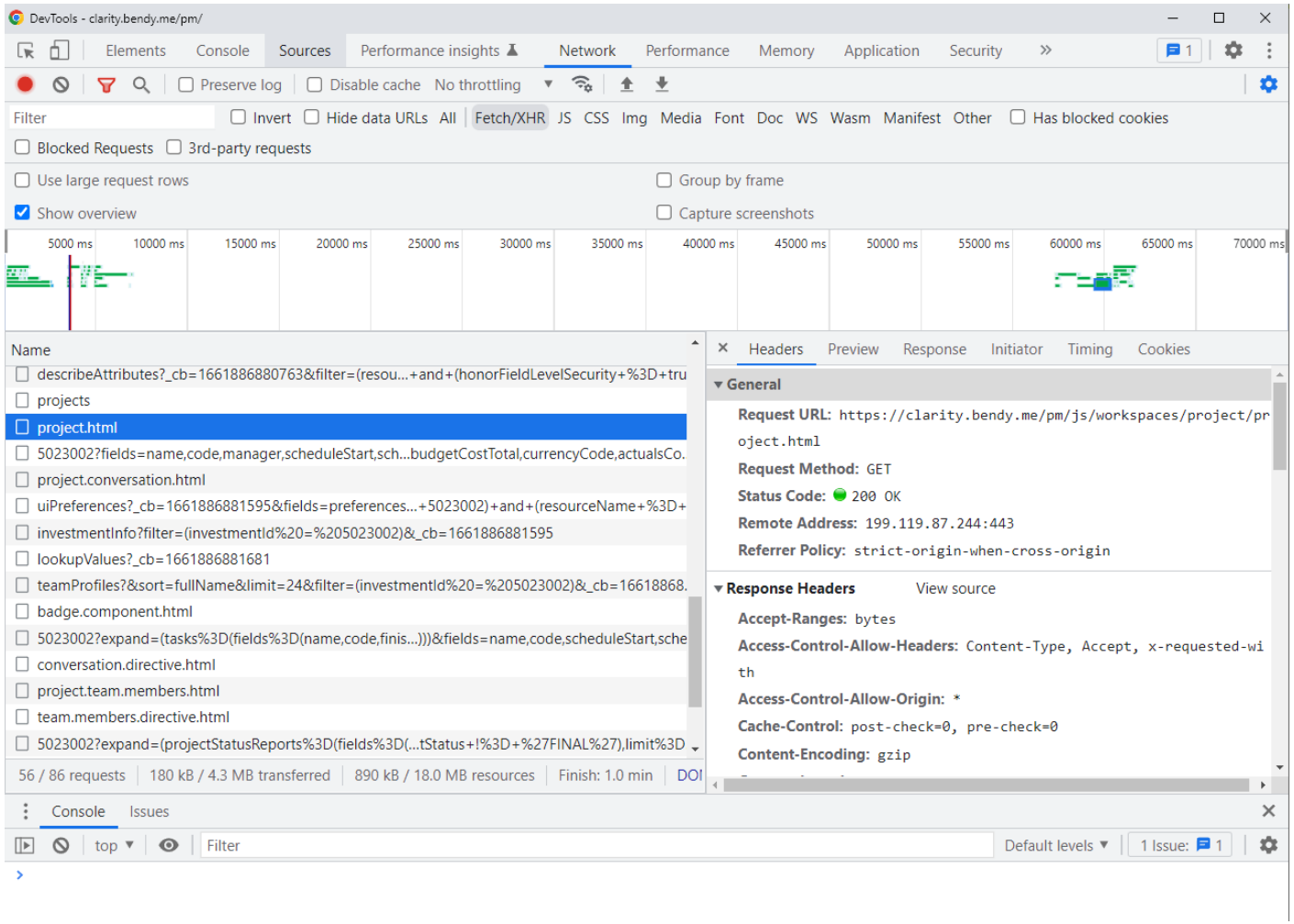
Overview

- The REST API is built out sufficiently that most of the data that needs to be manipulated can be done without needing to fall back on the SOAP API
- Some new features can cause the SOAP API to break (Cost Plans with multiple rows per grouping / data entered in different fiscal periods)
- 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 I:
External Tools

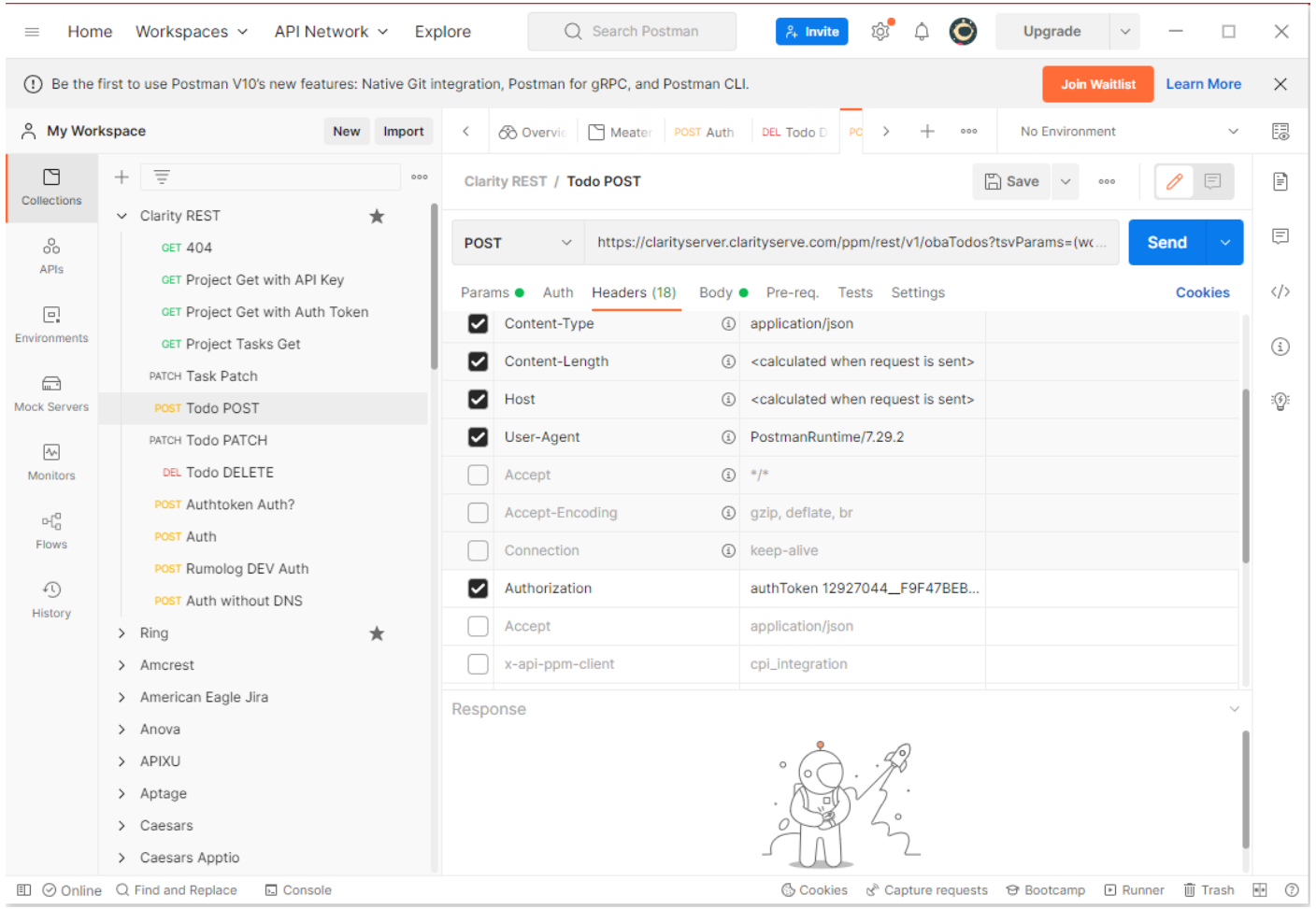
Chrome Developer Tools

- 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



Part III:
Best Practices

Best Practices

- Generate JSON in a readable way
- Escape illegal JSON characters
  - Backspace to be replaced with \b (Probably not an issue in Clarity)
  - Form feed to be replaced with \f (Probably not an issue in Clarity)
  - Newline to be replaced with \n
  - Carriage return to be replaced with \r
  - Tab to be replaced with \t
  - Double quote to be replaced with \"
  - Backslash to be replaced with \\
- Multiple ways to handle illegal JSON
  - Java (String Replace or JSONObject)
  - SQL: REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(i.name,'\','\\'),'"','\"'),CHR(10),'\n'),CHR(13),'\r'),CHR(9),'\t') (Oracle)
  - Jam Tags

Best Practices - Clarity REST URL

- Programmatically retrieve the base URL with a ConfigurationManager.

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

- Use the Rest API Entry URL

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

- Define URL for REST call

```
<core:set var="restResource" encode="false" escapeText="false">
<![CDATA[${baseUrl}/rest/v1/projects]]>
</core:set>
```

Best Practices - External REST Endpoints

- Applications handle authentication in several different ways:
  - Basic Auth: Username and Password
  - Token
  - Oauth
- Credentials should be stored in a way that is secure
  - Rego's Connection Manager encrypts passwords, tokens, and certificates

Best Practices - External REST Endpoints

- Three common methods for calling REST endpoints
  - java.net.URL
  - Apache Commons HTTP Client.
  - Jam Tags
- Applications handle the HTTP PATCH verb in different ways. 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')}" />
```

Part III:
Jam Tags

Jam Tags - Namespaces

- New namespace for Clarity specific tags

```
<gel:script xmlns:clarity="jelly:com.rego.jam.ClarityTagLibrary" >
```

- New namespace for REST specific tags
- `<gel:script xmlns:rest="jelly:com.rego.jam.RestTagLibrary" >`

Jam Tags - REST Auth

- clarity:restAuth

```
<clarity:restAuth var="restAuth" userName="admin" />
OR
<clarity:restAuth var="restAuth" userId="${gel_processInitiator}" />
```

- clarity:destroyRestAuth

```
<clarity:destroyRestAuth var="result" map="${restAuth}" />
```

Jam Tags - REST Actions

- rest:get
- rest:post
- rest:delete
- rest:patch
- rest:put

```
<rest:post var="restResponse" url="${restResource}" headers="${restAuth}"
content="${restBody}" />
```

Jam Tags - REST Actions

- var (Optional)
  - The result of the request.
  - Return Type: com.rego.net.clients.rest.impl.TagHttpResponse
- url (Required)
  - The URL of the resource where the action will be performed.
  - Type: java.lang.String
- headers (Optional)
  - A container of headers that will be passed with the request. Two types of containers are supported. Any other type will throw an exception.
    - java.util.HashMap<String, String>
    - java.util.ArrayList<org.apache.http.NameValuePair>
  - Type: java.lang.Object

Jam Tags - REST Actions

- params (Optional)
  - A container of URL query parameters that will be passed with the Request. Two types of containers are supported. Any other type will throw an exception.
    - java.util.HashMap<String, String>
    - java.util.ArrayList<org.apache.http.NameValuePair>
  - Type: java.lang.Object
- content (Optional)
  - The request's body. Either contents or json can be used but not both.
  - Type: java.lang.String
- json (Optional)
  - The request's body. Either json or contents can be used but not both.
  - Type: com.google.gson.JsonElement

Jam Tags - REST Response

- Result of all REST actions is a container class (com.rego.net.clients.rest.impl.TagHttpResponse) with the following fields:
  - **code**: the HTTP status code (int)
  - **success**: Whether or not the request succeeded. This is calculated by checking if the code is greater than or equal to 200 and less than 300. (boolean)
  - **json**: The response to the request. When the json field is accessed, the response is attempted to be parsed into a Google GSON object which can fail and throw an exception if the JSON was malformed. (com.google.gson.JsonElement)
  - **body**: The raw response to the request as a String. (java.lang.String)
  - **headers**: A map with the response headers. (java.util.Map<String, String>)

Jam Tags - JSON

- rest:json
  - Creates a com.google.gson.JsonElement from the provided input

```
<rest:json var="restBody" encode="false" escapeText="false">
<![CDATA[{
  "name": "Test Project",
  "code": "PRJ-TEST"
}]]>
</rest:json>
```

  - Allows you to reference values in the JSON later

```
<gel:log>Project Name: ${restBody.name.getAsString()}</gel:log>
```

Jam Tags - JSON

- rest:escapeJson
  - Escapes a string to be safe in JSON

```
<core:set var="inputString"><![CDATA[They didn't say, "Hamburger!"]]></core:set>
<rest:escapeJson var="result" value="${inputString}"/>
```

result: They didn't say, \"Hamburger!\"

- rest:unescapeJson
  - Unescapes a JSON string

```
<core:set var="inputString"><![CDATA[They didn't say, \"Hamburger!\"]]></core:set>
<rest:unescapeJson var="result" value="${inputString}"/>
```

result: They didn't say, "Hamburger!"

Jam Tags - URL Encoding

- rest:decodeUrl
  - Decodes URL encoded string

```
<core:set var="fields" encode="false"><![CDATA[clientID%2CownerID]]></core:set>
<rest:decodeUrl var="decodedFields" value="${fields}"/>
```

decodedFields: clientID,ownerID

- rest:encodeUrl
  - Encodes a string for use in a URL

```
<core:set var="fields" encode="false">ClientID,ownerID</core:set>
<rest:encodeUrl var="encodedFields" value="${fields}"/>
```

encodedFields: clientID%2CownerID

Part III:
Clarity REST Examples

Questions?

Thank You For Attending Rego University

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