

IUG 2024 Intro to REST

April 9-11, 2024 | Dublin, Ireland

Dave Presuhn | Medical Device IoT Platform Manager



Dave Presuhn

Medical Device IoT Platform Manager

I'm a platform manager, not a developer. I'm also not Ryan Reynolds' brother.

Cumulocity IUG 2024

Before we start...

The information provided in this presentation (a) is provided “as is” without warranty of any kind, either express or implied including, without limitation, completeness or accuracy of information, (b) is for informational purposes only and should not be construed as legal or medical advice, and (c) is provided on behalf of the presenter and not on behalf of presenter’s employer

Unless otherwise stated, images by Unknown Author are licensed under CC BY-S

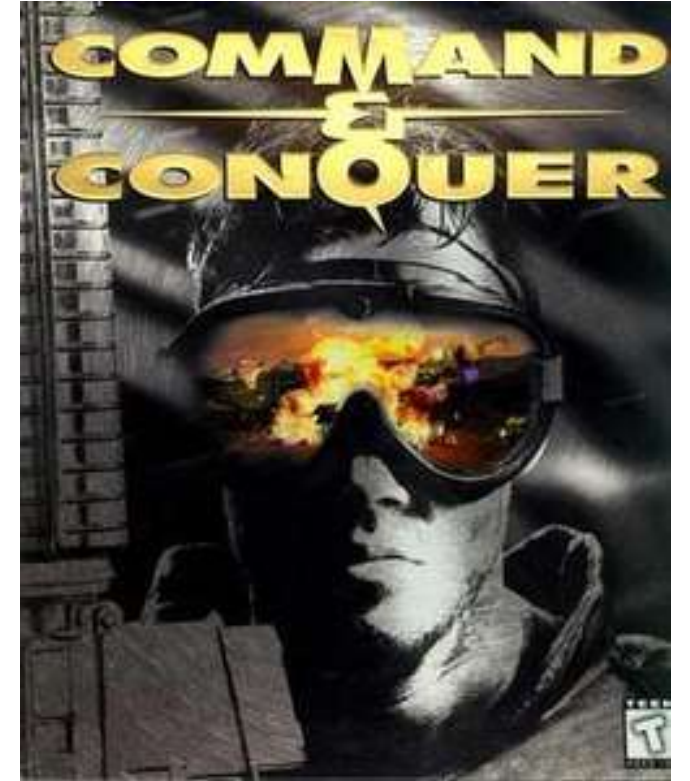
What is REST

- A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.
- REST is a set of architectural constraints
- Response is delivered in one of several formats via HTTP: JSON (JavaScript Object Notation), HTML, XLT, Python, PHP, or plain text
- Stateless: No client information is stored between get requests and each request is separate and unconnected. Each request must contain all the information needed to process the request.

Some basic commands used in Cumulocity

And how to conquer using REST...

- POST: request to create records
 - GET: request to read or get a resource (a document or image, a collection of other resources) from the server
 - PUT: request to update records
 - DELETE request to delete a resource from a server
-
- These are part of a CRUD (Create, Read, Update, and Delete) action suite



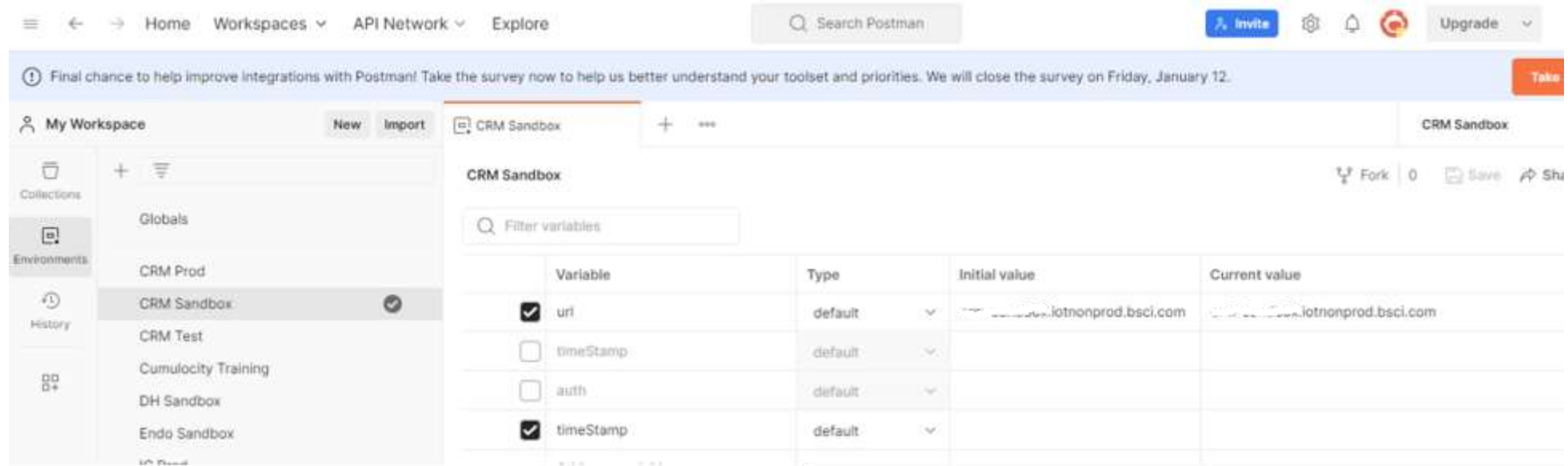
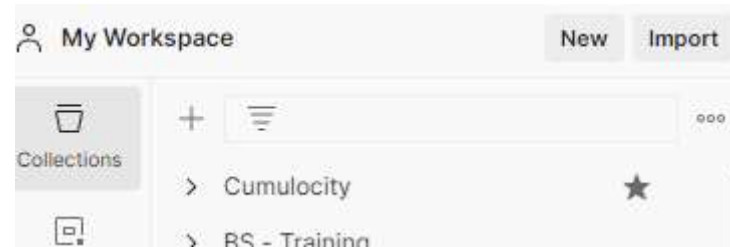
Commonly used tools with REST

- Postman
 - <https://www.postman.com/downloads/>
 - Used to create queries
- Python
 - <https://www.python.org/>
 - Used to create repeatable scripts and to iterate through queries and data
- Curl
 - <https://curl.se/download.html>
 - Available for Amiga, BeOS, DOS, OS/2 and others!
 - Many examples use curl
- API examples
 - https://cumulocity.com/guides/files/rest/Cumulocity_API.postman_collection.json



Setting up Postman

- Download and install Postman
- Import Postman Collection
- Set up your environments



Set up SSO in Postman

Params • Authorization • Headers (9) Body Pre-request Script Tests Settings

Type OAuth 2.0

The authorization data will be automatically generated when you send the request. Learn more about [authorization](#)

Add authorization data to Request Headers

Current Token

This token is only available to you. Sync the token to let collaborators on this request use it.

Token 1-17-24

Header Prefix Bearer

Params • Authorization • Headers (9) Body Pre-request Script Tests Settings

OAuth token
This will allow anyone with access to this request to view and use it.

Configure New Token

Token Name 1-17-24

Grant type Authorization Code

Callback URL 1 Redirect URL

☐ Authorize using browser

Auth URL 1 Azure AD Address/tenant/oauth2/authorize

Access Token URL 1 Redirect URL/token

Client ID 1 Application ID

Client Secret 1 Client Secret

Scope 1 openid

State 1 jwt

Client Authentication 1 Send client credentials in body

Use the JWT

Click Get New Access Token

- What's in the token? Jwt.io

MANAGE ACCESS TOKENS

All TokensDelete

Token Details

Use Token

1-17-24

Token Name

Access Token

access_token_url

client_id

client_secret

timestamp

1-17-24

GFIMTY1Z
IOIJcmOu
vbSIsInN1
DlxYmJOC
E3MDY1NT
J1MzA3LC
JljoIZIVZc1
129YpmK
nK88G9nx
/i0OSHvTj
qLGZskun
EfeMRwK
BnebkFNw
yqvUxVFN

https://...iot.bscl.com/tenant/oauth/token

1706551070713

Encoded



Decoded

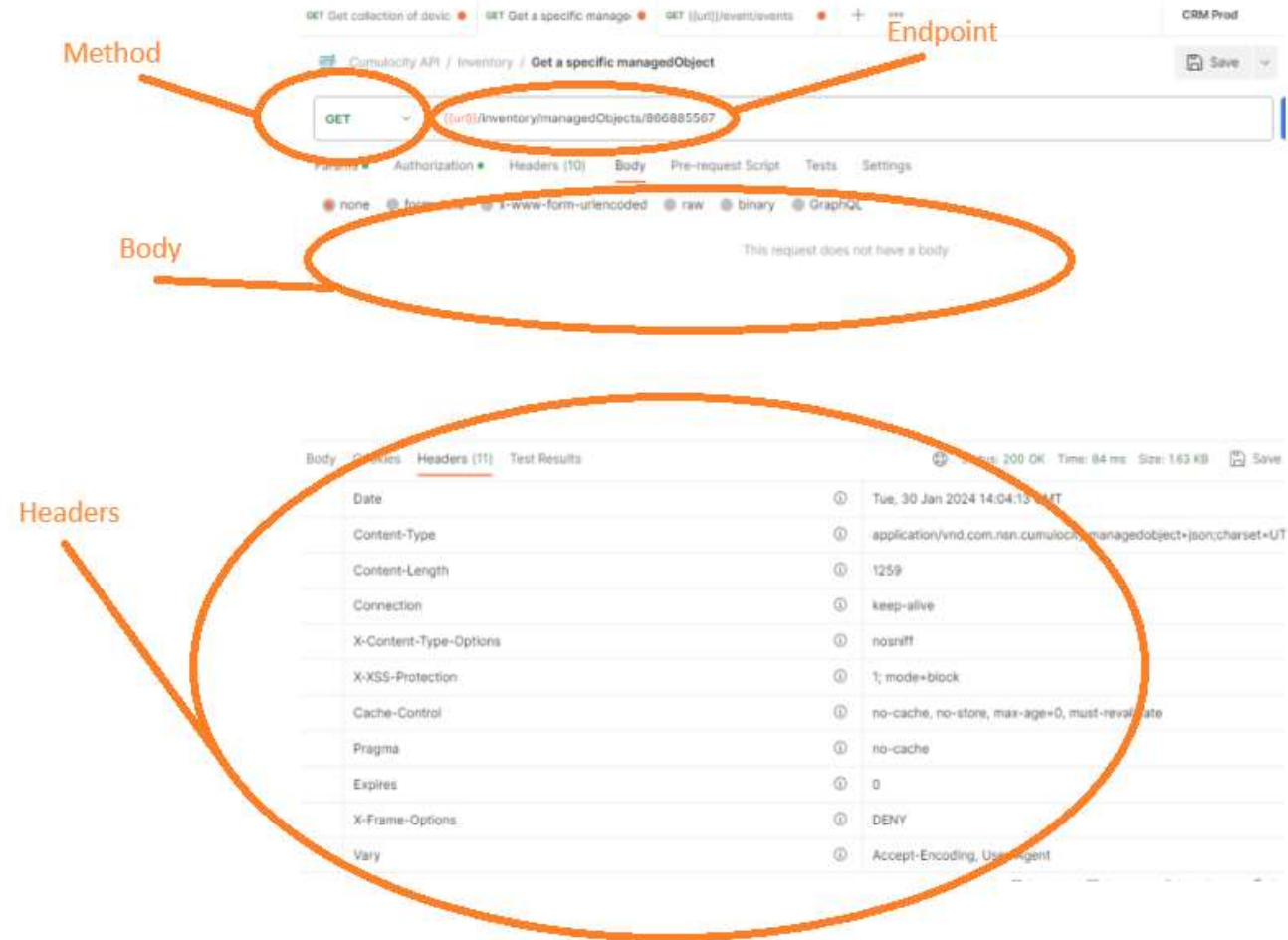


That was fun but let's actually do something

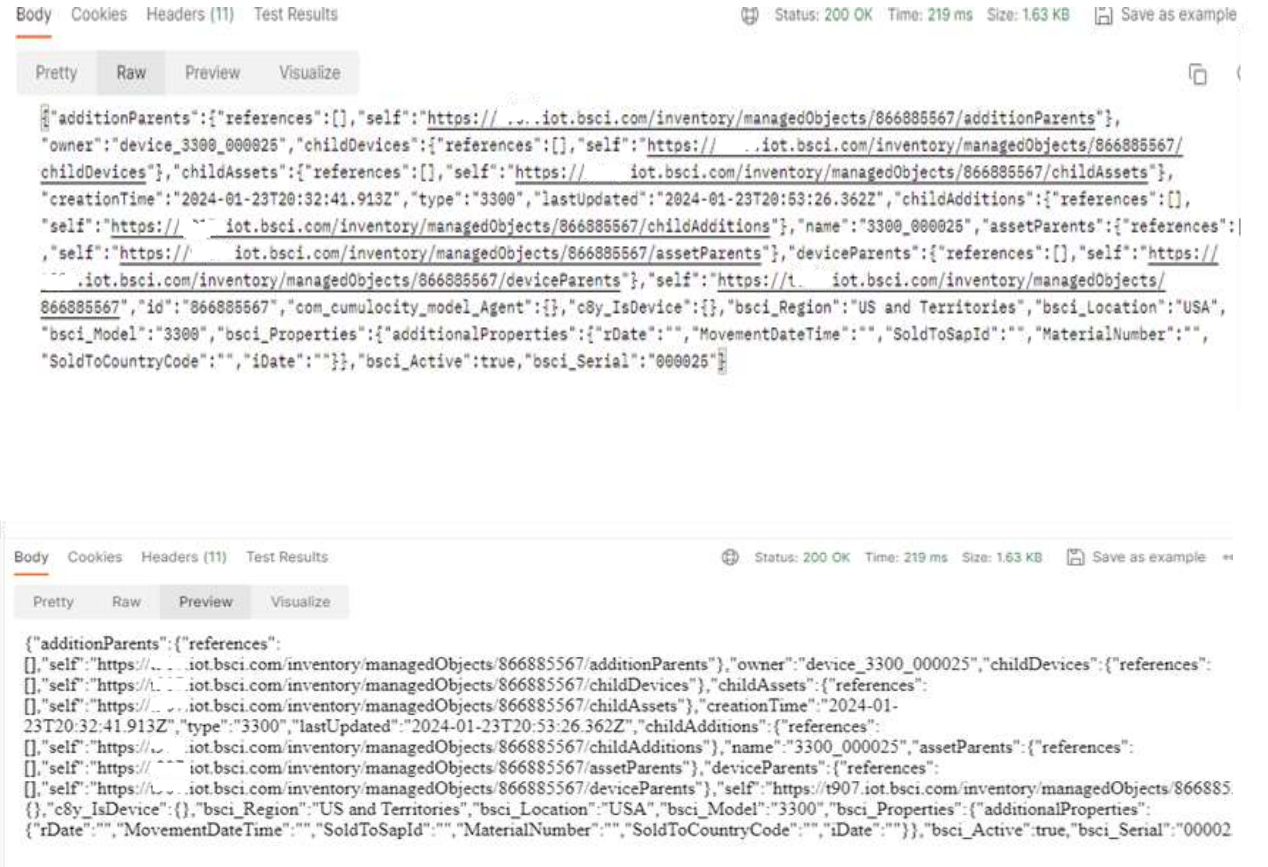
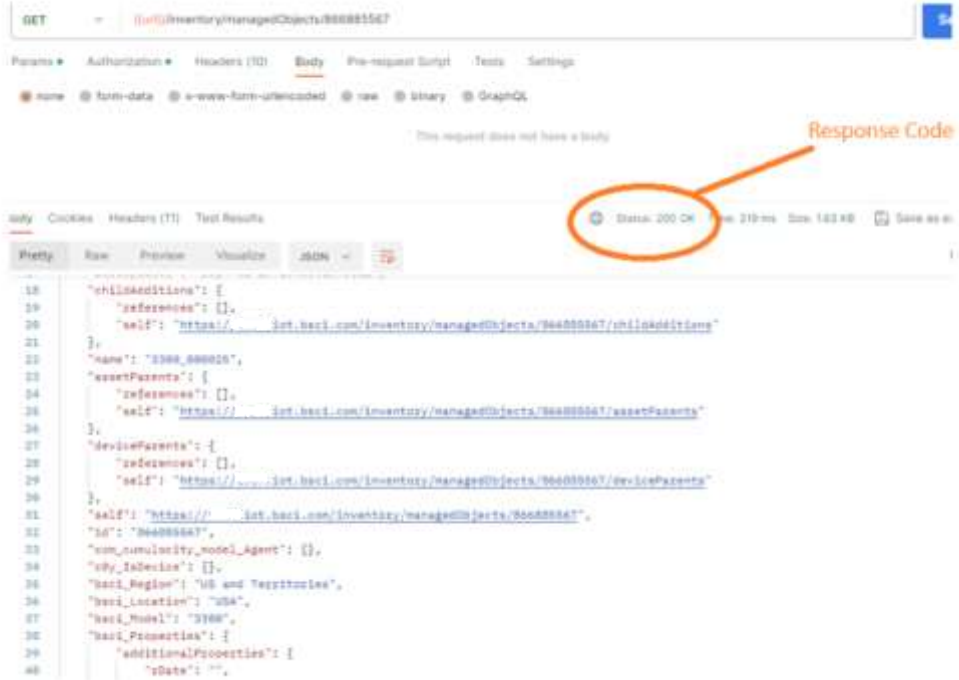
Let's run a query!

What are the elements of a query?

- Method: what to do:
Get/Put/Post/Delete
- Endpoint: who are we trying to reach
- Body: additional data for the server. Get requests quite often do not have a body
- Headers: information relevant to the client and server. Mainly authentication data and response format



And what happened?



Response codes

- There are a lot of codes
- Most important:
 - 200: Request has succeeded
 - 201: Created
 - 400: Bad request. Could not be understood by the server due to syntax error
 - 403: Forbidden. Client does not have access rights to the content
 - 404: Not Found. Server cannot find the requested resource
 - 500: Internal Server Error. Server cannot fill the request due to an unexpected condition



Find all of them at <https://restfulapi.net/http-status-codes/>

My personal favorite: <https://datatracker.ietf.org/doc/html/rfc2324> (response code 418: I'm a teapot)

We want to do more

What else

- Sometimes we can pass a parameter to the query to get better responses
 - Use “?” and parameter
- We can pass multiple parameters to a query
 - Use “&” to pass multiple parameters

GET {{url}}/event/events

gets ALL the events. Right now that's 220671 pages, with 5 events per page

pageSize=100

withTotalPages=true

GET {{url}}/event/events?pageSize=100&withTotalPages=true

Now our total pages are 11034



That's still a lot of crap to weed through

- I only want to see events between Jan 29 and Jan 30, 2024
- Get {{url}}/event/events?dateFrom=2024-01-29&dateTo=2024-01-30&pageSize=100&withTotalPages=true
- Now I'm down to 47 pages of data to slog through
- How will I know that?



Fragment

Key-value pair

Events are recorded in Zulu time

```
1393 {
1394   "source": {
1395     "name": "3380_618887",
1396     "self": "https://iot.bsci.com/inventory/managedObjects/46345",
1397     "id": "46345"
1398   },
1399   "type": "bsci_GetSoftwarePackagesCall",
1400   "lastUpdated": "2024-01-30T18:10:06.076Z",
1401   "self": "https://iot.bsci.com/event/events/6963184",
1402   "time": "2024-01-30T18:10:06.072Z",
1403   "id": "6963184",
1404   "text": "API call: get all available software packages"
1405 },
1406 "Statistics": {
1407   "totalPages": 47,
1408   "pageSize": 100,
1409   "currentPage": 1
1410 },
1411 }
```

JSON-formatted data

What is all this stuff?

Fragment

Key-value pair

Events are recorded in Zulu time

JSON-formatted data

```
1393   "source": {
1394     "name": "3300_010087",
1395     "self": "https://10.10.10.10:8080/iot.bsci.com/inventory/managedObjects/45345",
1396     "id": "45345"
1397   },
1398   "type": "bsci_GetSoftwarePackagesCall",
1399   "lastUpdated": "2024-01-30T18:10:06.076Z",
1400   "self": "https://10.10.10.10:8080/iot.bsci.com/event/events/6963184",
1401   "time": "2024-01-30T18:10:06.072Z",
1402   "id": "6963184",
1403   "text": "API call: get all available software packages"
1404 },
1405 ],
1406 "Statistics": {
1407   "totalPages": 47,
1408   "pageSize": 100,
1409   "currentPage": 1
1410 },
1411 }
```

I'm looking for something specific

- Here's the kind of event I'm looking for:

```

1363     {
1364         "creationTime": "2024-01-30T18:10:43.149Z",
1365         "source": {
1366             "name": "3300_006463",
1367             "self": "https://10.10.10.10.iot.bsci.com/inventory/managedObjects/39274",
1368             "id": "39274"
1369         },
1370         "type": "bsci_GetAvailableSoftwarePackagesEvent",
1371         "lastUpdated": "2024-01-30T18:10:43.149Z",
1372         "self": "https://10.10.10.10.iot.bsci.com/event/events/6955752",
1373         "time": "2024-01-30T13:10:40.338-05:00",
1374         "id": "6955752",
1375         "text": "Device requested all available software packages"
1376     },
1377 }

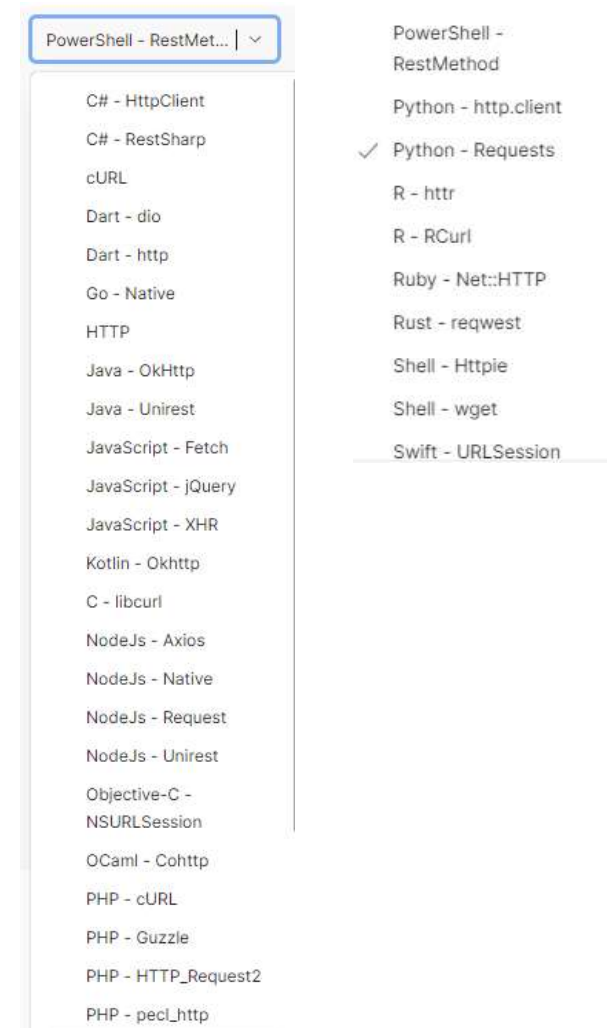
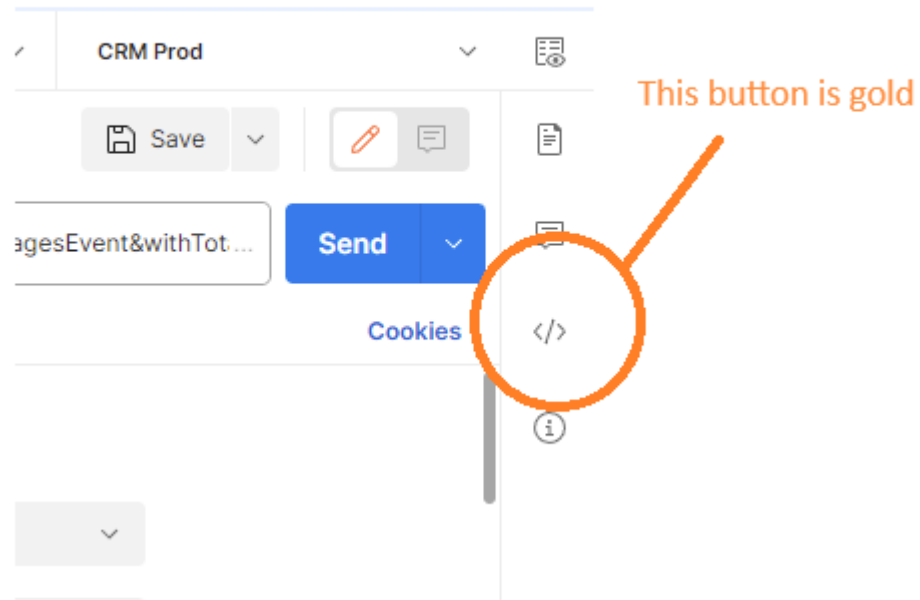
```

```
{{url}}/event/events?dateFrom=2024-01-29&dateTo=2024-01-30&pageSize=1&type=bsci_GetAvailableSoftwarePackagesEvent&withTotalPages=true
```

And now I'm down to 12 pages of events!

I don't want to slog through 12 pages of events

Now what do I do with that data



Python requests

Let's get one event – the first one of the day on Jan 29 2024

```
import requests

url = "[environment].iot.bsci.com/event/events?dateFrom=2024-01-29&dateTo=2024-0130&pageSize=1&type=bsci_GetAvailableSoftwarePackagesEvent&withTotalPages=true"

payload = {}

headers = {
    'Authorization': 'Bearer [JWT]'
}

response = requests.request("GET", url, headers=headers, data=payload)

print(response.text)
```

- One thing to pay attention to: the URL as supplied fails. Add https:// or you'll get an "invalid URL" error.
- That returns a string response.

A string? Ewww...

But we can make a function

```
def fnQueryAndReturnDict (strURL, strAuth, strPayload):
```

```
    """This function takes 3 parameters: the URL to query, the authorization string, and the payload.
```

```
    To query one environment for events between 1/1/70 and 6/15/22 titled
```

```
    GetAvailableSoftwarePackagesEvent,
```

```
    enter the query (for example), use
```

```
    "https://[subtenancy].iot.bsci.com/event/events?dateFrom=1970-01-01&dateTo=2022-06-15&pageSize=20&source=221326&withSourceAssets=true&withSourceDevices=true&type=bsci_
    GetAvailableSoftwarePackagesEvent" """
```

```
    import requests
```

```
    import json
```

```
    headers = {'Authorization': 'Basic ' + strAuth}
```

```
    response = requests.request("GET", strURL, headers=headers, data=strPayload)
```

```
    dictResponse = json.loads(response.text)
```

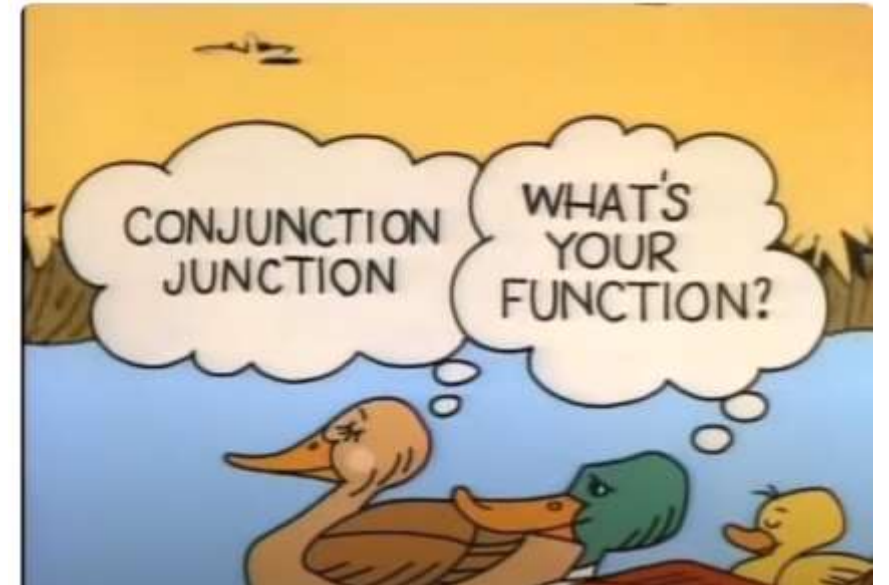
```
    if 'error' in dictResponse:
```

```
        print (dictResponse['error'] + '\n')
```

```
        print (dictResponse['message'])
```

```
    return dictResponse
```

And now we have: A dictionary!



© Schoolhouse Rock

What did that look like?

```
{"next":"https://[tenancy].iot.bsci.com/event/events?dateTo=2024-01-30&pageSize=1&dateFrom=2024-01-29&type=bsci_GetAvailableSoftwarePackagesEvent&currentPage=2&withTotalPages=true","self":"https://t907.iot.bsci.com/event/events?dateTo=2024-01-30&pageSize=1&dateFrom=2024-01-29&type=bsci_GetAvailableSoftwarePackagesEvent&currentPage=1&withTotalPages=true","events":[{"creationTime":"2024-01-29T23:57:58.242Z","source":{"name":"3300_006786","self":"https://[tenancy].iot.bsci.com/inventory/managedObjects/26003","id":"26003"},"type":"bsci_GetAvailableSoftwarePackagesEvent","lastUpdated":"2024-01-29T23:57:58.242Z","self":"https://[tenancy].iot.bsci.com/event/events/6942727","time":"2024-01-29T15:57:36.952-08:00","id":"6942727","text":"Device requested all available software packages"}],"statistics":{"totalPages":1114,"pageSize":1,"currentPage":1}}
>>> type(response.text)
<class 'str'>
```

It's not pretty...

Let's do this manually

```
>>> import json
>>> dictResponse = json.loads(response.text)
>>> type(dictResponse)
<class 'dict'>
>>> dictResponse.keys()
dict_keys(['next', 'self', 'events', 'statistics'])
```

```
>>> dictResponse['events']
[{'creationTime': '2024-01-29T23:57:58.242Z', 'source': {'name': '3300_006786', 'self':
'https://[tenancy].iot.bsci.com/inventory/managedObjects/26003', 'id': '26003'}, 'type':
'bsci_GetAvailableSoftwarePackagesEvent', 'lastUpdated': '2024-01-29T23:57:58.242Z', 'self':
'https://[tenancy].iot.bsci.com/event/events/6942727', 'time': '2024-01-29T15:57:36.952-08:00', 'id':
'6942727', 'text': 'Device requested all available software packages'}]
```

```
>>> type(dictResponse['events'])
<class 'list'>
```

It's a list of dictionaries!

Let's do a custom measurement

We want to store duty cycles

The command: POST

The endpoint: /api/measurement/measurements

JSON-formatted data:

```
{
  "bc_DutyCycleMeasurement": {
    "BSCDC": {
      "value": 188,
      "unit": "Cycles"
    },
    "time": "{{timestamp}}",
    "id": "1996076",
    "type": "BSCDC_Cycles"
  }
}
```

When was this submitted (in Zulu time):

The Device ID:

Status: 201 Created Time: 687ms Size: 780 B

Result code:

The result:

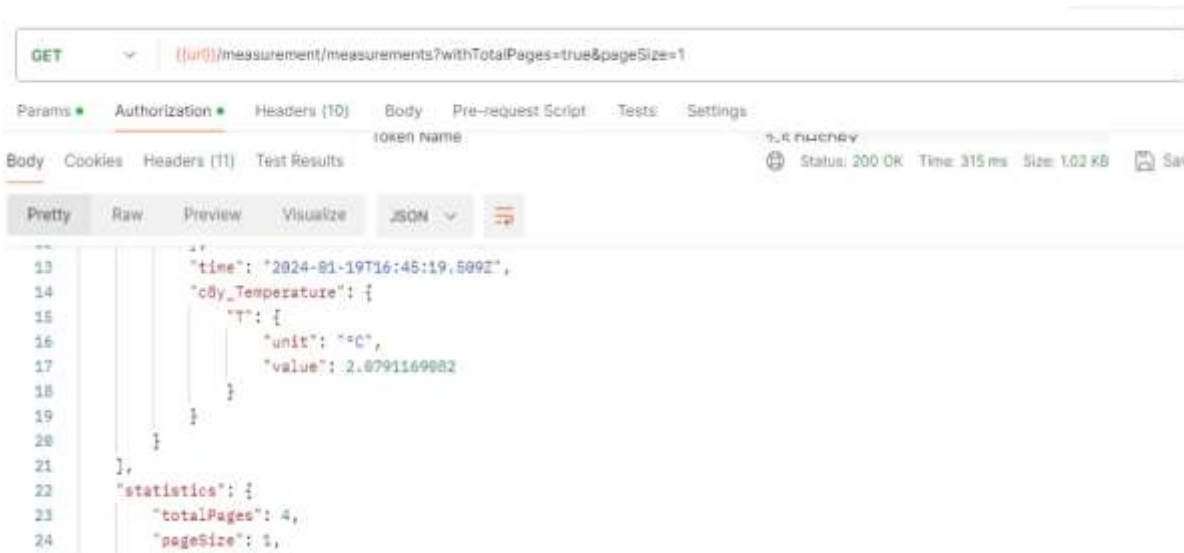
```
{
  "self": "https://...iotnoproduct.baci.com/measurement/measurements/2146348",
  "id": "2146348",
  "type": "bc_DutyCycles",
  "source": {
    "self": "https://...iotnoproduct.baci.com/inventory/managedObjects/1996076",
    "id": "1996076"
  },
  "time": "2024-02-06T16:36:16.979Z",
  "bc_DutyCycleMeasurement": {
    "BSCDC": {
      "unit": "Cycles",
      "value": 188
    }
  }
}
```



That's nice, but..

What do you do with them?

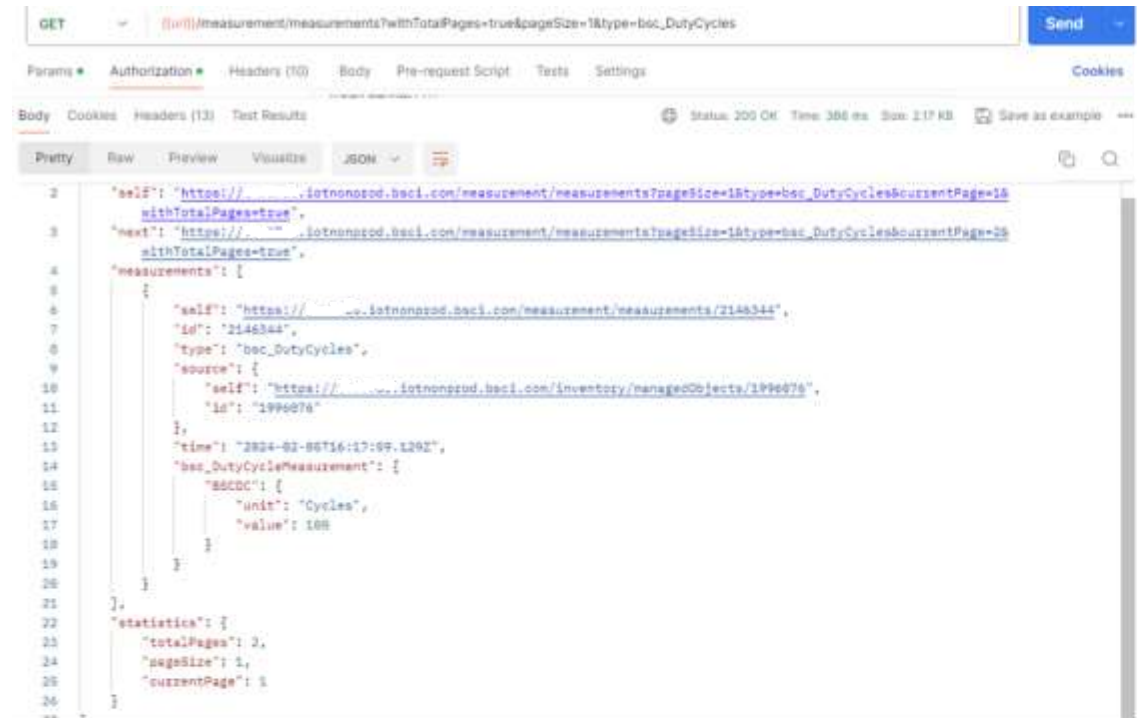
All measurements in that environment



```
GET /measurement/measurements?withTotalPages=true&pageSize=1

{
  "time": "2024-01-19T16:45:19.509Z",
  "c8y_Temperature": {
    "T": {
      "unit": "°C",
      "value": 2.0791169082
    }
  },
  "statistics": {
    "totalPages": 4,
    "pageSize": 1
  }
}
```

Just the bsc_DutyCycles



```
GET /measurement/measurements?withTotalPages=true&pageSize=1&type=bsc_DutyCycles

{
  "self": "https://.../measurement/measurements?withTotalPages=true&pageSize=1&type=bsc_DutyCycles&currentPage=1",
  "next": "https://.../measurement/measurements?withTotalPages=true&pageSize=1&type=bsc_DutyCycles&currentPage=2",
  "measurements": [
    {
      "self": "https://.../measurement/measurements/2146344",
      "id": "2146344",
      "type": "bsc_DutyCycles",
      "source": {
        "self": "https://.../inventory/managedObjects/1996076",
        "id": "1996076"
      },
      "time": "2024-01-05T16:17:09.120Z",
      "bsc_DutyCycleMeasurement": {
        "BSCDC": {
          "unit": "Cycles",
          "value": 108
        }
      }
    }
  ],
  "statistics": {
    "totalPages": 2,
    "pageSize": 1,
    "currentPage": 1
  }
}
```

Last but not least...

How to filter results

- GET {{url}}/inventory/managedObjects?q=\$filter=(bsci_Location+eq+'USA')
- For those times where you want to start with a smaller data set
- Spaces are not processed well
- <https://onlinetexttools.com/url-encode-text> to encode

**Rest can insert, report on,
and update a LOT of things in
Cumulocity but it can be
challenging to learn how to
do it well**

Questions or additions:

David.Presuhn@bsci.com

Disagreements or complaints: null@bitbucket.com

