How to use the CPW REST Interface

Introduction

The Comparative Pathology Workbench (CPW) now has a REST (Representational state transfer) interface. REST interfaces provide a common set of operations to access systems via the set of HTTP commands and JSON text files.

This means that the CPW can now be accessed via an API (Application Programming Interface, for use by third party applications), by providing the relevant JSON file and using the appropriate HTTP command.

Typically, within REST interfaces, the HTTP command GET is used to read data, POST to create data, PUT to update data, and DELETE to delete data, all in combination with the relevant JSON data structure in a text file, directed at the relevant "endpoint" or URL.

Background

The motivation for creating this REST interface was to provide a means of Dumping and Restoring data. For example, a user might have a particularly interesting Bench, that another user might decide they want to recreate. This interface allows existing data to be exported into a JSON format, which can then be reimported to the same or another CPW instance, as required.

A Discussion about Identifiers

The CPW REST Interface provides a means for users to directly manipulate the data held within the CPW, in terms of 3 separate types of component:

1. Benches
2. Cells
3. Images
4. Collections

A Bench consists of many Cells, and a Cell may contain an Image.

The CPW uses automatically incremented integer identifiers, internally generated, to uniquely identify each of these components. Whenever a new Bench, Cell or Image is created in the CPW, the next highest integer is allocated for that entity.

The REST interface regards each component within the CPW as a uniquely identifiable "Resource", addressable via its own unique URL. This unique URL uses the automatically generated identifier for each component, to address the "resource" directly.

For example:

- The URL ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/*** references the Bench that has the identifier of ***1***;

- The URL ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/cells/1/*** references the Cell that has the identifier of ***1***;

- The URL ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/images/1/*** references the Image that has the identifier of ***1***.

These identifiers have NO relationship to ANY other outside resource or website.

The REST Interface provides a complete set of URLs (or "endpoints"), to Create, Read, Update and Delete, ANY and ALL resources held In a CPW instance.

Read, Update and Delete operations via the REST Interface must provide a fully qualified URL, including the relevant identifier, thus: ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/***.

The Create operation does not require an identifier, as this operation will result in a new identifier being allocated, so must be directed to a non-fully qualified URL, thus: ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/***.

The REST Interface provides a means to access the resources in the CPW as if it were a Database. Bench and Image Resources can be manipulated using the usual Create, Read,

Update or Delete ("CRUD") Database operations, be choosing the matching HTTP command, and executing this over the HTTP protocol via the correct URL with a JSON file as required.

Therefore:

- A **POST** command WITH a JSON file, directed at the URL ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/***, will attempt to create a new Bench;

- A **GET** command directed at the URL ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/***, will attempt to read the Bench that has an identifier of ***1***;

- A **PUT** command WITH a JSON file, directed at the URL ***https://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/***, will attempt to fully update the existing Bench that has an identifier of ***1***;

- The **PATCH** command is NOT supported in this Interface.

- A **DELETE** command directed at the URL **https://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/**, will attempt to delete the existing Bench that has an identifier of **1**.

NB. In this implementation, there is no distinction between a Full and a Partial Update – they execute the same processing.

The JSON CPW Format

First, we need to write syntactically correct JSON to represent a Bench within the CPW. The Interface will automatically check for Syntax errors, therefore allowing malformed JSON to be identified immediately, before any processing and possible update takes place.

Each entity (“bench”, “cell” and “image”) must be enclosed in Braces: “{“ and “}”. An Array of components (eg. “cells”) must be enclosed in Square Brackets: “[“ and “]”. Every Attribute consists of a key and a value, each enclosed in “”” and separated by a “:”, and terminated by a “,” except for the final key value pair where no “,” is required.

Here is some example JSON to represent an entire Bench, consisting of Bench attributes and an array of Cells; each Cell has attributes, including examples of a specific Image or Null for no image; and an Image also has attributes; thus:


```
{
    "id": 65,
    "url": "http://127.0.0.1:8000/api/benches/65/",
    "title": "Test Admin Bench",
    "description": "BENCH I",
    "height": 150,
    "width": 150,
    "owner": "admin",
    "bench_cells": [
        {
            "cell_id": 2412,
            "url": "http://127.0.0.1:8000/api/cells/2412/",
            "title": "FAP Ademona 6 - 2016-08-25 13.55.22.ndpi [0]",
            "description": "FAP Ademona 6 - 2016-08-25 13.55.22.ndpi [0]",
            "column_index": 2,
            "row_index": 1,
            "image": {
                "id": 129,
                "url": "http://127.0.0.1:8000/api/images/129/",
                "server": "mwicks23@omero-czi-cpw.mvm.ed.ac.uk",
                "image_id": 301,
                "roi": 0,
                "owner": "admin"
            }
        },

REPEATED N x CELLS

    ]
}
```


NB. Well-formed JSON for a Bench is easily produced by using the GET command on a suitable URL, if a starting point is required!

The next sections describe this JSON format in detail.

JSON to represent a Bench

A Bench is described by the following Fields:

id
This is the system generated Identifier, that uniquely identifies this Bench.
This field is IGNORED when CREATING a new bench, or UPDATING an existing bench.

url
This is the unique URL that can be used to access this Bench “resource”.
This field is IGNORED.

title
Any text, up to 255 Characters.

description
Any text, up to 4095 Characters.

height
An Integer, between 75 and 450.
This is the height in pixels of the “cells” in the bench as they appear in the browser.

width
An Integer, between 75 and 450.
This is the width in pixels of the “cells” in the bench as they appear in the browser.

owner
This is the username of the owner of the data, and MUST exist within the CPW instance already.

bench_cells
An array of Cells in JSON – See next section

JSON to represent a Cell

A Cell is described by the following fields:

cell_id
This is the system generated Identifier, that uniquely identifies this cell.
To create a NEW Cell this id MUST be Zero.
If this identifier is not Zero, then the Interface will attempt to update the Cell based on the contents supplied, for the matching existing cell in the database.
The Interface will delete any existing cells it finds in the database that are not in the supplied JSON for the Bench.

url

This is the unique URL that can be used to access this Cell “resource”.
This field is IGNORED.

title
Any text, up to 255 Characters.

description
Any text, up to 4095 Characters.

column_index
An integer from 0 to X, where X is the largest possible value in a X by Y grid of cells.
NB. If Max X = 4, then there are 5 Columns in the grid.

row_index
An integer from 0 to Y, where Y is the largest possible value in an X by Y grid of cells.
NB. If Max Y = 4, then there are 5 Rows in the grid.

image
An Image in JSON format – see next section
OR
“null” for no image in this cell.

Notes

- When submitting an array of Cells within a Bench JSON structure, there must be at least 9 cells – 1 central data cell (column_index = 1, row_index = 1), surrounded by 8 row/column header/footer cells.

- For an X by Y array of Cells, ALL cells must be specified for ALL possible combinations of X and Y.

- Each combination of X and Y for all cells in a Bench, MUST be unique to ensure geometric consistency of the grid of cells.

- For all cells that have X equal to Zero, no image can be specified – these are Column Header Cells and must be empty.

- For all cells that have X equal to the Maximum X, no image can be specified – these are Column Footer Cells and must be empty.

- For all cells that have Y equal to Zero, no image can be specified – these are Row Header Cells and must be empty.

- For all cells that have Y equal to the Maximum Y, no image can be specified – these are Row Footer Cells and must be empty.

JSON to represent an Image

An Image is described by the following fields:

id
This is the system generated Identifier, that uniquely identifies this Image.
This field is IGNORED when CREATING a new image, or UPDATING an existing image.

url
This is the unique URL that can be used to access this Bench “resource”.
This field is IGNORED.

server
This is the name of the server that hosts the image.
For example, the list current available servers used as sources of images, are:

- "@idr.openmicroscopy.org"
    - Source Name: “IDR”

- "mwicks23@omero1.igmm.ed.ac.uk"
    - Source Name: “mwicks23@omero1”

- "mwicks23@omero-czi-cpw.mvm.ed.ac.uk"
    - Source Name” “mwicks23@CompPathUofE”

- "public_user@omero-czi-cpw.mvm.ed.ac.uk"
    - Source Name: “public_user@CompPathUofE”

- "@workbench-czi-cpw.mvm.ed.ac.uk/wordpress"
    - Source Name: “Your WordPress Images”

image_id
An Integer. Images can be hosted on an OMERO or a Wordpress server.

roi
An Integer. For OMERO images, a Region Of Interest (ROI) within the image may also be specified, as an integer, else Zero.

owner
This is the username of the owner of the data, and MUST exist within the CPW instance already.

JSON to represent a Collection

A Collection is described by the following Fields:

id
This is the system generated Identifier, that uniquely identifies this Collection.
This field is IGNORED when CREATING a new collection, or UPDATING an existing collection.

url
This is the unique URL that can be used to access this Collection “resource”.
This field is IGNORED.

title
Any text, up to 255 Characters.

description
Any text, up to 4095 Characters.

owner
This is the username of the owner of the data, and MUST exist within the CPW instance already.

images
An array of Images in JSON – See next section

Usage

Benches

The following are examples accessing the “benches” endpoint in the REST interface using the command line program “HTTPie” (https://httpie.org/):

- **`http -a uid:pwd POST http://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/ < bench_IN.json`**

- **`http -a uid:pwd --pretty=format --json GET http://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/ > bench_OUT.json`**

- **`http -a uid:pwd PUT http://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/ < bench_IN.json`**

- **`http -a uid:pwd DELETE http://workbench-czi-cpw.mvm.ed.ac.uk/api/benches/1/`**

Cells

JSON representing a Cell can ONLY be submitted WITHIN a Bench JSON structure to the “benches” endpoint. Cells CANNOT be manipulated individually via the REST Interface.

Images

The following are examples accessing the “images” endpoint in the REST interface using the command line program “HTTPie” (https://httpie.org/):

- **`http -a uid:pwd POST http://workbench-czi-cpw.mvm.ed.ac.uk/api/images/ < bench_IN.json`**

- **`http -a uid:pwd --pretty=format --json GET http://workbench-czi-cpw.mvm.ed.ac.uk/api/images/1/ > bench_OUT.json`**

- **`http -a uid:pwd PUT http://workbench-czi-cpw.mvm.ed.ac.uk/api/images/1/ < bench_IN.json`**

- **`http -a uid:pwd DELETE http://workbench-czi-cpw.mvm.ed.ac.uk/api/images/1/`**

Collections

The following are examples accessing the “collections” endpoint in the REST interface using the command line program “HTTPie” (https://httpie.org/):

- `http -a uid:pwd POST http://http://workbench-czi-cpw.mvm.ed.ac.uk/api/collections/ < collection_IN.json`

- `http -a uid:pwd --pretty=format --json GET http://http://workbench-czi-cpw.mvm.ed.ac.uk/api/collections/1 > collection _OUT.json`

- `http -a uid:pwd PUT http://http://workbench-czi-cpw.mvm.ed.ac.uk/api/collections/1 < collection _IN.json`

- `http -a uid:pwd PATCH http://http://workbench-czi-cpw.mvm.ed.ac.uk/api/collections/1 < collection _IN.json`

- `http -a uid:pwd DELETE http://http://workbench-czi-cpw.mvm.ed.ac.uk/api/collections/1`

Notes

1. HTTPie is a command line program suitable for simple requests; "Postman", (with a Graphical User Interface) is more suitable for "heavy duty" access to the REST Interface (https://www.postman.com/)

2. HTTP Response Codes (https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

3. Creating or updating benches via the REST interface produces a lot of network traffic, so possible timeout issues may occur! "HTTPie" provides a means to increase this above the standard 30 seconds by adding the "—timeout NN" option.

Permissions

The CPW REST interface has a comprehensive set of permissions built in.

ALL access to the Interface MUST be Authenticated (ie. A User Id and Password MUST always be supplied).

The **GET** command is available to any user accessing the Interface.

The **POST** command is only available to users that have been correctly set up to communicate with the WordPress Blogging Engine.

The **PUT** and **DELETE** commands are available to the owner of the data being updated or deleted, as well as any admin users, provided they have been correctly set up to communicate with the WordPress Blogging Engine.

The **PUT** and **DELETE** commands are also available to the “editors” of the data being updated or deleted, provided they have been correctly set up to communicate with the WordPress Blogging Engine.

M N Wicks
26th June 2025