C++ REST API

Jason M. Coposky  
@jason_coposky  
Executive Director, iRODS Consortium

June 8-11, 2021  
iRODS User Group Meeting 2021  
Virtual Event
Create a simple to use, easy to deploy, fast, and light-weight REST API

- Based on the C++ API
- Acts as a proxied mid-tier application layer
- Utilizes JSON Web Tokens for AAI

Provides a single executable per endpoint, suitable for containerized deployment

https://github.com/irods/irods_client_rest_cpp
The REST API provides an executable for each individual API endpoint. An nginx template is provided for reference.

Template configuration files are installed by default:

- `/etc/irods/irods_client_rest_cpp.json.template`
- `/etc/irods/irods-client-rest-cpp-reverse-proxy.conf.template`

Copy the `irods_client_rest_cpp.json.template` to `/etc/irods` and edit accordingly.

Copy `irods_client_rest_cpp.json` to `/etc/nginx/sites-available` and link to `/etc/nginx/sites-enabled`. 
```json
{
    "irods_rest_cpp_access_server": {
        "port": 8080,
        "threads": 4,
        "maximum_idle_timeout_in_seconds": 10
    },
    "irods_rest_cpp_admin_server": {
        "port": 8087,
        "threads": 4,
        "maximum_idle_timeout_in_seconds": 10
    },
    "irods_rest_cpp_auth_server": {
        "port": 8081,
        "threads": 4,
        "maximum_idle_timeout_in_seconds": 10
    },
    "irods_rest_cpp_get_configuration_server": {
        "port": 8088,
        "threads": 4,
        "maximum_idle_timeout_in_seconds": 10,
        "api_key": "default_api_key"
    },
    "irods_rest_cpp_put_configuration_server": {
        "port": 8089,
        "threads": 4,
        "maximum_idle_timeout_in_seconds": 10
    },
    "irods_rest_cpp_list_server": {
        "port": 8082,
        "threads": 4,
        "maximum_idle_timeout_in_seconds": 10
    },
    "irods_rest_cpp_query_server": {
        "port": 8083,
        "threads": 4,
        "maximum_idle_timeout_in_seconds": 10
    }
}
```
server {
  listen 80;

  add_header 'Access-Control-Allow-Origin' '*' always;
  add_header 'Access-Control-Allow-Headers' '*' always;
  add_header 'Access-Control-Allow-Methods' 'AUTHORIZATION,ACCEPT,GET,POST,OPTIONS,PUT,DELETE' always;

  location /irods-rest/1.0.0/access {
    if ($request_method = 'OPTIONS') {
      return 204;
    }
    proxy_pass http://localhost:8080;
  }

  location /irods-rest/1.0.0/admin {
    if ($request_method = 'OPTIONS') {
      return 204;
    }
    proxy_pass http://localhost:8087;
  }

  location /irods-rest/1.0.0/auth {
    if ($request_method = 'OPTIONS') {
      return 204;
    }
    proxy_pass http://localhost:8081;
  }

  location /irods-rest/1.0.0/configuration {
    if ($request_method = 'OPTIONS') {
      return 204;
    }

    if ($request_method = GET ) {
      proxy_pass http://localhost:8088;
    }

    if ($request_method = PUT ) {
      proxy_pass http://localhost:8089;
    }
  }
}
Authentication

This REST API relies on the use of JSON Web Tokens to pass:

- Identity
- Authentication information
- Authorization information
- Future role based information

The /auth endpoint must be invoked first, generating a JWT

Send the JWT via the Authorization header for subsequent endpoints

For example:

```bash
curl -X GET -H "Authorization: $\{TOKEN\}" ...
```
This endpoint provides a service for the generation of an iRODS ticket to a given logical path, be that a collection or a data object.

**Method**: POST

**Parameters**:
- `path`: The url encoded logical path to a collection or data object for which access is desired

**Example Curl Command**:
```
curl -X POST -H "Authorization: ${TOKEN}" "http://localhost/irods-rest/1.0.0/access?path=%2FtempZone%2Fhome%2Frods%2Ffile0"
```

**Returns**:
An iRODS ticket token within the X-API-KEY header, and a URL for streaming the object.

```
{
    "headers": [
        "X-API-KEY: CS11B8C4KZX2BI1"
    ],
    "url": "/irods-rest/1.0.0/stream?path=%2FtempZone%2Fhome%2Frods%2Ffile0&offset=0&limit=33064"
}
```
The administration interface to the iRODS Catalog which allows the creation, removal and modification of users, groups, resources, and other entities within the zone.

**Method**: POST

**Parameters**:

- action: dictates the action taken: add, modify, or remove
- target: the subject of the action: user, zone, resource, childtoresc, childfromresc, token, group, rebalance, unusedAVUs, specificQuery
- arg2: generic argument, could be user name, resource name, depending on the value of action and target
- arg3: generic argument, see above
- arg4: generic argument, see above
- arg5: generic argument, see above
- arg6: generic argument, see above
- arg7: generic argument, see above

**Example Curl Command**:

```bash
1 curl -X POST -H "Authorization: ${TOKEN}" "http://localhost/irods-rest/1.0.0/admin?action=add&target=resource[arg2=ufs0][arg3=]
```

**Returns**:

"Success" or an iRODS exception
This endpoint provides an authentication service for the iRODS zone.

Currently only native iRODS authentication is supported, as Basic or Native.

**Method**: POST

**Parameters**: None

**<token>**: base64 encoded username:password payload

**Example Curl Command**:

```bash
export TOKEN=$(curl -X POST -H "Authorization: Basic <token>" "http://localhost:80/irods-rest/1.0.0/auth")
```

**Returns**:
An encrypted JWT which contains everything necessary to interact with the other endpoints. This token is expected in the Authorization header for the other services.
This endpoint will return a JSON structure holding the configuration for an iRODS server. This endpoint takes a known API key for authorization which is configured in /etc/irods/irods_client_rest_cpp.json

**Method**: GET

**Parameters**: None

**Example Curl Command**:
```
curl -X GET -H "X-API-KEY: ${API_KEY}" "http://localhost/irods-rest/1.0.0/configuration" | jq
```

**Returns**: A json array of objects whose key is the file name and whose contents is the configuration file.

**Note**: As of 4.3+ the iRODS server will be able to leverage centralized configuration using this service.
Method: GET

Example Return Value:

```json
{
  "host_access_control_config.json": {
    <SNIP>
  },
  "hosts_config.json": {
    <SNIP>
  },
  "irods_client_rest_cpp.json": {
    <SNIP>
  },
  "server_config.json": {
    <SNIP>
  },
  "server_config.json": {
    <SNIP>
  }
}
```
This endpoint will write the url encoded JSON to the specified files in /etc/irods

**Method**: PUT

**Parameters:**

- **cfg** - a url encoded json string of the format

```json
{
  "file_name": "test_rest_cfg_put_1.json",
  "contents": {
    "key0": "value0",
    "key1": "value1"
  }
},
{
  "file_name": "test_rest_cfg_put_2.json",
  "contents": {
    "key2": "value2",
    "key3": "value3"
  }
}
```

**Example Curl Command:**

```
export CONTENTS="%5B%7B%22file_name%22%3A%22test_rest_cfg_put_1.json%22%2C%20%22contents%22%3A%22%7B%22%22%7B%22key0%22%3A%22value0%22%2C%22key1%22%3A%22value1%22%7D%7D%2C%22test_rest_cfg_put_2.json%22%2C%20%22contents%22%3A%22%7B%22%22%7B%22key2%22%3A%22value2%22%2C%22key3%22%3A%22value3%22%7D%7D%7D"

curl -X PUT -H "Authorization: ${TOKEN}" "http://localhost/irods-rest/1.0.0/configuration?cfg=${CONTENTS}"
```

**Returns:**

None
This endpoint provides a recursive listing of a collection, or stat, metadata, and access control information for a given data object.

**Method**: GET

**Parameters**:
- **path**: The url encoded logical path which is to be listed
- **stat**: Boolean flag to indicate stat information is desired
- **permissions**: Boolean flag to indicate access control information is desired
- **metadata**: Boolean flag to indicate metadata is desired
- **offset**: number of records to skip for pagination
- **limit**: number of records desired per page

**Example Curl Command**:
```
1 curl -X GET -H "Authorization: ${TOKEN}" "http://localhost/irods-rest/1.0.0/list?path=%2FtempZone%2Fhome%2Frods&stat=0&permis
```
Returns:
A JSON structured response within the body containing the listing, or an iRODS exception

```json
{  "_embedded": [  {    "logical_path": "/tempZone/home/rods/subcoll",
      "type": "collection"
    },
    {    "logical_path": "/tempZone/home/rods/subcoll/file0",
      "type": "data_object"
    },
    {    "logical_path": "/tempZone/home/rods/subcoll/file1",
      "type": "data_object"
    },
    {    "logical_path": "/tempZone/home/rods/subcoll/file2",
      "type": "data_object"
    },
    {    "logical_path": "/tempZone/home/rods/file0",
      "type": "data_object"
    }  ],
  "_links": {  "first": "/irods-rest/1.0.0/list?path=%2FtempZone%2Fhome%2Frods&stat=0&permissions=0&metadata=0&offset=0&limit=100",
               "last": "/irods-rest/1.0.0/list?path=%2FtempZone%2Fhome%2Frods&stat=0&permissions=0&metadata=0&offset=UNSUPPORTED&limit=100",
               "next": "/irods-rest/1.0.0/list?path=%2FtempZone%2Fhome%2Frods&stat=0&permissions=0&metadata=0&offset=100&limit=100",
               "prev": "/irods-rest/1.0.0/list?path=%2FtempZone%2Fhome%2Frods&stat=0&permissions=0&metadata=0&offset=0&limit=100",
               "self": "/irods-rest/1.0.0/list?path=%2FtempZone%2Fhome%2Frods&stat=0&permissions=0&metadata=0&offset=0&limit=100"
  }}
```
This endpoint provides access to the iRODS General Query language, which is a generic query service for the iRODS catalog.

**Method**: GET

**Parameters**:  
- `query_string`: A url encoded general query  
- `query_limit`: Number of desired rows  
- `row_offset`: Number of rows to skip for paging  
- `query_type`: Either 'general' or 'specific'

**Example Curl Command:**
```
1 curl -X GET -H "Authorization: $TOKEN" "http://localhost/irods-rest/1.0.0/query?query_limit=100&row_offset=0&query_type=ge"
```
Returns:

A JSON structure containing the query results

```
{
    "_embedded": [
        [{
            "path": "/tempZone/home/rods",
            "data": "file0"
        },
        [{
            "path": "/tempZone/home/rods/subcoll",
            "data": "file0"
        },
        [{
            "path": "/tempZone/home/rods/subcoll",
            "data": "file1"
        },
        [{
            "path": "/tempZone/home/rods/subcoll",
            "data": "file2"
        }]
    ],
    "_links": {
        "first": "/irods-rest/1.0.0query?query_string=SELECT%20COLL_NAME%2C%20DATA_NAME%20WHERE%20COLL_NAME%20LIKE%20%27%2FtempZone%2Fhome%2Frods%2C%20DATA_NAME%20LIKE%20%27%2F",
        "last": "/irods-rest/1.0.0query?query_string=SELECT%20COLL_NAME%2C%20DATA_NAME%20WHERE%20COLL_NAME%20LIKE%20%27%2FtempZone%2Fhome%2Frods%2C%20DATA_NAME%20LIKE%20%27%2F",
        "next": "/irods-rest/1.0.0query?query_string=SELECT%20COLL_NAME%2C%20DATA_NAME%20WHERE%20COLL_NAME%20LIKE%20%27%2FtempZone%2Fhome%2Frods%2C%20DATA_NAME%20LIKE%20%27%2F",
        "prev": "/irods-rest/1.0.0query?query_string=SELECT%20COLL_NAME%2C%20DATA_NAME%20WHERE%20COLL_NAME%20LIKE%20%27%2FtempZone%2Fhome%2Frods%2C%20DATA_NAME%20LIKE%20%27%2F",
        "self": "/irods-rest/1.0.0query?query_string=SELECT%20COLL_NAME%2C%20DATA_NAME%20WHERE%20COLL_NAME%20LIKE%20%27%2FtempZone%2Fhome%2Frods%2C%20DATA_NAME%20LIKE%20%27%2F"
    },
    "count": 4,
    "total": 4
}
```
Stream data into and out of an iRODS data object

**Methods**: PUT and GET

**Parameters**:
- path : The url encoded logical path to a data object
- offset : The offset in bytes into the data object
- limit : The maximum number of bytes to read

**Example Curl Command**:

```bash
$ curl -X PUT -H "Authorization: ${TOKEN}" -d "This is some data" "http://localhost/irods-rest/1.0.0/stream?path=%2FtempZone%2Fhome%2Frods%2FfileX"  
```

**Returns**:
- PUT : Nothing, or iRODS Exception
- GET : The data requested in the body of the response
Requests a JSON formatted iRODS Zone report, containing all configuration information for every server in the grid.

Method: POST

Parameters:
- none

Example Curl Command:
```
1 curl -X POST -H "Authorization: ${TOKEN}" "http://localhost/irods-rest/1.0.0/zone_report" | jq
```

Returns:
JSON formatted Zone Report

```
{
  "schema_version": "file:///var/lib/irods/configuration_schemas/v3/zone_bundle.json",
  "zones": [  
    {      
      <!-- snip -->
    }
  ]
}
```
Questions?