We will be adding code snippets in other languages.

2020-11-11: Updated to use API key authentication
2020-07-07: Added XQuery examples
2019: 10-07: Added an R example
2019-07-09: Added a Java example
2019-05-25: Added a SAS example

Introduction

In this article, you will find several ways to connect to the CDISC Library programmatically using REST API.

Fictitious API keys are used in these example code snippets. Substitute them the keys assigned to you, which are accessible via the API Management Developer Portal at https://api.developer.library.cdisc.org. To request a CDISC Library account to access the API Management Developer Portal, visit https://www.cdisc.org/cdisc-library/api-account-request.

Python

The objective is to obtain a list of CDISC Controlled Terminology that are loaded into the metadata repository. Two libraries you will need to make a REST API request and store the JSON output:

```
ct_products_json_1.py

import requests
import json
```

A REST API GET operation is basically a URL with three parts: base URL, endpoint, path.

```
ct_products_2.py

baseURL = "http://library.cdisc.org/api"
queryEndpoint = "/mdr/products"
queryPath = "/Terminology"
```

You may also specify the output format (or, media type). For JSON, you will specify to accept application/json in the request header:

```
ct_products_3_json.py

# fictitious API key used, real one can be obtained through API Management Developer Portal
req = requests.get(baseURL + queryEndpoint + queryPath, headers={'api-key': 'abcdef0123456789abcdef0123456789', 'Accept': 'application/json'})
```

XML is an alternate output format you may specify:

```
ct_products_3_xml.py

# fictitious API key used, real one can be obtained through API Management Developer Portal
req = requests.get(baseURL + queryEndpoint + queryPath, headers={'api-key': 'abcdef0123456789abcdef0123456789', 'Accept': 'application/xml'})
```

SAS

Below is a sample program to also obtain a list of CDISC Controlled Terminology. This program is compatible with v9.4 and SAS OnDemand for Academics. For further reading about PROC HTTP, visit on-line documentation at https://documentation.sas.com/?docsetId=proc&docsetVersion=9.4&docsetTarget=n1tv58hcgw9q4ln1bqjicqddw3.htm&locale=en. Example 9 discusses how to specify HEADER statements for a GET operation. For creating access to SAS OnDemand for Academics, visit https://www.sas.com/en_us/software/on-demand-for-academics.htm.
Java

Below is a basic CDISC Library client in Java. This example client, by Jozef Aerts, demonstrates how to get started creating a CDISC Library REST client in Java.
BasicCdiscLibraryClient.java

/*
 * Copyright 2020 Jozef Aerts.
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.xml4pharma.cdisclibrary.test;

import javax.ws.rs.client.*; // Client and ClientBuilder - javax.ws.rs-api-2.1.jar
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
// Jersey-2 imports
import org.glassfish.jersey.client.*;

public class BasicCDISCLibraryClient {
    private String base = "https://library.cdisc.org/api"; // the base of the service
    private String apiKey = "xxxxxxxxxxxxxxxxxxxxxxxx"; // your API key

    private Client client;

    public BasicCDISCLibraryClient() {
        ClientConfig clientConfig = new ClientConfig();
        // see e.g. https://jersey.github.io/documentation/latest/logging_chapter.html
        // for how to add logging
        // clientConfig.property(LoggingFeature.LOGGING_FEATURE_VERBOSITY_CLIENT,
        // LoggingFeature.Verbosity.PAYLOAD_ANY);
        client = ClientBuilder.newClient(clientConfig);
    }

    // Get the properties of the provided SDTM-IG variable in the provided domain
    // and for the provided IG version
    public String getSDTMVariableProperties(String igVersion, String domain, String variable) {
        WebTarget webTarget = client.target(base).path("mdr").path("sdtmig").path(igVersion).path("datasets")
            .path(domain).path("variables").path(variable);
        Invocation.Builder invocationBuilder = webTarget.request(MediaType.APPLICATION_XML);
        // Response requires javax.annotation-api-1.2.jar, hk2-api-2.5.0-b42.jar,
        // hk2-locator-2.5.0-b42.jar, hk2-utils-2.5.0-b42.jar
        // and jersey-media-json-binding.jar (when requesting JSON response)
        // from the Jersey-2 library - see https://jersey.github.io/
        Response response = invocationBuilder.header("api-key", apiKey).get();
        System.out.println("HTTP Response Status = " + response.getStatus());
        String xml = response.readEntity(String.class);
        // System.out.println(xml);
        return xml;
    }

    public static void main(String[] args) {
        BasicCDISCLibraryClient cl = new BasicCDISCLibraryClient();
        String response = cl.getSDTMVariableProperties("3-3", "PE", "PESTRESC");
        System.out.println("Response = " + response);
    }
}
Below is a basic CDISC Library client in R, tested using R Studio v1.2.5 with R v3.6. This example demonstrates how to obtain a product catalog using the /mdr/products endpoint.

```r
ct_products_listing.R

library(httr)
library(jsonlite)

resp <-
  GET(
    "https://library.cdisclibrary.org/api/mdr/products/Terminology",
    # fictitious API key used, real one can be obtained through API Management Developer Portal
    add_headers("api-key" = "abcdef0123456789abcdef0123456789")
  )

respBody <- content(resp, "text")
respBody_json <- fromJSON(respBody)
View(respBody_json)
```

XQuery

Below demonstrates how to request a quarter of SEND CT from CDISC Library using XQuery. For technical overview about XQuery, visit https://www.w3.org/TR/xquery/all/.

```xml
CDISC_Library_XQuery.xq

(: Copyright 2020 Jozef Aerts.
Licensed under the Apache License, Version 2.0 (the "License");
You may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.
:)

(: Example XQuery calling the CDISC Library using the API :) (: apikey is passed from external :) declare variable $apikey external;


{: The XQuery standard does not natively support basic authentication.
The EXPath extension http-client ( see http://expath.org/modules/http-client/) however does support it.
Some free/open source libraries like BaseX do support this 'http-client' extension.
So you might want to use the BaseX library to run such queries e.g. from Java programs. :)

let $resp := http:send-request(
  <http:request method='get' href='{$query}'>
    <http:header name="api-key" value="{$apikey}"/>
  </http:request>)
return <test>{$resp/json}</test>

Also available is a Java program that incorporates this XQuery method:

BaseXdiscLibraryXQuery.java
```
package org.basex.examples.api;

import java.io.*;
import org.basex.core.BaseXException;
import org.basex.core.Context;
import org.basex.core.cmd.XQuery;

public class BaseXCDISCLibraryXQuery {
    // For running the code, you will need the BaseX Java library, e.g.
    // "BaseX911.jar"
    private Context context;
    private String apiKey = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";

    // CDISC Library XQuery
    private static String cdiscLibraryXQueryString = "(: apikey is passed from external :)\r\n" + "declare variable $apikey external; \r\n" + "let $response := http:send-request(" + "<http:request method='get' href='https://api.library.cdisc.org/api/mdr/products'>" + "<http:header name="api-key" value="$apikey'/> \r\n" + "</http:request>)\r\n" + "return $response";

    public BaseXCDISCLibraryXQuery() {
        context = new Context();
    }

    public void runXQuery() {
        XQuery xq = new XQuery(cdiscLibraryXQueryString);
        xq.bind("apikey", apiKey);
        try {
            String response = xq.execute(context);
            // as XQuery is for XML, the response will be an XML representation of the
            // returned JSON
            System.out.println("response = \n" + response);
        } catch (BaseXException e) {
            e.printStackTrace();
        }
    }

    public String readCdiscLibraryXQueryFormFile(String fileLocation) {
        String query = "";
        BufferedReader reader;
        try {
            reader = new BufferedReader(new FileReader(fileLocation));
        } catch (FileNotFoundException e1) {
            e1.printStackTrace();
            return null;
        }
        StringBuilder stringBuilder = new StringBuilder(); 
        String line = null;
        String ls = System.getProperty("line.separator");
        try {
            while ((line = reader.readLine()) != null) {
                stringBuilder.append(line);
                stringBuilder.append(ls);
            }
        } finally {
            reader.close();
        }
        return stringBuilder.toString();
    }
}
} 
reader.close();
} catch (IOException e) {
    e.printStackTrace();
    return null;
}

// return as a single string
query = stringBuilder.toString();
System.out.println("CDISC Library XQuery was read from file = " + fileLocation);
return query;

public static void main(String[] args) {
    BaseXCDISCLibraryXQuery b = new BaseXCDISCLibraryXQuery();
    // uncomment end edit the following line for reading the XQuery from file
    // cdiscLibraryXQueryString =
    // b.readCdiscLibraryXQueryFormFile("C:\baseX\XQueries\CDISC_Library_XQuery.xq");
    b.runXQuery();
}