1View - Rest API

Next Generation Data Integration

Table of Contents

Document Changes

Document updates	Author	Date
Initial issue	Pete Lovell	10th Oct 2014
Updated with new API calls	Pete Lovell	3rd July 2015
Minor update	Pete Lovell	20th Feb 2016
Updated with Import and Export calls	Pete Lovell	25th July 2016

Introduction

This document is a user guide for the Rest API used for exchanging information with the 1View application.

Overview

The 1View Rest API is a uses HTTP calls and JSON responses to interact with the 1View application for low level data access. The API uses HTTP verbs POST, GET and DELETE for to send and receive data from the 1View server. URI naming is used such that building a request is simple as data objects in 1View can be referenced directly by URI.

Unlike some REST interfaces, the 1View API is not entirely stateless – authentication tokens are generated on login and should be used for each subsequent request.

The use of the Rest API requires a valid 1View user account that has not expired. 1View can be configured with an API level account that does not allow access to the UI with a longer password expiry than usual.

Tools

To develop software using the 1View API it is recommended that a suitable REST API client is used to test the URI's used in the development of a client application. One such API client is the Advanced Rest API client plug-in for the Google Chrome Browser.

Browser

Download Google Chrome here:

https://support.google.com/chrome/answer/95346?hl=en-GB

Rest API Client

Once downloaded and installed you can add the REST API client here:

https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo

This tool allows access the 1View API to login and return data, with a simple user interface. Any issues found using the 1View Rest API should be reproduced using this client before being reported to 1View support.

Note:

At the time of writing, (July 2016) the advanced rest API client appears to have a bug for some URLs where it does not honour the **Set-Cookie** header. This will manifest itself in the HTTP error 401 with a message indicating that the session has timed out.

If this occurs, check to see if the cookie requested in the initial login request has been supplied back to the subsequent requests that return 401. If not, it will be necessary to set the **Cookie** header:

```
Cookie:    JSESSIONID=<session id>;
```

JSON

This API uses the JSON language to return data results. The following tool provides an easily means to validate and prettify JSON data:

http://jsonlint.com/

Login

To login to the 1View application you will need a valid 1View user account. This account must have been used at least once to login to the user interface before it becomes active. (Support for the API is controlled by user privilege and may need to be enabled for a given user. Please contact 1View support if this may be the case.)

Login to the API using the following POST request:

```
https://www.1viewsolutions.com/oneview/restapi/login
```

With the following parameters:

Parameter	Value
username	<1View user name>
password	<1View password>

This will return the following if successful:

```
{
"message": "Successfully authenticated: <user>"
"result": 200
"token": "<token>"
}
```

The token value should then be sent in all other requests. The token can be passed as either a parameter to the POST/GET/DELETE requests or as an HTTP header with the same name.

To test the login, we send a GET request to the same URI:

```
https://www.1viewsolutions.com/oneview/restapi/login
```

And the following parameters/headers:

Parameter	Value
username	<1View user name>
password	<1View password>
Token	<token from login request>

This will return the following if successful:

```
{
"message": "Session exists for: <user>"
"result": 200
"token": "<token>"
}
```

To remove a login, use the same URI and parameters but send a DELETE request:

```
https://www.1viewsolutions.com/oneview/restapi/login
```

Error Handling

A failure in any one of the services requested will return and error code and a message briefly explaining the failure case. Most messages are self-explanatory, but in the case that a message is not, please contact 1View support.

Error messages are returned in the following form:

```
{
"message": "Token invalid."
"result": 401
}
```

Where the result code above is an HTTP response code (that is the same as the HTTP response code for the request). All responses will return an HTTP result code and so a generic handler can be built.

Rest API Handlers

The 1View user interface supports access multiple databases at the same time and so the 1View API is no different. The Rest API handlers therefore require that the database name be specified in the URI of the request.

General Form

The general form of the requests to the API is as follows:

```
https://www.1viewsolutions.com/oneview/restapi/<request>/<db>/<domain>
```

The following table summarises the three fields:

Field	Description
request	The name of the request, for example ‘folders’, ‘domains’, ‘upload’, …
db	The name of the target database. This name will be refdata if using the hosted solution or will be made available by a system administrator.
domain	A domain in 1View is typically the name of a company. In the hosted solution this name will likely be the name of a company. The name of the domain to use will be provided when the 1View user account has been set up – or is available in the user interface at the top of the folder hierarchy. A domain ID can also be used in place of the name of the domain.

Domains

The domain handler will return information pertaining to the domains within the 1View application that a user has permission to see. The request is a GET request is of the following form:

```
https://host/oneview/restapi/domains/<db>
```

This returns JSON in the following format:

```
{
"message": ""
"result": 200
"domains": [
0:  {
"id": <domainid>
"name": "* Summary and Statistics"
"type": 0
}
…
]}
```

The domains that the login user has access to are contained within the array called “domains”.

Folder Hierarchy

The folder children handler allows access to the folder hierarchy in 1View, given a folder id as a root. It is a GET command of the following form:

```
https://host/oneview/restapi/folderchildren/<db>/<domain>
```

This will return all of the folders in the domain, at the root level, that the user has access to, in the following form:

```
{
"message": ""
"result": 200
"folders": [
0:  {
"define": true
"children": [
      0:  {
      "queryid2": 34376
      "define": true
      "queryid1": 34377
      "children": [0]
      "parentid": 14191
      "createdby": 206
```

```
            "queryname1": "All Input data"
            "queryname2": "Last Input: (Nov 24 2015 5:23PM)"
            "swatch": "CCFFCC"
            "id": 14193
            "listsequence": 30
            "querylisttype1": "L"
            "queryname3": "History"
            "querylisttype3": "L"
            "querylisttype2": "L"
            "name": "corporateactions"
            "write": true
            "inputid": 5891
            "filetype": "csv"
            "querycount3": 0
            "tabletype": 1
            "querycount1": 774
            "querycount2": 0
            "queryid3": 34378
            },
            ...
            ]
      ]}
```


This returns the folder hierarchy as returned in the main 1View user interface, for the domain chosen in the URL. Child folders returned, using the same object structure, in the children array contained within the folder object.

Folder Data

The folder handler returns the data contents of the folder in question. A query id is also required which can be obtained from the folder children request. From that request the field **queryid1** returns the green query in the user interface, **queryid2** the amber query and **queryid3** the red query.

The form of the request is as follows:

```
https://host/oneview/restapi/folders/<db>/<domain>/<folderid>/<queryid>
```

This will return data in the following response format:

```
{
"message": ""
"pagesize": 100
"result": 200
"count": 774
"queryid": 34377
"folderid": 14193
"data": []
"filterop": "EQUAL"
"offset": 0
"fields": []
}
```

In the above example the field 'fields' is an array of the following object, one for each field returned in the folder query:

```
{
"id": "38564|241340"
"fieldlistid": 1071371
"joinid": 38564
"fieldid": 241340
"name": "Insref"
"ispicklist": false
"maxcharsize": 7
"table": "J38564"
"type": "N"
"searchable": false
"header": "Insref"
}
```

The ID found above is then used to refer to the data in the data object:

```
{
"38564|241367": "Field Value"
"38564|241380": "2015-10-22 00:00:00.0"
```

```
"38564|241340": 2670
"38564|241345": 20635559
}
```

Data results are limited by a page size of 100. To be able to access further data, another request must be made, passing in an offset. The count field returns the number of results for a query. The following form is used to obtain further results:

```
…/restapi/folder/<db>/<domain>/<folderid>/<queryid>?offset=100
```

The following table summarises additional parameters that can be used to return filtered results:

Parameter	Description
search	Toggle a search term. Pass in this term to search for any value in the result list that matches this term. The search will return a record for the text appearing in any field.
sort	Sort the result using the field and table name in question: `&sort=J38564.[Insref] desc`
filterfield	As with sort, but refers to the name of the filter field: `&filterfield=J38564.[Insref`
filterop	The operation for the filter – can be one of: • Equal or like for Strings • Before or after for Dates • Lt, gt, ge, le for numeric fields.
filtervalue	The value to filter on.
searchtoken	When a search term is toggled on, a token is returned. This token should be passed back to the server with an HTTP DELETE command to free the resources used by the search.
quick	Can be true or false. If set to true a quicker form of the search expression is used that only works on certain key fields, configurable in the user interface.

Import

The import handler can be used to send a file to 1View for the purposes of loading. The file has been loaded before and suitably configured, the results will be automatically differenced.

Submitting a file for loading should be handled with a POST request to the URL:

```
https://host/oneview/restapi/import/<db>/<domain>
```

The following parameters are supported:

Parameter	Value
load	The contents of the file, send as an HTML file input object, in a form.

The use of this feature requires that the contents be to be sent using an HTML multi-part form. For example:

```
<form action="login" method="post" enctype="multipart/form-data">
<label for="load">Upload</label>
<input id="fileupload" type="file" name="load" />
<input type="submit" value="Upload" />
</form>
```

A full example of the upload form that submits the form using Javascript can be found in the OpenRefData API documentation at the following URL:

```
https://www.1viewsolutions.com/oneview/ord/api.html
```

Note: The user must have the 1View user function **'Data Input'** to be able to use this command.

Input

The input handler returns a list of the inputs that a user has loaded into the 1View system. It can also be used to return the current loading status of an input. Inputs are stored against folders in 1View and so are also accessible using the folder hierarchy API calls. This API call is provided for the convenience of loading and to provide a status screen similar to the **'My Files'** folder view in the core 1View application.

The request is a GET request is of the following form:

```
https://host/oneview/restapi/input/<db>/<domain>
```

There are no additional necessary parameters for this call. The response is a list of input objects returned in the following form:

```
{
"inputID": 6341
"loadedQID": 36176
"status": "OK"
"targetKeyFieldID": 46524
"fullFileName": "BXESymbols"
"lastUpdated": "Sun Jul 24 11:02:21 GMT 2016"
"rowsLoaded": 5431
"folderID": 14867
"id": 21554
"userID": 206
"domainID": 171
"targetDisplayName": "Exchange"
"sourceKeyFieldID": 256234
"targetKeyField": "MIC"
"targetInputID": 1801
}
```

The following table summarises each field:

Field	Typical Purpose
id	The ID of this record itself.
domainID	The ID of the domain in which the input belongs.
inputID	The ID of the input as loaded into the system.
folderID	The ID of the folder containing the input and which owns the query stored in loadedQID.
loadedQID	The ID of the query that can be used to retrieve the input data as loaded into the system.
rowsLoaded	The number of rows last loaded.
status	A text string representing current status of the input. This can be OK, Loading or Afterload Processing.
fullFileName	The filename as loaded into 1View

userID	The ID of the user that owns the input.
lastUpdated	The date of last update
sourceKeyFieldID	Where an input is mapped into a target, the field id in this input used as a key field.
ranking	The current ranking of the input, as used in the differencing.
targetDisplayName	The name of the currently mapped target file as used in the differencing engine. Where an input is mapped to more than one target, it may appear multiple times in this list.
targetInputID	The input ID of the target input.
targetKeyFieldID	Where an input is mapped into a target, the field id in the target input used as a key field.
targetKeyField	The field name of the above id.
exportQID	The ID of the query that provides the results of the differencing. This contains the original source data with all of the enrichments and modifications added in from 1View.
updatesQID	A query the same as the exportQID, but one that only shows rows that contain either an update or a difference. (I.e. identical rows are excluded)
nomatchQID	The ID of a query that returns all of the records in the source input that did not match against the target, using the key field.
matchedQID	The ID of a query that returns all of the records in the source input that did match against the target, using the key field.
compareQID	The ID of a query that can be used to compare against the original source and show where data differences and enrichments occur.

Export

The export call supports extracting data from 1View in the form of a CSV, XML or XLSX file. Using this call in conjunction with the import and input commands allows for an automated workflow using the Rest API to drive 1View.

The request is a GET request is of the following form:

```
https://host/oneview/restapi/export/<db>/<domain>/<queryid>
```

Unlike other calls, this call will return the file directly. If there is an error the result will be the usual JSON response. To determine this, check for the MIME type **application/json**. The example detailed in the next section shows how to handle this in Javascript.

The following table summarises the parameters for this request:

Parameter	**Description**
title	The filename of the exported file.
diffQueryID	A comparison query id as returned by the input command. This will return the data of the two queries in a side by side format. For example, using the updatesQID and the loadedQID will produce a side by side comparison similar to the standard 1View differencing view.
format	CSV, XML or XLSX

Note: The user must have the 1View user function **'Cluster List Export'** to be able to use this command.

System Calls

The following API calls exist, but are reserved for system usage:

- **upload** – Allows the passing of file information between 1View systems.
- **publicdomains** – Returns a list of domains in the system that can be shown in the public facing user interface.
- **published** – Returns a list of documents that are publicly available.

Example

An HTML 5/Javascript example of the login, domain, import, export and input commands exists in the following location:

```
https://www.1viewsolutions.com/oneview/ord/api.html
```

This is supported by the javascript file:

```
https://www.1viewsolutions.com/oneview/script/api.js
```

Contact

If you have any difficulty using this guide or accessing the API, please contact us on:

Telephone: +44 (0) 203 174 2109

Email: info@1ViewSolutions.com

Web: www.1viewsolutions.com