

Report API Web Services

Last updated Tuesday, 16 September 2014





Contents

| | |
|---|----|
| Overview | 3 |
| 1. Creating Customisable Reports | 4 |
| 2. Configuring Report API keys..... | 6 |
| 3. Creating a Report API end point from Customisable Report | 7 |
| 4. Creating a Report API request – Remote connection | 8 |
| 5. Making a Report API request – Internal CLS connection | 9 |
| Example: Custom Module Page using jQuery | 10 |
| Example: Custom Module Page using AngularJS directive | 11 |
| 6. Appendix: Report API..... | 12 |
| GetAPIs | 12 |
| GetParams..... | 13 |
| Report API Call | 14 |
| Report API Call as count only | 15 |
| Report API response type | 15 |
| 7. Appendix: Troubleshooting..... | 16 |

Overview

Janison CLS has a rich adhoc reporting framework called "Customisable reports". It allows users to create their own reports based on a certain model (entity) and then picking the desired output columns and filter criteria.

This document will describe how you can use a customisable report as a web service end point for extracting data from the Janison CLS in either JSON or XML formats.

There are several pieces required to work in conjunction with each other to expose and consume CLS Report API. Here's an overview of what required:

1. A customisable report needs to be created in CLS,
2. The customisable report needs to be exposed via creating a Report API Endpoint,
3. Reporting API keys will need to be generated,
4. A programmer will need to write a client app to consume the Report API (we have sample code in C# upon request).

So let's get started.

1. Creating Customisable Reports

You can access this feature by navigating in the CLS to: **Manage Learning -> Customisable Reports**.

Below is an example of a customisable report:

Enrolments in London

Delete Report

Entity

Enrolment

Filters

AND

Add Item

Add Group

Delete all

FieldCatalogueOperator=ValueLondon☐ Allow user control

▼ Select Columns

User Details

☐ Select All

☐ User Name
☐ Active
☐ Created
☐ Email
☒ First name
☐ Identifier
☐ Image Url
☐ Last Logged on
☒ Last name
☐ Mobile
☐ Org Unit
☐ Organisation
☐ Password
☐ Previous Log-on
☒ Username
☐ Work Phone

System Details

☐ Select All

☒ Role
☐ Groups
☐ Points
☒ Manager Username
☐ Manager Name

Custom Attributes

☐ Select All

☐ Global Employment Category ID
☐ Employment Status

Group Types

☐ Select All

☐ FunctionalArea
☒ Geographical Region

Enrolment Details

☐ Select All

☒ Catalogue Item
☒ Completed date
☐ Enrolment Created
☐ Due date
☐ First visit
☐ Expiry Date
☒ Enrolment Status
☐ Score Raw
☐ Points

Other

☐ Select All

☐ Catalogue Item Tags
☐ Catalogue Item Tag Types

Run Report

 or [Cancel](#)

[illegible]

2. Configuring Report API keys

Before the CLS Report API can be consumed API keys will need to be generated in CLS. Only after the API keys are generated can the Report API be used.

The keys will be used by the consumer of the Report API as a "Shared Secret" to generate a hash of the request. This ensures that only service consumers with this shared secret are able to access the CLS Reporting API.

A tenant admin will be required to generate these keys and should be shared with your development team.

We recommend that the Primary Key be used by a first-party client with high trust. We recommend that the Secondary Key be shared with third-party systems. Why? If you supply a Primary key to a third-party and you need to revoke a key (by generating a new key) you will affect your own systems.

Note: Once a key is in use do not generate a new key unless key revocation is necessary. Revoking a key will cause systems that consume the Report API to fail until they're updated to the new key.

1. Navigate in the CLS to: **Settings -> Authentication.**
2. Under **Janison APIs**, select **Generate new key** for the **Primary** and **Secondary** keys.

Authentication

> Authentication Model

▼ Janison APIs

Each Janison API client connecting to this tenant should be configured to use the primary key or the secondary key if the primary key fails validation. This is to allow keys to be reset independently without affecting client connectivity.

Copy the key below and paste into the client's configuration.

| | | |
|----------------|----------------------|-----------------------------------|
| Primary Key: | <input type="text"/> | <button>Generate new key</button> |
| Secondary Key: | <input type="text"/> | <button>Generate new key</button> |

> Access

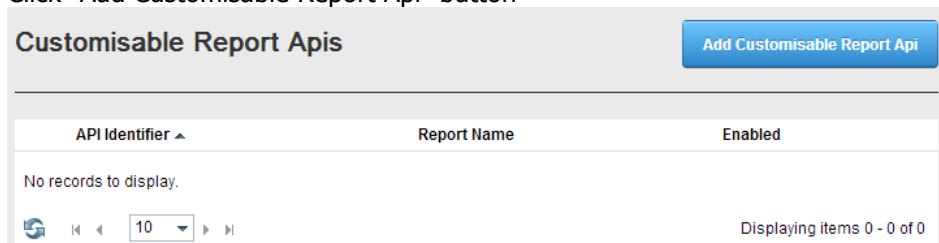
Save settings or [Cancel](#)

> Contextual Help

3. Creating a Report API end point from Customisable Report

A Customisable Report needs to be linked to a "Report API Endpoint" so that it may be consumed. Please follow these instructions to create a Report API endpoint.

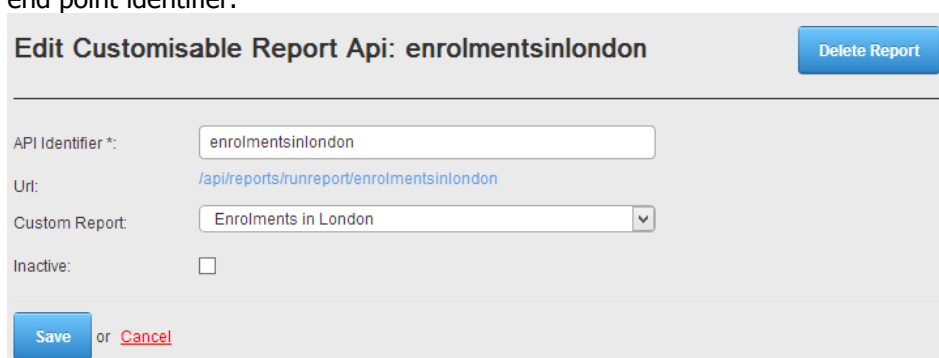
1. Navigate in the CLS to: **Manage Learning -> Reports APIs.**
2. Click "Add Customisable Report Api" button



| API Identifier ▲ | Report Name | Enabled |
|------------------------|-------------|---------|
| No records to display. | | |

Displaying items 0 - 0 of 0

3. Select the report you want to expose and provide an identifier. The identifier will be used as the end point identifier.



Edit Customisable Report Api: enrolmentsinlondon [Delete Report](#)

API Identifier *:

Url: </api/reports/runreport/enrolmentsinlondon>

Custom Report:

Inactive: ☐

[Save](#) or [Cancel](#)

4. The "Url" link displays the url for the end point (excluding the current scheme and domain).

The URL is the endpoint that you will make the request to. It is possible to send through parameters to a Report API to filter results. There's more information below in section Appendix: Report API.

4. Creating a Report API request – Remote connection

To make a valid request to the Report API a special Hash will need to be transported in the payload. This hash is generated in part with the request and the shared secret (as generated in Configuring Report API keys).

Note: You need only use this HMAC shared secret if you're accessing CLS remotely. If you're using the Report API from within CLS (eg via a Custom Module Page) this step is not required.















1. To call a Report API, a correctly formatted http request must be sent to the Report API endpoint Url with a valid HMAC.
2. The following header fields are required: Authorization, Date and Method.
 - a. The 'Authorization' header field = "JanisonAPI <userid>:<GeneratedHash>"
 - i. <userid> = CLS Username of the calling user
 - ii. <GeneratedHash> = HMAC (SHA256) of <hashData> using either Primary or Secondary API keys
 1. To calculate the HMAC:
 - a. Convert the Base-64 encoded Key to a string
 - b. Encode the <hashData> as a UTF8 byte array
 - c. Compute the Hash of the data using the key
 - d. Convert the result to Base-64
 2. <hashData> = "GET\n<url>\n<userId>\n<timestamp>"
 - a. <url> = Report API endpoint url.
 - b. <userId> = CLS Username of the calling user
 - c. <timestamp> = Current timestamp in RFC1123 format.
 - b. The 'Date' header field = <timestamp>
 - i. <timestamp> = Current timestamp (must match timestamp above)
 - c. The 'Method' header field = "GET"
3. The API returns XML by default but by specifying a JSON content-type in the HTTP request it will return JSON.

5. Making a Report API request – Internal CLS connection

If you're creating a request to the Report API from within CLS, for example a Custom Module Page creating a HMAC via a shared secret is not required. The shared secret is not required as the user is already authenticated and the users existing session will be used.

Below are some examples how to make a request to the Report API from Custom Module Pages, via custom javascript. Please also read the section titled Appendix: Report API below.

Security: Please note that basic users will need access to the Report API endpoints. A Janison administrator will need to provide this access to the right.

| | |
|--|---|
|  Web Services | <input type="radio"/> Allow <input checked="" type="radio"/> Deny  |
|  Enrolments | <input checked="" type="radio"/> Inherit <input type="radio"/> Allow <input type="radio"/> Deny  |
|  Reports | <input checked="" type="radio"/> Inherit <input type="radio"/> Allow <input type="radio"/> Deny  |
|  Get Apis | <input checked="" type="radio"/> Inherit <input type="radio"/> Allow <input type="radio"/> Deny  |
|  Get Params | <input checked="" type="radio"/> Inherit <input type="radio"/> Allow <input type="radio"/> Deny  |
|  Run Report Counter | <input checked="" type="radio"/> Inherit <input type="radio"/> Allow <input type="radio"/> Deny  |
|  Run Report | <input checked="" type="radio"/> Inherit <input type="radio"/> Allow <input type="radio"/> Deny  |

Example: Custom Module Page using jQuery

In the example below a Custom Module Page is configured to make a request to the /runreportcounter which will return the name of the Custom Report and the Count of the records in the resultset. Note that the actual row data is not returned, only the count.

Making the same request to /api/reports/**runreport**/dashboard-accepted will return all the rows in the result set.

Accepted - Panel 1

Title*

Accepted content

Content Template*

Plain Text (system) ▼

Use IDE

?

✓

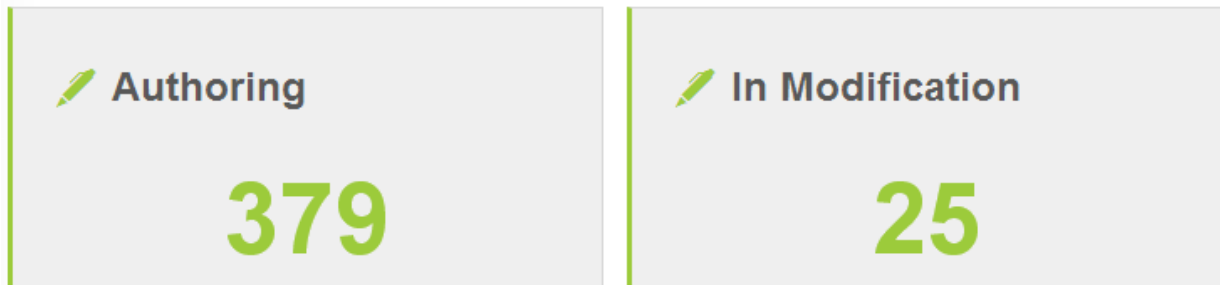
New file

index.html *

```
1 <div class="accepted">
2   <span id="total"></span>
3 </div>
4
5 <script>
6 $(document).ready(function() {
7   $.get('/api/reports/runreportcounter/dashboard-accepted', function(result) {
8     $('#accepted #total').html(result.count);
9   }, 'json');
10 });
11 </script>
```

☐ Show output

Here's an example of a Janison client who's using the Report API to great effect.



Example: Custom Module Page using AngularJS directive

CLS also offers a simplified way to fetch Report API data. This can be performed via Custom Module Pages using AngularJS.

Note: AngularJS directive does not support parameter filtering at this time.

Breakdown AngularJS directive

```
<div custom-report="dashboard-accepted" collect="count"></div>
```

- The custom-report value should be set to the Report API identifier,
- The collect attribute should be configured to instruct the directive what to do with the resultset (and how to make the request). This may be:
 - count (automatically requests /runreportcounter)
 - first
 - last
- The result of the query will be injected into the <div> automatically.

Please take a look at the examples below:

Accepted - Panel 1



Title*

Accepted content

Content Template*

Plain Text (system) ▼

Use IDE

+

New file

index.html *

1

<div custom-report="dashboard-accepted" collect="count"></div>

☐ Show output

To return a list of records:

```
<div custom-report="dashboard-accepted" collect="first 10"></div>
```

Or the last of the records

```
<div custom-report="dashboard-accepted" collect="last 10"></div>
```

Pagination is **not** currently supported eg "first 10 skip 10".

6. Appendix: Report API

We've also provided some helper API endpoints that programmers will appreciate being available. With these it's possible to:

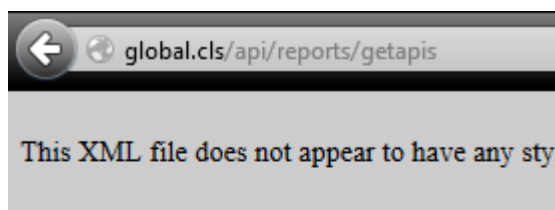
- List of Report APIs available
 - /api/reports/getapis
- List of parameters for a single Report API
 - /api/reports/getparams/[report-api-identifier]
- Run a Report API
 - /api/reports/runreport/[report-api-identifier]?params

GetAPIs

It may be helpful to provide a list of APIs available in a diagnostic or development helper page, or perhaps in an executive reporting page.

/api/reports/getapis

Here's an example of calling the GetAPIs endpoint.



```
- <GetApis>
  <api>GetEnrolments</api>
  <api>GetCatalogueItems</api>
  <api>GetCatalogueItems2</api>
  <api>GetUsers</api>
</GetApis>
```

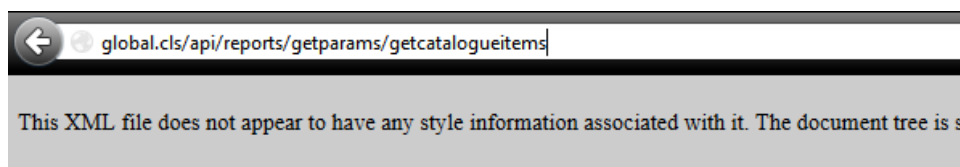
GetParams

A Report API may return too much information. It's possible to filter the result set by using parameters. To identify the parameters that are available for a given Report API use the following endpoint.

An example of how to filter with parameters is displayed in "Sample Report API call" below.

/api/reports/getparams/[report-api-identifier]

Here's an example of the available parameters for the specified API.



```
- <Catalogue_Items>
  - <CatalogueItemDetails>
    <Field>CatalogueItemID</Field>
    <Field>CatalogueItem.Name</Field>
    <Field>CatalogueItem.Identifier</Field>
    <Field>CatalogueItem.Duration</Field>
    <Field>CatalogueItem.Overview</Field>
    <Field>CatalogueItem.LargeThumbnailURL</Field>
    <Field>CatalogueItem.SmallThumbnailURL</Field>
    <Field>CatalogueItem.Created</Field>
    <Field>CatalogueItem.dteLastUpdated</Field>
    <Field>ReEnrolment</Field>
    <Field>ReEnrolmentRaw</Field>
    <Field>AllowReEnrolmentPeriod</Field>
    <Field>CompletionExpiry</Field>
    <Field>CompletionExpiryRaw</Field>
    <Field>AllowContentAccessPeriod</Field>
    <Field>HasCertificate</Field>
    <Field>Status</Field>
    <Field>StatusRaw</Field>
  </CatalogueItemDetails>
</Catalogue_Items>
```

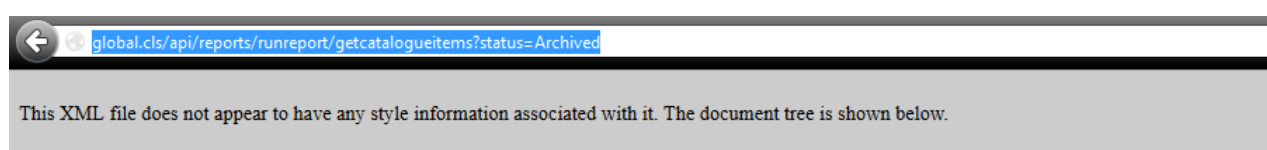
Report API Call

To consume the Report API results you will use the RunReport endpoint. This will return report results as JSON, JSONP or XML (based on the content type below). Here's an example of the endpoint request URL to fetch report data.

```
/api/reports/runreport/[report-api-identifier]
```

It is possible to also use parameters to filter the data via a query string. The available parameters for an API call can be displayed by calling **GetParams** (see previous section). Here's an example of the endpoint request URL with query string for filtering report data.

```
/api/reports/runreport/[report-api-identifier]?param=value&param2=value
```



```
- <Catalogue_Items>
- <CatalogueItemDetails>
  <Catalogue_Item_ID>fd203d78-54e0-4f0a-b01c-17caf9a77250</Catalogue_Item_ID>
  <Name>Janison CAFE Assessment Design</Name>
  <Identifier>janison-cafe-assessment-design</Identifier>
  <Approx_Duration_minutes>
- <Overview>
  The Janison CAFE Assessment Design course provides reference material on all aspects of item design and development.
</Overview>
- <Large_Thumbnail_URL>
  /uploads/janison/Images/CatalogueItemImages/PelicansAtUrunga.jpg
</Large_Thumbnail_URL>
<Small_Thumbnail_URL>/uploads/janison/Images/thumbnails/ov_thumb11.png</Small_Thumbnail_URL>
<Created>
<Modified_On>26/02/2013 03:46 PM</Modified_On>
<Re-enrolment_option>
<Re-enrolment_period>
<Expiry>
<Content_access_period>
<Has_Certificate>False</Has_Certificate>
<Status>Archived</Status>
</CatalogueItemDetails>
- <CatalogueItemDetails>
  <Catalogue_Item_ID>58a71c07-b36d-4cba-bc1e-c7f8fff99725</Catalogue_Item_ID>
  <Name>Janison Content Development</Name>
  <Identifier>janison-content-development</Identifier>
  <Approx_Duration_minutes>
  <Overview>Janison Content Development Overview</Overview>
- <Large_Thumbnail_URL>
  /uploads/janison/Images/CatalogueItemImages/MODMuttonBirdSunset.png
</Large_Thumbnail_URL>
<Small_Thumbnail_URL>/uploads/janison/Images/thumbnails/ov_thumb1.png</Small_Thumbnail_URL>
<Created>
<Modified_On>07/03/2013 03:19 PM</Modified_On>
<Re-enrolment_option>
<Re-enrolment_period>
<Expiry>
<Content_access_period>
<Has_Certificate>False</Has_Certificate>
<Status>Archived</Status>
</CatalogueItemDetails>
</Catalogue_Items>
```

Report API Call as count only

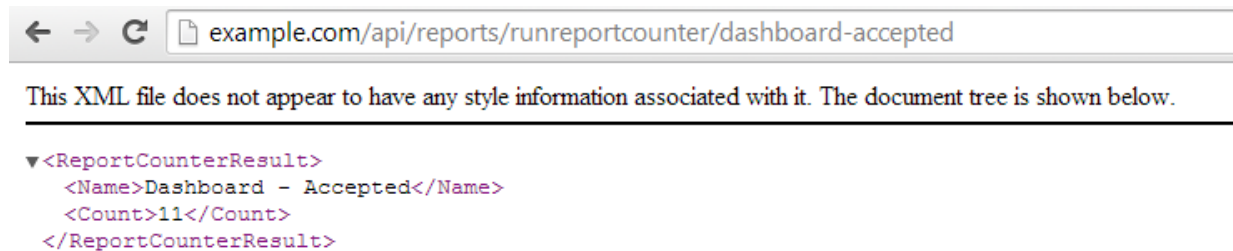
Report API /RunReport is designed to return the result set of the report as JSON, JSONP or XML. If you're calling RunReport just to fetch the number of records there is a separate /RunReportCounter endpoint that can be called.

Using this counter endpoint the API will return the name of the report being executed and the number of records in the resultset.

`/api/reports/runreportcounter/[report-api-identifier]`

Or optionally to filter the report with parameters

`/api/reports/runreportcounter/[report-api-identifier]?param=value¶m2=value`



Report API response type

When using /RunReport or /RunReportCounter you can control the data format of the response. Currently the CLS Report API supports:

- JSON
- JSONP
- XML

The Report API inspects the Accepts request header for:

- JSON: "application/json"
- JSONP: "application/javascript"
- XML: "text/xml" (or any other accepts header)

To set the accepts header on your request you may need to set the Accept header and/or the Content-Type.

Accept : "text/xml",

"Content-Type": "text/xml"

Appendix: Report API

7. Appendix: Troubleshooting

Here are some troubleshooting tips for common issues:

"Request header does not contain expected Authorization value."

- Check to make sure that the API is being called with the same protocol (http/https) that the site is configured to use.

"Invalid authentication data in the request header."

- Check that the Primary/Secondary key is set.
- Check that the key you are using matches the key for the subdomain you are using.