Report Lookup API

Brought to you by IData Incorporated

Table of contents

1 Introduction

The following API specification is a stable draft, but is subject to change as a result of ongoing extensions and enhancements. The report lookup service lets you look up details of specific reports using either the precise report name (ignoring case), the report ID, or the ID of a specific version of a report. If no match is found, the response will return a status of 0, but no results will be shown in the body of the request. The report lookup service can return either the attributes of a report specification or the list of terms related to the report.

2 Request Specification

The DNS name in the URL, to which you submit a request, takes care of telling us the institution whose terms you want to look up. Requests will be submitted over the Internet via the HTTPS protocol.

2.1 URL

This service is provided at:
`https://<your_subdomain>.datacookbook.com/institution/reports/lookup`

2.2 Parameters

Authentication

- **pw** – password of the user who will be authenticating the request. This could be encoded or encrypted. Encoding would be straightforward. Encrypting becomes more complicated because of managing certificates and differences in encoding implementations on different platforms and systems, and between different programming languages.
- **un** – username of user who will be authenticating the request.
- **OR** – include authentication token(s) retrieved using the *service_login* service as cookies in the request.

Required

- **lookup** – string you want to search for. It can be the exact report name (default), the report id, or a report version id. This is used in combination with the **matchType** parameter described below. If you pass an ID, **matchType** needs to reflect the type of ID you are looking up.
- **requestType** – type of request. For report lookup, the request type can be either "***report_lookup***" or "***report_termlist***". These two request type options are described in greater detail below.

Optional

- **jsonFunction** – name of function you want JSON to be passed to on return.
- **jsonVariable** – name of variable you want JSON to be assigned to on return.
- **matchType** – the type of identifier assigned to **lookup** parameter above. Must be one of these values:
    - *exact_text* – Default. Not case sensitive. matchType parameter is not necessary if lookup is done on report name. This will only search approved reports.
    - *report_id* - Searches based on the numerical ID of the report. See explanation box below on how to find a report id. Will only search approved reports.
    - *version_id* - Searches based on the numerical ID of a report version. Unlike the above two match types that are limited to approved reports; this lookup will return that report version regardless of its status in the workflow.
- **outputFormat** – Format you want the results to be rendered in. Potential supported formats: "***rss***", "***html***" (basic, with simple classes on each element), "***xml***" or "***json***". Default is "xml".

Finding Report or Version ID

Every item in the cookbook has a numerical ID. This ID can usually be inferred by looking at the URL of the item. The typical report URL looks like:

```
https://<your_subdomain>.datacookbook.com/institution/reports/1417
```

The number '1417' in the URL above is the Report ID. Similarly, a typical report version URL looks like:

```
https:/…datacookbook.com/institution/reports/1417/versions/1558
```

The number '1558' in this URL is the Report Version ID. The Report Version ID is unrelated to that version's sequence # in the parent report. There is also no connection between Report ID and Report Version ID.

2.3 Request Format

You can choose from one of two request formats: standard HTTP form parameters or request XML.

2.3.1 HTTP Query String Request Format

If you choose to implement requests using http parameters, a search request would be either:

- An HTTPS GET request that has all the parameters above as a part of the query section in the URL.
- An HTTPS POST request that has each of the required parameters above stored as form inputs in the body of the request, as well as any additional optional parameters.

You should only place each parameter in the request once. If a parameter appears in the request more than once and the instances have different values, there is no guarantee which will be used in processing the request.

Any spaces or other special characters within parameters must be escaped in the URL.

Sample HTTP GET query string request:

```
https://idata.datacookbook.com/institution/reports/lookup?un=capncrunc
h&pw=nopeeking!&requestType=report_lookup&lookup=Incoming%20Freshman%2
0Report&matchType=exact_text&outputFormat=xml
```

2.3.2 XML Request Format

The XML request contains the same information, but it is more structured, and is in an XML transaction dialect that is used by all of our XML-based APIs. The XML request in our dialect contains a list of request parameters and <Parameter> elements stored in a <ParameterList>. The required and optional parameters differ for each request type.

Parameters are stored in a ParameterList element, one to a Parameter element. Each parameter contains a <Name> and <Value> element.

An XML request is sent as the body of an HTTP(S) request to the service you want to invoke.

Sample XML service transaction request:

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceTransaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <ServiceRequest serviceName="report_lookup">
        <ParameterList>
            <Parameter>
                <Name>un</Name>
                <Value>capncrunch</Value>
            </Parameter>
            <Parameter>
                <Name>pw</Name>
                <Value>nopeeking!</Value>
            </Parameter>
            <Parameter>
                <Name>requestType</Name>
                <Value>report_lookup</Value>
            </Parameter>
            <Parameter>
                <Name>lookup</Name>
                <Value>Incoming Freshman Report</Value>
            </Parameter>
            <Parameter>
                <Name>matchType</Name>
                <Value>exact_text</Value>
            </Parameter>
            <Parameter>
                <Name>outputFormat</Name>
                <Value>xml</Value>
            </Parameter>
        </ParameterList>
    </ServiceRequest>
</ServiceTransaction>
```

3 Response Specification

The lookup results will be in the body of the HTTPS response to the HTTPS request for report lookup. The response can be returned in a number of formats: RSS, HTML, XML, or JSON.

If you plan on accessing services using AJAX, then JSON or HTML will make more sense, though either RSS or XML would be implementable, as well.

If you are planning on interacting with the API programmatically on a server, then it probably makes more sense to use either RSS or XML, though some server-side languages let you parse JSON, too, and we can also do a passable job treating HTML like XML since we return XHTML.

Each request type will return different data in the response. Response formats for the different report lookup service request types are outlined below.

3.1 Report Lookup Request Type

The Report Lookup request type returns various attributes of the report specification.

Depending on the format, some (RSS, HTML) or all (XML, JSON) of the following fields will be present in the response. See individual response samples for more info.

- **report name** – Name of the report.
- **report-id** - System-generated ID of the report. This ID is part of the URL when accessing the report details within the application.
- **id** – System-generated ID of the report version. This ID is part of the URL when accessing a specific version of the report within the application.
- **description** - Text description of the report.
- **purpose** - Business purpose for creating the report.
- **owner** - Name of the individual, group, department, etc., who has ownership over the report within the institution.
- **additional-details** - Additional requirements or details of the report that do not fit into other fields.
- **access-details** - Information on how the report is accessed. May contain a URL.
- **display-header** - Text mockup of the report page header.
- **display-details -** Mockup or text description of the report body.
- **display-footer** - Text mockup of the report page footer.
- **display-delimiter** - Field delimiter for the report if the report output format is a flat file extract.

3.2 Report Termlist Request Type

The Report Termlist request type returns a list of terms associated with the specified report. This is the same list available under the 'Terms' tab inside a report. The request will return details for each term assigned to the relevant report version. When *matchType* is 'exact_text' or 'report_id', the application will only look at approved reports. If it cannot find an approved report the resulting term list will be empty. If *matchType* is set with 'version_id', the application will return the terms associated with that particular version regardless of its state.

The returned term list will have the latest version of a term, unless that version is rejected. The structure is similar to the global Term List API.

Depending on the format, some (RSS, HTML) or all (XML, JSON) of the following listed *Term* fields will be present in the response.

- **name** – the term name.
- **term-id** – the id of this term
- **id** – the version id that contains the content.
- **functional-definition** – the functional definition of the above version.
- **perma-link-url** – the direct url to the term page for this term.
- **state** – the status of this version (pending, approved etc.).

3.3 RSS Response Format

In an RSS response, an RSS document is the body of the HTTPS response. The lookup implementation will either return one report if there is a match, or no reports if not. No reports are returned if there is an error.

3.3.1 Report Lookup RSS Response Contents

All relevant content is listed under an <item> element. For <item>s returned from the Report Lookup service, the report name is the <title>, the <description> element contains the report description and report purpose; the last time the report was updated is stored in the <pubDate> element; and the URL to the Data Cookbook page for the report is in the <link> and <guid> elements.

For <item>s returned from the Report Termlist service, the term name is the <title>, the term's functional definition is the <description>; the last time the term was updated is stored in the <pubDate> element; and the URL to the Data Cookbook page for the term is in the <link> and <guid> elements.

Our API implements RSS 2.0.

3.3.2 Sample Report Lookup RSS Response

```
<rss version="2.0">
  <channel>
```

```
    <title>IData Data Cookbook Service Response: request type report_lookup</title>
    <description>IData Data Cookbook Service Response: request type report_lookup. Status Code:
0; Status Message: Success!</description>
    <link>http://idatau.datacookbook.local</link>
    <langage>en-us</langage>
    <copyright>Copyright 2012, IData, Inc.</copyright>
    <generator>IData Data Cookbook</generator>
    <managingEditor>bparish@idatainc.com (Brian Parish, President, IData, Inc.)</managingEditor>
    <webMaster>kdezio@idatainc.com (Ken Dezio, CTO, IData Inc.)</webMaster>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <pubDate>Mon, 21 May 2012 20:32:25 -0400</pubDate>
    <lastBuildDate>Mon, 21 May 2012 20:32:25 -0400</lastBuildDate>
    <ttl>30</ttl>
    <item>
      <title>AR Aging Dashboard</title>
      <link>http://idatau.datacookbook.local/institution/reports/1456</link>
      <description>
        <![CDATA[
Description: Top-level bar chart showing the categories of receivables. Top-level pie chart
showing the aging of the receivables.<br/>Purpose: To show the balance based on aging period.
This shows us how old our receivables are. This helps us determine the risk of our older
receivables.
 This can be used to make decisions on how to handle collections.
        ]]>
      </description>
      <pubDate>Fri, 23 Mar 2012 12:26:55 -0400</pubDate>
      <guid isPermaLink="true">http://idatau.datacookbook.local/institution/reports/1456</guid>
    </item>
  </channel>
</rss>
```

3.3.3 Sample Report Termlist RSS Response

```
<rss version="2.0">
  <channel>
    <title>IData Data Cookbook Service Response: request type report_termlist</title>
    <description>IData Data Cookbook Service Response: request type report_termlist. Status Code:
0; Status Message: Success!</description>
    <link>http://idatau.datacookbook.local</link>
    <langage>en-us</langage>
    <copyright>Copyright 2012, IData, Inc.</copyright>
    <generator>IData Data Cookbook</generator>
    <managingEditor>bparish@idatainc.com (Brian Parish, President, IData, Inc.)</managingEditor>
    <webMaster>kdezio@idatainc.com (Ken Dezio, CTO, IData Inc.)</webMaster>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <pubDate>Mon, 21 May 2012 19:57:11 -0400</pubDate>
    <lastBuildDate>Mon, 21 May 2012 19:57:11 -0400</lastBuildDate>
    <ttl>30</ttl>
    <item>
      <title>Aging Period</title>
      <link>http://idatau.datacookbook.local/institution/terms/14757</link>
      <description>
        <![CDATA[
A receivable becomes aged once it is past due. The Aging Period for a receivable indicates how
old it is. These are broken into groups, like 0-30 days, 31-60 days, 61-120 days and > 120 days.
This could be based on the AR bill date, the AR effective date or the AR due date
        ]]>
      </description>
      <pubDate>Fri, 23 Mar 2012 16:19:39 -0400</pubDate>
      <guid isPermaLink="true">http://idatau.datacookbook.local/institution/terms/14757</guid>
    </item>
    <item>
      <title>Receivable Category</title>
      <link>http://idatau.datacookbook.local/institution/terms/14758</link>
      <description>
        <![CDATA[
The category for a receivable is a way of grouping similar transactions. At My University, we
have 5 main categories all transactions roll into: Housing Installment Charges Meal Plans
Registration Registration Fees
```

```
        ]]>
      </description>
      <pubDate>Thu, 15 Dec 2011 15:03:42 -0500</pubDate>
      <guid isPermaLink="true">http://idatau.datacookbook.local/institution/terms/14758</guid>
    </item>
    <item>
      <title>Age Date</title>
      <link>http://idatau.datacookbook.local/institution/terms/14759</link>
      <description>
        <![CDATA[
The age of a receivable. This is used to see how old receivables are to determine what action to
take.
        ]]>
      </description>
      <pubDate>Thu, 15 Dec 2011 15:03:43 -0500</pubDate>
      <guid isPermaLink="true">http://idatau.datacookbook.local/institution/terms/14759</guid>
    </item>
  </channel>
</rss>
```

3.4 HTML Response Format

An HTML response returns the list of matching reports in a simple structure in HTML, with classes and names assigned so you can target CSS to fit their appearance in your application.

3.4.1 Sample HTML Report Lookup Transaction Response

```
<div class="IDATA_services">
  <div class="IDATA_DC_ReportVersionList">
    <div class="IDATA_DC_report" name="Incoming Freshman Report" id="252">
    <div class="IDATA_DC_Id">252
    <div class="IDATA_DC_ReportId">69</div>
    <div class="IDATA_DC_ReportName">Incoming Freshman Report</div>
    <div class="IDATA_DC_ReportDescription">Show the persons name, current mailing address, and
major if known</div>
    <div class="IDATA_DC_ReportPurpose">To display a list of incoming freshman for distribution
to professors.</div>
    <div class="IDATA_DC_AdditionalDetails">This is where you can add additional details like
other requirements.</div>
    <div class="IDATA_DC_Owner">Scott Flory</div>
    <div
class="IDATA_DC_AccessDetails">http://informer.myinst.edu/informer/informer.html#action=reportdet
ails&reportid=12345</div>
    <div class="IDATA_DC_DisplayHeader">                                    Incoming Freshman
Report as of MM/DD/YYYY

                                        For entry in term TTTTTT
</div>
    <div class="IDATA_DC_OutputDescription">The report will produce a text file with the layout
shown below.  This text file will be able to be sent via email to the department heads.</div>
    <div class="IDATA_DC_DisplayLayout">STUDENT NAME                  CITY             ST
ACADEMIC PROGRAM
----------------------        ---------------  ---                 ----------------------
Lastname, Firstname           Anytown          XX                  Mathematics</div>
  </div>
  </div>
</div>
```

3.4.2 Sample HTML Report Termlist Transaction Response

```
<div class="IDATA_services">
  <div class="IDATA_DC_TermList">
    <div class="IDATA_DC_Term" name="Aging Period" id="19056">
    <div class="IDATA_DC_TermName">Aging Period</div>
    <div class="IDATA_DC_FunctionalDefinition">A receivable becomes aged once it is past due.
The Aging Period for a receivable indicates how old it is.  These are broken into groups, like 0-
30 days, 31-60 days, 61-120 days and > 120 days.  This could be based on the AR bill date, the AR
effective date or the AR due date</div>
    <div class="IDATA_DC_URL"><a
href="http://idatau.datacookbook.local/institution/terms/14757">http://idatau.datacookbook.local/
institution/terms/14757</a></div>
  </div>
    <div class="IDATA_DC_Term" name="Receivable Category" id="17103">
    <div class="IDATA_DC_TermName">Receivable Category</div>
    <div class="IDATA_DC_FunctionalDefinition">The category for a receivable is a way of grouping
similar transactions.
At My University, we have 5 main categories all transactions roll into:
Housing
Installment Charges
Meal Plans
Registration
Registration Fees
</div>
    <div class="IDATA_DC_URL"><a
href="http://idatau.datacookbook.local/institution/terms/14758">http://idatau.datacookbook.local/
institution/terms/14758</a></div>
  </div>
    <div class="IDATA_DC_Term" name="Age Date" id="17104">
    <div class="IDATA_DC_TermName">Age Date</div>
    <div class="IDATA_DC_FunctionalDefinition">The age of a receivable.  This is used to see how
old receivables are to determine what action to take.</div>
    <div class="IDATA_DC_URL"><a
href="http://idatau.datacookbook.local/institution/terms/14759">http://idatau.datacookbook.local/
institution/terms/14759</a></div>
  </div>
  </div>
</div>
```

3.5 XML Response Format

An XML response contains the same report information as the other options, but also includes more detail, such as response status and message, as well as (potentially) a recap of the request information. The XML below is a draft of an XML response for a lookup. If no match is found, there will be either no <ReportList> element or an empty <ReportList> element in the response.

3.5.1 Sample XML Report Lookup Transaction Response

```
<ServiceTransaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ServiceResponse serviceName="report_lookup">
    <ResponseStatus>
      <ResponseCode>0</ResponseCode>
      <ResponseMessage>Success!</ResponseMessage>
    </ResponseStatus>
    <ReportList>
      <Report>
        <access-details>
http://informer.myinst.edu/informer/informer.html#action=reportdetails&reportid=12345
        </access-details>
        <additional-details>
          This is where you can add additional details like other requirements.
        </additional-details>
        <description>
          Show the persons name, current mailing address, and major if known
```

```
        </description>
        <display-delimiter/>
        <display-footer>Page X of Y</display-footer>
        <display-header>
          Incoming Freshman Report as of MM/DD/YYYY For entry in term TTTTTT
        </display-header>
        <display-layout>
          STUDENT NAME CITY ST ACADEMIC PROGRAM ---------------------- -------------- --- ------
---------------- Lastname, Firstname Anytown XX Mathematics
        </display-layout>
        <id type="integer">252</id>
        <name>Incoming Freshman Report</name>
        <owner>Scott Flory</owner>
        <purpose>
          To display a list of incoming freshman for distribution to professors.
        </purpose>
        <report-id type="integer">69</report-id>
      </Report>
    </ReportList>
  </ServiceResponse>
</ServiceTransaction>
```

3.5.2 Sample XML Report Termlist Transaction Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceTransaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ServiceResponse serviceName="report_termlist">
    <ResponseStatus>
      <ResponseCode>0</ResponseCode>
      <ResponseMessage>Success!</ResponseMessage>
    </ResponseStatus>
    <TermList>
      <Term>
        <id type="integer">19056</id>
        <name>Aging Period</name>
        <term-id type="integer">14757</term-id>
        <perma-link-url>http://idatau.datacookbook.local/institution/terms/14757</perma-link-url>
        <term-functional-definition>A receivable becomes aged once it is past due.  The Aging
Period for a receivable indicates how old it is.  These are broken into groups, like 0-30 days,
31-60 days, 61-120 days and &gt; 120 days.  This could be based on the AR bill date, the AR
effective date or the AR due date</term-functional-definition>
        <state>approved</state>
      </Term>
      <Term>
        <id type="integer">17103</id>
        <name>Receivable Category</name>
        <term-id type="integer">14758</term-id>
        <perma-link-url>http://idatau.datacookbook.local/institution/terms/14758</perma-link-url>
        <term-functional-definition>The category for a receivable is a way of grouping similar
transactions.
At My University, we have 5 main categories all transactions roll into:
Housing
Installment Charges
Meal Plans
Registration
Registration Fees
</term-functional-definition>
          <state>pending</state>
      </Term>
      <Term>
        <id type="integer">17104</id>
        <name>Age Date</name>
        <term-id type="integer">14759</term-id>
        <perma-link-url>http://idatau.datacookbook.local/institution/terms/14759</perma-link-url>
        <term-functional-definition>The age of a receivable.  This is used to see how old
receivables are to determine what action to take.</term-functional-definition>
         <state>approved</state>
      </Term>
    </TermList>
```

```
  </ServiceResponse>
</ServiceTransaction>
```

3.6 JSON Response Format

JSON is generally used to support a JavaScript AJAX implementation of API requests from within a browser. It uses the JavaScript call-back method of implementing cross-domain JSON AJAX requests because it is the most straightforward method of implementing cross-domain AJAX. In this method, we specify the name of a function that is invoked in our return JavaScript to tell the calling page to process the results of the request. The calling page is responsible for implementing that method. To invoke the API, the calling page implements the call-back method and then includes a JavaScript <script> tag, with the API call as the URL. The API performs the search and returns JavaScript, formatted like the sample below, which invokes the call-back function. In this sample, the function the consumer would have to implement is named "processServiceResponse()". You specify the name of the callback function using the optional jsonFunction parameter.

The JSON response contains the same granularity and level of detail found in the XML response.

3.6.1 Sample JSON Report Lookup Transaction Response


```
{         ServiceName : "report_lookup",
          ResponseStatus : {
                    ResponseCode : 0,
                    ResponseMessage : "Success!"
          },
         ReportList : [
                {
                     "Report":{
                     "name":"Incoming Freshman Report",
                     "display_header":"                                        Incoming Freshman Report
as of MM/DD/YYYY\r\n                                          \r\n
For entry in term TTTTTT\r\n",
                     "display_delimiter":"",
                     "additional_details":"This is where you can add additional details like other
requirements.",
                     "display_footer":"                                                  Page X of
Y\r\n",
                     "id":252,
                     "report_id":69,
                     "display_layout":"STUDENT NAME                  CITY             ST
ACADEMIC PROGRAM\r\n----------------------        ---------------  ---               -----------
-----------\r\nLastname, Firstname           Anytown          XX                Mathematics",

"access_details":"http://informer.myinst.edu/informer/informer.html#action=reportdetails&reportid
=12345",
                     "description":"Show the persons name, current mailing address, and major if
known","purpose":"To display a list of incoming freshman for distribution to professors.",
                     "owner":"Scott Flory"
                }
        }
     ]
}
```

3.6.2 Sample JSON Report Termlist Transaction Response


```
{
       ServiceName : "report_termlist",
       ResponseStatus : {
              ResponseCode : 0,
              ResponseMessage : "Success!"
       },
TermList :
  [
    {
      "Term":
        {
          "name":"Aging Period",
          "state":"approved",
          "term_id":14757,
          "id":19056,
          "term_functional_definition":"A receivable becomes aged once it is past due.  The Aging
Period for a receivable indicates how old it is.  These are broken into groups, like 0-30 days,
31-60 days, 61-120 days and > 120 days.  This could be based on the AR bill date, the AR
effective date or the AR due date",
          "perma_link_url":"http://idatau.datacookbook.local/institution/terms/14757"
        }
      },
      {
        "Term":
          {
            "name":"Receivable Category",
            "state":"pending",
            "term_id":14758,
            "id":17103,
            "term_functional_definition":"The category for a receivable is a way of grouping
similar transactions.\nAt My University, we have 5 main categories all transactions roll
into:\nHousing\nInstallment Charges\nMeal Plans\nRegistration\nRegistration Fees\n",
            "perma_link_url":"http://idatau.datacookbook.local/institution/terms/14758"
          }
        },
        {
          "Term":
            {
              "name":"Age Date",
              "state":"approved",
              "term_id":14759,
              "id":17104,
              "term_functional_definition":"The age of a receivable.  This is used to see how old
receivables are to determine what action to take.",
              "perma_link_url":"http://idatau.datacookbook.local/institution/terms/14759"
            }
      }
  ]
}
```