Service Login API

Brought to you by IData, Inc

Table of contents

1 Introduction

The service login service accepts a username and password and returns a generic authentication token that can be used to authenticate in subsequent service requests. It lets you authenticate against an institution's install of the Data Cookbook, then use the credentials returned to authenticate ongoing interactions with the instance, instead of having to pass login credentials with each request.

2 Request Specification

The DNS name in the URL to which you submit a request tells us the institution whose instance you want to authenticate against. Requests will be submitted over the Internet via the HTTP protocol. Our whole application is accessed securely via HTTPS, so while our framework supports both secure and non-secure requests, this API will only accept HTTPS connections. Some Data Cookbook API requests may be asynchronous. The service login API is not. Authentication information will be returned in the HTTP response to the authentication request.

2.1 URL

This service is provided at:
`https://<your_subdomain>.datacookbook.com/session/service_login`

2.2 Parameters

Required

- **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.
- **requestType** – type of request. For service login, the request type is "service_login".
- **un** – username of user who will be authenticating the request.

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.
- **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". We need to figure out the default. Default is "xml".

2.3 Format

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

2.3.1 HTTP Query String Request Format

If you choose to implement requests using form input parameters, a search request would be either an HTTP(S) GET request that has each of the non-optional parameters above in its query string, and that could also contain any of the other parameters, or an HTTP(S) POST request that has each of the non-optional parameters above stored as form inputs in the body of the request, and that could also contain any of the other parameters. You should only place each parameter in the request once. If you accidentally place a parameter there twice and the two instances have two different values, we can't guarantee which will be used in processing your request.

Sample HTTP GET query string request:

```
https://idata.datacookbook.com/session/service_login?un=jonathanmorgan
&pw=nopeeking!&requestType=service_login&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, <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. This structure could accommodate much more complicated nested parameter structures if needed (nested ParameterLists inside a <Value> element, etc.), but for now, we are just planning on implementing name-value pairs.

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="service_login">
        <ParameterList>
            <Parameter>
                <Name>un</Name>
                <Value>jonathanmorgan</Value>
            </Parameter>
            <Parameter>
                <Name>pw</Name>
                <Value>nopeeking!</Value>
            </Parameter>
            <Parameter>
                <Name>requestType</Name>
                <Value>service_login</Value>
            </Parameter>
             <Parameter>
                <Name>outputFormat</Name>
                <Value>json</Value>
            </Parameter>
        </ParameterList>
    </ServiceRequest>
</ServiceTransaction>
```

3 Response Specification

The authentication results will be the body of the HTTP(S) response to the HTTP(S) request for service login. The response can be returned in a number of formats: RSS, HTML, XML, and 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 planing on interacting with this programmatically on a server, then it makes more sense to use either RSS or XML, though some server-side languages let you parse JSON, too, and can also do a passable job treating HTML like XML since we return xhtml.

For service login, we recommend that you authenticate on a server, then embed the token that results from authentication in subsequent requests, especially any requests that might be embedded on a web page, such as those that implement a term search widget on a web page.

The authentication token returned should be included in all requests as a cookie, in the request's headers. The response will contain an XML specification of the cookie (or potentially cookies) that need to be passed with each request to tell our servers that the requester has already been authenticated.

3.1 Fields Present in all Response Formats

There will be a list of cookies that are required for authentication. For each cookie, you'll get:

- **cookie name** – Name of the cookie that should be passed with each service request.
- **cookie value** – Value that should be passed for the name above.
- **cookie expire time** – Time from now when the cookie should expire. If empty or not present, then cookie should be kept only for the current session.
- **cookie domain** – Domain value for the cookie. In most cases, will be the domain to which you submitted the request.
- **cookie path** – Path in which the cookie should be sent. If "/", then cookie should be sent with all requests, regardless of the path.
- **cookie secure flag** – "true" if cookie should only be sent with HTTPS connections, "false" if it should be sent with all requests, regardless of protocol.

3.2 RSS Response Format

In an RSS response, an RSS document would be the body of the HTTP(S) response, with each item containing a cookie needed for authentication. Our API implements RSS 2.0. For each item, the cookie name is the <title>, the full cookie string is the <description>, and the URL in <link> and <guid> will always be to the root of the datacookbook site that the login was authenticated against.

Sample RSS service transaction response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="2.0">
    <channel>
         <title>Data Cookbook Service Response: request type
service_login</title>
         <description>Data Cookbook Service Response: request type
service_login. Status Code: 0; Status Message: Success!</description>

<link>http://idata.datacookbook.com/session/service_login</link>
         <langage>en-us</langage>
         <copyright>Copyright 2010, 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, 08 Mar 2010 13:16:02 -0500</pubDate>
        <lastBuildDate>Mon, 08 Mar 2010 13:16:02 -0500</lastBuildDate>
        <ttl>30</ttl>
        <debug>array</debug>
        <item>
            <title>_itdb_session</title>
            <link>http://idata.datacookbook.com</link>

 <description><![CDATA[_itdb_session=BAh7CToMdXNlcl9pZGlDIgpmbGFzaElDO
idBY3Rpb25Db250cm9sbGVyOjpGbGFzaDo6Rmxhc2hIYXNoewY6C25vdGljZSICjwF7ImZ
sYXNoIj0+e30sIDpfY3NyZl90b2tlbj0+IjJ0bHhrOHZacTNGMnRhUlFRTmM3cDd4Tk15T
1NCbjV3MnRPcGhodGJ2ODA9IiwgOnNlc3Npb25faWQ9PiIzZjY5YjI3ZmUyNWY4MzRmNGQ
0ZGRhOTI4NDliMGFkNSJ9Q29va2llPzogQkFoN0NDSUtabXhoYzJoSlF6b25RV04wYVc5d
VEyOXVkSEp2Ykd4bGNqbzZSbXhoYzJnNk9rWnNZWE5vU0dGemFIc0FCam9LUUhWelpXUjd
BRG9RWDJOemNtWmZkRzlyWlc0aU1USjBiSGhyT0haYWNUTkdNblJoVWxGUlRtTTNjRGQ0V
GsxNVQxTkNialYzTW5SUGNHaG9kR0oyT0RBOU9nOXpaWE56YVc5dVgybGtaVV6WmpZNVl
qSTNabVV5TldZNE16Um1OR1EwWkdSaE9USTRORGxpTUdGemFIc0FCam9LUUhWelpXUjdBR
GI1OTIxNDFhM2RiMjU4NDhjNzU1ODkxMjc1M2MGOgpAdXNlZHsGOwdUOhBfY3NyZl90b2t
lbiIxMnRseGs4dlpxM0YydGFSUVFOYzdwN3hOTXlPU0JuNXcydE9waGh0YnY4MD06D3Nlc
3Npb25faWQiJTNmNjliMjdmZTI1ZjgzNGY0ZDRkZGE5Mjg0OWIwYWQ1--
11d1e4a6afab5ce50b1c181c5f6cb3bdb373f853; path=/;
domain=idata.datacookbook.local; secure=false]]></description>
            <pubDate>Mon, 08 Mar 2010 13:16:02 -0500</pubDate>
            <guid
```

```
isPermaLink="true">http://idata.datacookbook.com</guid>
        </item>
    </channel>
</rss>
```

3.3 HTML Response Format

In an HTML response, the list of cookies will be returned in a simple structure in HTML, with classes and names assigned so you could target CSS to fit their appearance to your application. This type of service doesn't really lend itself to being displayed publicly, but html is implemented for administrative and testing purposes.

Sample HTML service transaction response:

```
<div class="DATA_COOKBOOK_services">
    <h1>This service is not intended for public display.</h1>
    <div class="DC_CookieList">
        <div class="DC_Cookie">
            <div class="DC_Cookie_Name">_itdb_session</div>
            <div
class="DC_Cookie_Value">BAh7CToMdXNlcl9pZGlDIgpmbGFzaElDOidBY3Rpb25Db2
50cm9sbGVyOjpGbGFzaDo6Rmxhc2hIYXNoewY6C25vdGljZSICjwF7ImZsYXNoIj0+e30s
IDpfY3NyZl90b2tlbj0+IjJ0bHhrOHZacTNGMnRhUlFRTmM3cDd4Tk15T1NCbjV3MnRPcG
hodGJ2ODA9IiwgOnNlc3Npb25faWQ9PiIzZjY5YjI3ZmUyNWY4MzRmNGQ0ZGRhOTI4NDli
MGFkNSJ9Q29va2llPzogQkFoN0NDSUtabXhoYzJoSlF6b25RV04wYVc5dVEyOXVkSEp2Yk
d4bGNqbzZSbXhoYzJnNk9rWnNZWE5vU0dGemFIc0FCam9LUUhWelpXUjdBRG9RWDJOemNt
WmZkRzlyWlc0aU1USjBiSGhyT0haYWNUTkdNblJoVWxGUlRtTTNjRGQ0VGsxNVQxTkNial
YzTW5SUGNHaG9kR0oyT0RBOU9nOXpaWE56YVc5dVgybGtaVV6WmpZNVlqSTNabVV5TldZ
NE16Um1OR1EwWkdSaE9USTRORGxpTUdGa05RPT0tLWNiMDBjYTEwNWY0ZGI1OTIxNDFhM2
RiMjU4NDhjNzU1ODkxMjc1M2MGOgpAdXNlZHsGOwdUOhBfY3NyZl90b2tlbiIxMnRseGs4
dlpxM0YydGFSUVFOYzdwN3hOTXlPU0JuNXcydE9waGh0YnY4MD06D3Nlc3Npb25faWQiJT
NmNjliMjdmZTI1ZjgzNGY0ZDRkZGE5Mjg0OWIwYWQ1--
11d1e4a6afab5ce50b1c181c5f6cb3bdb373f853</div>
            <div class="DC_Cookie_Expires"></div>
            <div class="DC_Cookie_Path">/</div>
            <div
class="DC_Cookie_Domain">idata.datacookbook.local</div>
            <div class="DC_Cookie_Secure">false</div>
        </div>
    </div>
</div>
```

3.4 XML Response Format

An XML response contains the same list of cookies as the other options, but also includes more detail than the other options, including response status and message and potentially a re-cap of the request information. The XML sample below is an XML response for a successful authentication of jonathanmorgan. If authentication fails, there will either be no <CookieList> or an empty <CookieList> in the response.

Sample XML service transaction response:

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceTransaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <ServiceResponse serviceName="service_login">
         <ResponseStatus>
            <ResponseCode>0</ResponseCode>
            <ResponseMessage>Success!</ResponseMessage>
         </ResponseStatus>
         <CookieList>
             <Cookie>
                 <name>_itdb_session</name>

 <value>BAh7CToMdXNlcl9pZGlDIgpmbGFzaElDOidBY3Rpb25Db250cm9sbGVyOjpGbG
FzaDo6Rmxhc2hIYXNoewY6C25vdGljZSICjwF7ImZsYXNoIj0+e30sIDpfY3NyZl90b2tl
bj0+IjJ0bHhrOHZacTNGMnRhUlFRTmM3cDd4Tk15T1NCbjV3MnRPcGhodGJ2ODA9IiwgOn
Nlc3Npb25faWQ9PiIzZjY5YjI3ZmUyNWY4MzRmNGQ0ZGRhOTI4NDliMGFkNSJ9Q29va2ll
PzogQkFoN0NDSUtabXhoYzJoSlF6b25RV04wYVc5dVEyOXVkSEp2Ykd4bGNqbzZSbXhoYz
JnNk9rWnNZWE5vU0dGemFIc0FCam9LUUhWelpXUjdBRG9RWDJOemNtWmZkRzlyWlc0aU1U
SjBiSGhyT0haYWNUTkdNblJoVWxGUlRtTTNjRGQ0VGsxNVQxTkNialYzTW5SUGNHaG9kR0
oyT0RBOU9nOXpaWE56YVc5dVgybGtJaVV6WmpZNVlqSTNabVV5TldZNE16Um1OR1EwWkdS
aE9USTRORGxpTUdGa05RPT0tLWNiMDBjYTEwNWY0ZGI1OTIxNDFhM2RiMjU4NDhjNzU1OD
kxMjc1M2MGOgpAdXNlZHsGOwdUOg9zZXNzaW9uX2lkIiUzZjY5YjI3ZmUyNWY4MzRmNGQ0
ZGRhOTI4NDliMGFkNToQX2NzcmZfdG9rZW4iMTJ0bHhrOHZacTNGMnRhUlFRTmM3cDd4Tk
15T1NCbjV3MnRPcGhodGJ2ODA9--
25e354817938837967c3ff04c8d3ac6fde9d9439</value>
                <path>/</path>
                 <expires></expires>
                 <domain>idata.datacookbook.local</domain>
                 <secure>false</secure>
             </Cookie>
        </CookieList>
    </ServiceResponse>
</ServiceTransaction>
```

3.5 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, 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 that 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. For this service, however, we do not recommend performing authentication from the user's browser. It is more secure to authenticate on the server, then embed the authentication token that results in requests sent from the browser.

Sample JSON service transaction response:


```
processServiceResponse(
  {
    ServiceName : "service_login",
    ResponseStatus : {
      ResponseCode : 0,
      ResponseMessage : "Success!"
    },
    CookieList : [
      {
        name : "_itdb_session",
        value :
"BAh7CToMdXNlcl9pZGlDIgpmbGFzaElDOidBY3Rpb25Db250cm9sbGVyOjpGbGFzaDo6Rm
xhc2hIYXNoewY6C25vdGljZSICjwF7ImZsYXNoIj0+e30sIDpfY3NyZl90b2tlbj0+IjJ0bHhrOH
ZacTNGMnRhUlFRTmM3cDd4Tk15T1NCbjV3MnRPcGhodGJ2ODA9IiwgOnNlc3Npb25faW
Q9PiIzZjY5YjI3ZmUyNWY4MzRmNGQ0ZGRhOTI4NDliMGFkNSJ9Q29va2llPzogQkFoN0ND
SUtabXhoYzJoSlF6b25RV04wYVc5dVEyOXVkSEp2Ykd4bGNqbzZSbXhoYzJnNk9rWnNZWE
5vU0dGemFIc0FCam9LUUhWelpXUjdBRG9RWDJOemNtWmZkRzlyWlc0aU1USjBiSGhyT0h
aYWNUTkdNblJoVWxGUlRtTTNjRGQ0VGsxNVQxTkNialYzTW5SUGNHaG9kR0oyT0RBOU9
nOXpaWE56YVc5dVgybGtJaVV6WmpZNVlqSTNabVV5TldZNE16Um1OR1EwWkdSaE9UST
RORGxpTUdGa05RPT0tLWNiMDBjYTEwNWY0ZGI1OTIxNDFhM2RiMjU4NDhjNzU1ODkxM
jc1M2MGOgpAdXNlZHsGOwdUOg9zZXNzaW9uX2lkIiUzZjY5YjI3ZmUyNWY4MzRmNGQ0Z
GRhOTI4NDliMGFkNToQX2NzcmZfdG9rZW4iMTJ0bHhrOHZacTNGMnRhUlFRTmM3cDd4T
k15T1NCbjV3MnRPcGhodGJ2ODA9--25e354817938837967c3ff04c8d3ac6fde9d9439",
        path : "/",
```

```
            domain : "idata.datacookbook.local",
            secure : "false",
        }
    ]
  }
);
```