DataShop Public API

A way to programmatically get and set data in DataShop

The DataShop Team + others
1/15/2010
Contents
1 API ..................................................................................................................................................... 3
2 Note on format .................................................................................................................................... 3
   Method Title ......................................................................................................................................... 3
3 What is an ID? ................................................................................................................................... 3
4 Representational state transfer (REST)............................................................................................. 3
   4.1 HTTP Response Status Codes .................................................................................................... 3
5 Authentication .................................................................................................................................. 4
   5.1 Access Credentials..................................................................................................................... 4
   5.2 Using your Access Credentials to Sign a Request ..................................................................... 4
6 Getting Data ...................................................................................................................................... 9
   6.1 Get Dataset Metadata............................................................................................................... 9
   6.2 Get Sample Metadata ............................................................................................................. 12
   6.3 Get Transactions ..................................................................................................................... 14
   6.4 Get Student-Step Records ....................................................................................................... 19
Appendix A .................................................................................................................................................. 24
   All Possible Result Codes..................................................................................................................... 24
1 API
This API is a list of methods for programmatically getting and (in the future) modifying DataShop data. The initial target is to provide enough functionality for getting transaction data with various parameters, and adding, modifying, and deleting custom-field data.

As of January 2010, we've implemented:

- Authentication
- Get Dataset Metadata
- Get Sample Metadata
- Get Transactions
- Get Student-Step Records

2 Note on format

Method Title

https://url/?optional_section
URL to perform the action.

3 What is an ID?
The DataShop API expects you to reference various objects by “ID”, a unique identifier for each dataset, sample, custom field, or transaction in the repository. The ID of any of these can be determined by performing a request to list the various items, which lists the IDs in the response. For example, a request for datasets will list the ID of each dataset in the “id” attribute of each dataset element.

4 Representational state transfer (REST)
The DataShop API design generally follows a "REST" approach to web services. In doing so, we've modeled DataShop as a collection of resources which can be retrieved and manipulated using HTTP. (See the "RESTful Web Services" section of Wikipedia's REST article.)

We do deviate from the REST approach slightly in one way: the DataShop web service ignores the type of HTTP method in each request. That means the service relies on unique URLs with verbs in them (e.g., "get" and "delete") instead of distinguishing requests from similar URLs by HTTP method (e.g., GET or DELETE).

4.1 HTTP Response Status Codes
DataShop Web Services will, in many cases, return an HTTP status code that you can use to determine how to analyze the body of the HTTP message. For example, if you know the HTTP response code was an error such as “401 Unauthorized”, then you know that the content of the body of the message will be XML describing the error.

For a list of web services result codes and the HTTP status codes that appear with them, see Appendix A.
5 Authentication

Authentication is a process for identifying and verifying who is sending a request. To use DataShop web services, you need to identify yourself as the sender of each request. This is accomplished by sending a digital signature that is derived from a pair of public/private access keys.

Note: To use DataShop web services, you must first request access. Visit http://pslcdatashop.web.cmu.edu/WebServicesCredentials to read the User Agreement and request access. You will need to first register with DataShop if you haven’t already.

Note: In addition to this documentation, consult the Java source of the sample DataShop web services client for code examples. See methods signedRequest, encrypt, and httpTimestamp in particular.

5.1 Access Credentials

Access credentials refers to your two access keys. You can retrieve or reset your access keys from DataShop’s Web Services Credentials page.

Access Key ID

Your Access Key ID identifies you as the party responsible for service requests. Include it with each request you send to us.

Secret Access Key

Your Access Key ID has a Secret Access Key associated with it. Use your Secret Access Key to calculate a signature to include in requests to DataShop web services. Your Secret Access Key is a secret, and should be known only by you and DataShop. You should never include your Secret Access Key in your requests to DataShop web services. You should never email your Secret Access Key to anyone. It is important to keep your Secret Access Key confidential to protect your account.

5.2 Using your Access Credentials to Sign a Request

To authenticate to DataShop, you will:

1. Create a request
2. Create a specific type of message signature
3. Send the request and signature to DataShop Web Services

DataShop Web Services will then:

1. Retrieve your Secret Access Key
2. Create the same type of signature
3. Compare the two signatures

If the two signatures match, the request is considered authenticated; if they fail to match, then the request fails authentication.
Step 1: Create a request

In this step, you create an HTTP request in your program which has a standard request line, a number of request headers, and an optional message body (some DataShop Web Services requests require a body while others are only a URL).

To authenticate, your request must contain the following request headers:

\[
\text{date}
\]
\[
\text{authorization}
\]

Set “date” according to the timestamp format (see Table 1 - Contents of the string to sign).

You will set the value of “authorization” in Step 3, described below.

Step 2: Create a message signature

Each request you send must include an HMAC-SHA signature calculated with your Secret Access Key. **HMAC-SHA** is an industry-standard message authentication procedure that uses the **SHA-1** cryptographic hash function in combination with a secret key to create a message signature. DataShop uses it to verify the authenticity of your request.

The signature must be URL-encoded in UTF8. This signature will be included in the “authorization” header as described in Step 3.
The string you use to calculate the signature (the *string to sign*) is composed as follows:

- **method**
- **contentMD5**
- **contentType**
- **timestamp**
- **path**

Each line in the string to sign is separated by a line break. See Table 1 - Contents of the string to sign for a description of each line.

Note: See the Java source of the sample DataShop web services client for a code example.

<table>
<thead>
<tr>
<th>part of the string</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>method</strong></td>
<td>the HTTP method used—GET, PUT, POST, or DELETE</td>
<td>GET</td>
</tr>
<tr>
<td><strong>contentMD5</strong></td>
<td>an MD5 hash of the message content if PUT or POST, an empty string if GET or DELETE</td>
<td></td>
</tr>
<tr>
<td><strong>contentType</strong></td>
<td>MIME type of content if PUT or POST, an empty string if GET or DELETE</td>
<td>text/xml</td>
</tr>
<tr>
<td><strong>timestamp</strong></td>
<td>an HTTP date string (see httpTimestamp method below)</td>
<td>Tue, 20 Oct 2009 15:22:52 GMT</td>
</tr>
<tr>
<td><strong>path</strong></td>
<td>the portion of the request URL that is after “services” and before the first question mark (&quot;?&quot;).</td>
<td>/datasets/1/samples/3</td>
</tr>
</tbody>
</table>

```java
/**
 * Format for HTTP date strings.
 * @return format for HTTP date strings
 */
private String httpTimestamp() {
    if (httpDateFmt == null) {
        httpDateFmt = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss zzz");
        httpDateFmt.setTimeZone(TimeZone.getTimeZone("GMT"));
    }
    return httpDateFmt.format(new Date());
}
```

Finally, you must URL-encode the signature you’ve created in UTF-8 character encoding. This is the application/x-www-form-urlencoded MIME format. Your programming language may have a utility method that will perform this encoding for you. For reference, see the Java class `java.net.URLEncoder`, which you can use to perform UTF-8 URL-encoding in Java.

**Step 3: Send the request and signature to DataShop Web Services**

After calculating the signature and URL-encoding it, you can now attach it to the request.
Set the “authorization” header as follows, where publicApiToken is your public token and signature is the HMAC-SHA signature you create in Step 2:

“DATASHOP " + publicApiToken + ":" + signature

For example:

DATASHOP OKLFETPSJZJJFSGR6D8E:wzXDj0%2BEY3iosiwcgH%2FsYktDZPM%3D%0D%0A

Note the space after “DATASHOP”.

Send the complete request to DataShop Web Services.

5.2.1 Example
You want to retrieve a description of the first sample (“All Data”) for the dataset with id “1”.

The request line looks like this:

GET /services/datasets/1/samples/1 HTTP/1.1

The request body is empty.

You create a “date” header:

date: Tue, 20 Oct 2009 16:59:47 GMT

Then create the string to sign:

method: GET
contentMD5: <empty>
cContentType: <empty>
timestamp: Tue, 20 Oct 2009 16:59:47 GMT
path: /datasets/1/samples/1

or

GET

Tue, 20 Oct 2009 16:59:47 GMT
/datasets/1/samples/1

Create the HMAC-SHA signature from the string to sign and URL-encode it in UTF-8:

wzXDj0%2BEY3iosiwcgH%2FsYktDZPM%3D%0D%0A

And with that, form the “authorization” header:

authorization: DATASHOP OKLFETPSJZJJFSGR6D8E:wzXDj0%2BEY3iosiwcgH%2FsYktDZPM%3D%0D%0A

The full request will then look like this:
6  Getting Data

6.1  Get Dataset Metadata

http://pslcdatashop.web.cmu.edu/services/datasets/[?id]
Get list of datasets matching the parameters, or information about a single dataset (if an id is specified).

6.1.1  Request Parameters

access
all, viewable or editable. Default is viewable. All means return all the datasets, those that are public, view, edit or private. Viewable means return datasets for projects on which you have “view” or “edit” privileges, in addition to public datasets. Editable means return only datasets for projects on which you have the “edit” privilege.

Note: If you request a single dataset and specify a value for the “access” parameter, DataShop will respect the access parameter. This will result in an empty set in the case that the dataset requested does not match the access parameter specified (e.g., you specified “access=editable” but the dataset is only viewable).

verbose
true or false. Default is false. False returns the simpler, less verbose description for each dataset (see directly below) while true returns the more verbose description for each dataset (see further below). A verbose response is only possible for datasets you can view or edit (i.e., not private datasets).

6.1.2  Response Fields

access is your level of access to the dataset. DataShop models each user's dataset access as either "public", "view", "edit", or "private".

- public: you and any other registered DataShop user can view the data
- view: you can view the data, create samples, add kc models, and add kc sets.
- edit: you can both view the data and modify metadata, add or modify papers and files, and add or modify custom fields.
- private: you cannot view the data nor edit anything about the dataset

number_of_samples is the total number of samples including those that are accessible and those that are not shared.

number_of_accessible_samples is the number of samples that you can access, ignoring private samples owned by others.

number_of_unique_steps is the number of unique steps, where uniqueness is defined as a step within a specific problem hierarchy (the curriculum location where the problem appears). The same step attempted by two students equals only one unique step.
**Note:** The order of datasets in the response XML is indeterminate.

### 6.1.3 Example request:

GET https://pslcdatashop.web.cmu.edu/services/datasets/31

### 6.1.4 Example response on success:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message result_code="0" result_message="Success.">
  <dataset id="31">
    <name>Pittsburgh Science of Learning Center Stoichiometry Study 1</name>
    <project>Stoichiometry Studies</project>
    <learnlab>chemistry</learnlab>
    <pi>bleber</pi>
    <start_date>2005-09-02</start_date>
    <end_date>2006-06-07</end_date>
    <status>complete</status>
    <access>edit</access>
    <public>yes</public>
    <number_of_students>34</number_of_students>
    <number_of_unique_steps>16453</number_of_unique_steps>
    <number_of_steps>124882</number_of_steps>
    <number_of_transactions>245093</number_of_transactions>
    <number_of_samples>17</number_of_samples>
    <number_of_accessible_samples>3</number_of_accessible_samples>
    <number_of_kc_models>4</number_of_kc_models>
  </dataset>
</pslc_datashop_message>
```

### 6.1.5 Example response on error:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message result_code="-1" result_message="Error. Dataset 31 is not valid.
```

### 6.1.6 Example request (verbose)

GET https://pslcdatashop.web.cmu.edu/services/datasets/31?verbose=true

### 6.1.7 Example verbose XML response (more metadata):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message result_code="0" result_message="Success.">
  <dataset id="31">
    <name>Pittsburgh Science of Learning Center Stoichiometry Study 1</name>
    <project>Stoichiometry Studies</project>
    <learnlab>chemistry</learnlab>
    <pi>bleber</pi>
    <start_date>2005-09-02</start_date>
    <end_date>2006-06-07</end_date>
    <status>complete</status>
    <access>edit</access>
    <public>yes</public>
    <curriculum>geometry 2005</curriculum>
    <tutor></tutor>
  </dataset>
</pslc_datashop_message>
```
<description></description>
<has_study_data>yes</has_study_data>
<hypothesis>Lorem ipsum</hypothesis>
<school>Wilkinsburg High School</school>
<additional_notes>School demographics for 2005-6 unless noted otherwise.
Converter Version 4.15
Loaded to production on 4/15/08 by Kyle.</additional_notes>

<number_of_students>34</number_of_students>
<number_of_unique_steps>16453</number_of_unique_steps>
<number_of_steps>124882</number_of_steps>
<number_of_transactions>245093</number_of_transactions>
<number_of_samples>17</number_of_samples>
<number_of_accessible_samples>3</number_of_accessible_samples>
<number_of_kc_models>4</number_of_kc_models>

<kci id="25"
    <name>Automatic-Model</name>
    <number_of_kcs>456</number_of_kcs>
    <observations_with_kcs>470117</observations_with_kcs>
    <lfa_status>unable to run</lfa_status>
</kc_model>

<kci id="32"
    <name>Single-KC</name>
    <number_of_kcs>1</number_of_kcs>
    <observations_with_kcs>126057</observations_with_kcs>
    <lfa_status>complete</lfa_status>
    <aic>142117.54</aic>
    <bic>142468.24</bic>
</kc_model>

<kci id="14"
    <name>Unique-step</name>
    <number_of_kcs>9157</number_of_kcs>
    <observations_with_kcs>0</observations_with_kcs>
    <lfa_status>not scheduled to run</lfa_status>
</kc_model>
</dataset>
</pslc_datashop_message>
6.2 Get Sample Metadata

https://pslcdatashop.web.cmu.edu/services/datasets/[id]/samples/[?id]
Get the list and descriptions of samples matching the parameters.

6.2.1 Request Parameters

access
viewable or editable. Default is viewable. Viewable means return only samples you can view. Editable means return only samples you own and can therefore edit.

Note: If you request a single sample and specify a value for the “access” parameter, DataShop will respect the access parameter. This will result in an empty set in the case that the sample requested does not match the access parameter specified (e.g., you specified “access=editable” but the sample is only viewable).

verbose
ture or false. Default is false. False returns the simpler, less verbose description for each sample (see directly below) while true returns the more verbose description for each sample (see further below), including a description of each filter in each sample.

6.2.2 Example request:
GET https://pslcdatashop.web.cmu.edu/services/datasets/422/samples/37

6.2.3 Example XML response for a sample query:
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message result_code="0" result_message="Success.">
  <sample id="37">
    <name>CWCTC-GeoArea06-07</name>
    <description>Just the CWCTC students on the area units (GEO-PA)</description>
    <owner>aleven</owner>
    <number_of_transactions>350384</number_of_transactions>
  </sample>
</pslc_datashop_message>

6.2.4 Example verbose XML response for a sample query:
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message result_code="0" result_message="Success.">
  <sample id="37">
    <name>CWCTC-GeoArea06-07</name>
    <description>Just the CWCTC students on the area units (GEO-PA)</description>
    <owner>aleven</owner>
    <number_of_transactions>350384</number_of_transactions>
    <filter>
      <column>School Name</column>
      <operator>=</operator>
      <filter_text>cwctc</filter_text>
    </filter>
  </sample>
</pslc_datashop_message>
6.2.5 Example response on error:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message
    result_code="-4"
    result_message="Error. Sample 37 is not valid for dataset 422." />
```
6.3  Get Transactions

https://pslcdatashop.web.cmu.edu/services/datasets/[id]/[?samples/id]/transactions

Returns zero or more transaction records in tab-delimited form, optionally zipped. If a sample is not specified, then the “All Data” sample is returned.

See “Interacting with Cached Data” note below.

6.3.1  Request Parameters

zip
true or false. Default is false. If true, transaction data is zipped.

cols
Optional list of column names, comma-delimited. Default is to include all columns. See Table 2 - Transaction columns below for the list of column values to choose from. To specify whether or not to include custom-field columns, use the cfs parameter.

cfs
all or none. Default is none. Optionally specify whether to include all custom fields or none. (A future release of web services will support requesting a subset of custom fields by specifying a comma-delimited list of custom-field IDs).

headers
true or false. Default is true. Optionally specify whether a header row should be included in the output. If false, a header row is omitted.

limit
Maximum number of transactions to retrieve. Default is 100, maximum is 5,000.

offset
From the beginning of the data, the number of transaction rows to skip. Default is 0. For example, an offset of 0 would return rows starting with the 1st row, while an offset of 100 would return rows starting with the 101st row. Loop through transactions in batches by combining offset and limit. For example, to get rows in batches of 100, request offset=0&limit=100 in the first iteration of the loop, which will return rows 1-100. Then in the next iteration of the loop, request offset=100&limit=100, which will return rows 101-200.

6.3.2  Example request (tab-delimited format, subset of columns):
GET
https://pslcdatashop.web.cmu.edu/services/datasets/114/samples/21/transactions?cols=row,anon_student_id,session_id,time,duration,student_response_type,tutor_response_type,problem_name,step_name,attempt_at_step,outcome,selection,input,feedback,kcs

6.3.3  Example response:
See Table 4 - Example tab-delimited transaction data.
6.3.4 Example response on error:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<pslc_datashop_message
    result_code="-7"
    result_message="Error. Invalid column: feedbacks." />
```

Interacting with Cached Data

DataShop maintains a cached export file for each sample, which is used to provide data when you request it. Due to changes made in the web application or more data being logged, the cached file can fall out-of-date with what is available in the DataShop database. When this happens, a request for the sample will return an error stating that the cached file is not yet available. DataShop will then start caching the sample (if it hasn’t already). You should try your request again after some time has passed. The amount of time needed for caching varies between a few minutes and a few hours, depending on the size of the sample.
Table 2 - Transaction columns

<table>
<thead>
<tr>
<th>Column name in request</th>
<th>Column(s) contained in response</th>
<th>Data type</th>
<th>Notes on value</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>Row</td>
<td>numeric</td>
<td></td>
</tr>
<tr>
<td>anon_student_id</td>
<td>Anon Student Id</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>session_id</td>
<td>Session Id</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Time</td>
<td>date</td>
<td>yyyy-MM-dd HH:mm:ss</td>
</tr>
<tr>
<td>time_zone</td>
<td>Time Zone</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>duration</td>
<td>Duration (sec)</td>
<td>numeric</td>
<td>Null values are represented with a dot (&quot;.&quot;)</td>
</tr>
<tr>
<td>student_response_type</td>
<td>Student Response Type</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>student_response_subtype</td>
<td>Student Response Subtype</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>tutor_response_type</td>
<td>Tutor Response Type</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>tutor_response_subtype</td>
<td>Tutor Response Subtype</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>problem_hierarchy</td>
<td>A variable number of &quot;Level(Level Type)&quot; columns, each describing a curriculum hierarchy level.</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>problem_name</td>
<td>Problem Name</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>step_name</td>
<td>Step Name</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>attempt_at_step</td>
<td>Attempt At Step</td>
<td>numeric</td>
<td>Possibly empty if step name not provided in data.</td>
</tr>
<tr>
<td>outcome</td>
<td>Outcome</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>selection</td>
<td>Selection</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>action</td>
<td>Action</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>input</td>
<td>Input</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>feedback</td>
<td>Feedback Text, Feedback Classification</td>
<td>string, string</td>
<td>Both possibly empty</td>
</tr>
<tr>
<td>help_level</td>
<td>Help Level</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>total_num_hints</td>
<td>Total # Hints</td>
<td>numeric</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>condition</td>
<td>Condition Name, Condition Type</td>
<td>string, string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td></td>
<td>These two columns will appear for each condition that applies to the student (there may be more than one).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kcs</td>
<td>KC(KC Model Name)</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td></td>
<td>Includes all KCs for all models. One column per KC per model. Multiple columns per KC model in the case where more than one KC is associated with the transaction.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>school</td>
<td>School</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>class</td>
<td>Class</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
</tbody>
</table>

Note: For more documentation on these columns and the semantics of the format, see the Import File Verification Tool documentation and Export By Transaction as documented in the DataShop web application.
### Table 3 – Custom field columns included when the “cfs” parameter is ‘all’

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Data type</th>
<th>Notes on value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF(Custom Field Name)</td>
<td>A single column per custom field requested.</td>
<td>variable</td>
<td></td>
</tr>
<tr>
<td>Row</td>
<td>Anon Student Id</td>
<td>Session Id</td>
<td>Time</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------</td>
<td>--------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>1</td>
<td>D8DD5</td>
<td>D8DD5-</td>
<td>2008-01-07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jan07-15-12-09</td>
<td>15:12:18.0</td>
</tr>
<tr>
<td>2</td>
<td>D8DD5</td>
<td>D8DD5-</td>
<td>2008-01-07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jan07-15-12-09</td>
<td>15:12:45.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attempt At Step</th>
<th>Outcome</th>
<th>Selection</th>
<th>Input</th>
<th>Feedback Text</th>
<th>Feedback Classification</th>
<th>KC(Default)</th>
<th>KC(Unique-step)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CORRECT</td>
<td>MC-1</td>
<td>1</td>
<td>SELECT-MC-ANSWER</td>
<td>KC1797</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CORRECT</td>
<td>MC-2</td>
<td>2</td>
<td>SELECT-MC-ANSWER</td>
<td>KC1241</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.4 Get Student-Step Records

http://pslcdatashop.web.cmu.edu/services/datasets/[id]/[?samples/id]/steps

Returns zero or more student-step records in tab-delimited format, optionally zipped. If a sample is not specified then the “All Data” sample is returned.

See “Interacting with Cached Data” note above.

6.4.1 Request Parameters

zip

true or false. Default is false. If true, the data is zipped.

cols

Optional list of column names, comma-delimited. Default is to include all columns. See Table 5 – Student-step columns below for the list of column values.

cfs [coming soon]

all, none, [ids]. Default is all. Optionally specify whether to include all, none, or some custom fields. If all, include all custom fields. If none, no custom fields are included. If a comma-delimited list of ids, then the specified custom fields are included, e.g. cfs=123,456.

kcms

all or none. Default is all. Optionally specify whether to include all KC models or none. If all, include all KC models. If none, no KC models are included. See Table 7 below for list of columns included with each KC model. (A future release of web services will support requesting a subset of KC models by specifying a comma-delimited list of KC model IDs).

headers

true or false. Default is true. Specify whether a header row should be included in the output. If false, a header row is omitted.

limit

Maximum number of student-step rows to retrieve. Default is 100, maximum is 5,000.

offset

From the beginning of the data, the number of student-step rows to skip. Default is 0. For example, an offset of 0 would return rows starting with the 1st row, while an offset of 100 would return rows starting with the 101st row. Loop through rows in batches by combining offset and limit. For example, to get rows in batches of 100, request offset=0&limit=100 in the first iteration of the loop, which will return rows 1-100. Then in the next iteration of the loop, request offset=100&limit=100, which will return rows 101-200.
6.4.2 Example request (subset of columns):
GET
https://pslcdatashop.web.cmu.edu/services/datasets/114/samples/21/steps?cols=row,anon_student_id,condition,problem_hierarchy,problem_name,step_name,step_duration,first_attempt&kcms=all

6.4.3 Example response:
See Table 7 - Example tab-delimited step data.

6.4.4 Example response on error:
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message
    result_code="-7"
    result_message="Error. Invalid column: time_zones."
/>
<table>
<thead>
<tr>
<th>Column name in request</th>
<th>Column(s) contained in response</th>
<th>Data type</th>
<th>Notes on value</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>Row</td>
<td>numeric</td>
<td></td>
</tr>
<tr>
<td>anon_student_id</td>
<td>Anon Student Id</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
| problem_hierarchy      | Problem Hierarchy
Single column (different than Get Transactions). | string    |                |
| problem_name           | Problem Name                   | string    |                |
| problem_view           | Problem View                   | numeric   |                |
| step_name              | Step Name                      | string    |                |
| step_start_time        | Step Start Time                | date      | yyyy-MM-dd HH:mm:ss |
| first_transaction_time | First Transaction Time         | date      | “             |
| correct_transaction_time| Correct Transaction Time      | date      | “             |
| step_end_time          | Step End Time                  | date      | “             |
| step_duration          | Step Duration (sec)            | numeric   |                |
| correct_step_duration  | Correct Step Duration (sec)    | numeric   |                |
| error_step_duration    | Error Step Duration (sec)      | numeric   |                |
| first_attempt          | First Attempt                  | string    | Possibly empty |
| incorrects             | Incorrects                     | numeric   | Possibly empty |
| hints                  | Hints                          | numeric   | Possibly empty |
| corrects               | Corrects                       | numeric   | Possibly empty |
| condition              | Condition
In the case of a student assigned to multiple conditions (factors in a factorial design), condition names are separated by a comma and space. This differs from the format in Get Transactions. | string    |                |

**Note:** For more documentation on these columns and the semantics of the format, see [Student-Step Rollup](#) as documented in the DataShop web application.
Table 6 – KC model columns included when requested using the "kcms" parameter

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Data type</th>
<th>Notes on value</th>
</tr>
</thead>
<tbody>
<tr>
<td>KC(KC Model Name)</td>
<td>In the case of multiple KCs assigned to a single step, KC names are separated by two tildes (&quot;~~&quot;).</td>
<td>string</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>Opportunity(KC Model Name)</td>
<td>Opportunity number for the listed KC(s). In the case of multiple KCs assigned to a single step, opportunity number values are separated by two tildes (&quot;~~&quot;) and are given in the same order as the KC names.</td>
<td>numeric</td>
<td>Possibly empty</td>
</tr>
<tr>
<td>Predicted Error Rate(KC Model Name)</td>
<td>Predicted error rate for the listed KC(s). In the case of multiple KCs assigned to a single step, predicted error rate values are separated by two tildes (&quot;~~&quot;) and are given in the same order as the KC names.</td>
<td>numeric</td>
<td>Possibly empty</td>
</tr>
</tbody>
</table>

**Note:** For each KC model, the three columns listed in the above table will appear in the output.

**Important:** The format of the KC model columns returned by web services Get Student-Step Records is different from the format of these columns in the web application and from the Get Transactions web service. In the web application’s current step format, multiple KCs associated with a step results in **multiple rows**. In the web services version, multiple KCs are contained in a **single value** and delimited with two tildes ("~~"), resulting in a single row for the student-step. The same rule is applied to the Opportunity and Predicted Error Rate columns.
Table 7 - Example tab-delimited step data

<table>
<thead>
<tr>
<th>Row</th>
<th>Anon Student Id</th>
<th>Condition</th>
<th>Problem Hierarchy</th>
<th>Problem Name</th>
<th>Step Name</th>
<th>Step Duration (sec)</th>
<th>First Attempt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D8DD5</td>
<td>a</td>
<td>Unit 1, Section 2</td>
<td>CHARGE1A</td>
<td>(CHOOSE-ANSWER MC-1 1)</td>
<td>5</td>
<td>correct</td>
</tr>
<tr>
<td>2</td>
<td>D8DD5</td>
<td>a</td>
<td>Unit 1, Section 2</td>
<td>CHARGE1A</td>
<td>(CHOOSE-ANSWER MC-2 2)</td>
<td>27</td>
<td>correct</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KC(Default)</th>
<th>Opportunity(Default)</th>
<th>Predicted Error Rate(Default)</th>
<th>KC(Unique-step)</th>
<th>Opportunity(Unique-step)</th>
<th>Predicted Error Rate(Unique-step)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT-MC-ANSWER~~APPLY-THEOREM</td>
<td>4~3</td>
<td>0.553~0.754</td>
<td>KC1797</td>
<td>2</td>
<td>0.876</td>
</tr>
<tr>
<td>SELECT-MC-ANSWER~~APPLY-THEOREM</td>
<td>5~4</td>
<td>0.451~0.750</td>
<td>KC1241</td>
<td>3</td>
<td>0.639</td>
</tr>
</tbody>
</table>
# Appendix A

## All Possible Result Codes

<table>
<thead>
<tr>
<th>Result Code</th>
<th>HTTP Status Code</th>
<th>Description</th>
<th>Services that could return this result code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>200 OK</td>
<td>Success. [Optional message]</td>
<td>All</td>
</tr>
<tr>
<td>-1</td>
<td>404 Not Found</td>
<td>Error. Dataset [id] is not valid.</td>
<td>All</td>
</tr>
<tr>
<td>-2</td>
<td>403 Forbidden</td>
<td>Error. Dataset [id] is not accessible.</td>
<td>All</td>
</tr>
<tr>
<td>-3</td>
<td>404 Not Found</td>
<td>Error. Sample [id] is not valid for dataset [id].</td>
<td>Get Samples, Get Transactions, Get Student-Steps, Get Custom Field Data</td>
</tr>
<tr>
<td>-4</td>
<td>401 Unauthorized</td>
<td>Error. Sample [id] is not accessible for dataset [id].</td>
<td>Get Samples, Get Transactions, Get Student-Steps, Get Custom Field Data</td>
</tr>
<tr>
<td>-5</td>
<td>400 Bad Request</td>
<td>Error. Invalid request parameter: [parameter].</td>
<td>All</td>
</tr>
<tr>
<td>-6</td>
<td>400 Bad Request</td>
<td>Error. Invalid value for parameter [param]: [value].</td>
<td>All</td>
</tr>
<tr>
<td>-7</td>
<td>400 Bad Request</td>
<td>Error. Invalid column: [column].</td>
<td>Get Transactions, Get Student-Steps</td>
</tr>
<tr>
<td>-8</td>
<td>400 Bad Request</td>
<td>Error. Required field(s) missing: [fields].</td>
<td>Add Custom Field, Modify Custom Field Name, Modify Custom Field Description, Set Custom Field</td>
</tr>
<tr>
<td>-9</td>
<td>404 Not Found</td>
<td>Error. Custom Field [id] is not valid for dataset [id].</td>
<td>All Custom Field services</td>
</tr>
<tr>
<td>-10</td>
<td>400 Bad Request</td>
<td>Error. Invalid data.</td>
<td>Set Custom Field</td>
</tr>
<tr>
<td>-11</td>
<td>404 Not Found</td>
<td>Error. No custom field data found.</td>
<td>Delete Custom Field</td>
</tr>
<tr>
<td>-12</td>
<td>401 Unauthorized</td>
<td>Error. Not owner.</td>
<td>Delete Custom Field</td>
</tr>
<tr>
<td>-13</td>
<td>409 Conflict</td>
<td>Error. Custom field with name [name] already exists for this dataset.</td>
<td>Add Custom Field</td>
</tr>
<tr>
<td>-14</td>
<td>404 Not Found</td>
<td>Error. [Transactions/Student-steps] cache file for sample [N] [does not exist / is out-of-date]. Caching process is starting. Try again later.</td>
<td>Get Transactions, Get Student-Steps</td>
</tr>
<tr>
<td>-99</td>
<td>404 Not Found</td>
<td>Error. No web service found matching the URL. For a list of valid URLs, see <a href="http://pslcdatashop.org/api/">http://pslcdatashop.org/api/</a></td>
<td>All</td>
</tr>
<tr>
<td>-100</td>
<td>500 Internal Server Error</td>
<td>Unknown error.</td>
<td>All</td>
</tr>
<tr>
<td>-101</td>
<td>401 Unauthorized</td>
<td>Authorization failed. Check your credentials.</td>
<td>All</td>
</tr>
<tr>
<td>Code</td>
<td>Status Code</td>
<td>Reason</td>
<td>Scope</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>-102</td>
<td>406 Not Acceptable</td>
<td>This content is available only as [content type].</td>
<td>All</td>
</tr>
</tbody>
</table>
| -103  | 405 Method Not Allowed | Operation not supported.  
(Note to reader: this is for unsupported PUT/POST/DELETE methods.) | All    |
| -104  | 405 Method Not Allowed | [Method] requests not supported.  
(Note to reader: this is for methods head, options, etc.) | All    |