HathiTrust Data API - version 2

0.1 Aug. 15, 2013 pfarber
Revised from version 1 documentation. **As of November 1, 2013, version 2 of the API supersedes version 1 and access to version 1 will terminate.**
0.2 Oct. 1, 2014 pfarber
Revised to update access authorization model to support access profiles in the rights database.

Introduction

This document describes a RESTful API that provides access to HathiTrust repository data and metadata resources. The HathiTrust Repository Data API is referred to simply as "*API*" in this document.

Related Applications

- API web client at `http://babel.hathitrust.org/cgi/htdc`
- Key Generation Service (KGS) at `http://babel.hathitrust.org/cgi/kgs/request`

Please refer to companion documentation at `http://www.hathitrust.org/data_api` for details on these applications.

Overview

Description

The API provides extensible, efficient and secure access to the data and metadata resources of the HathiTrust Repository. Access to the API is available either through a web client or programmatically.

The design is intended to support client applications that already have an item identifier and simply need the corresponding data (or metadata). It should make services and uses possible beyond those available through current applications. Examples of current applications are the HathiTrust Collection Builder and Pageturner.

The API accepts one ID per request. Applications that need a number of metadata records or data sets must request them one at a time. Another option is to have a dataset created. See http://www.hathitrust.org/datasets for details.

The repository resources consist primarily of digitized print or born-digital volumes composed of page images and OCR text and corresponding structural and administrative metadata. Other APIs and downloadable files provide sources of identifiers and bibliographic metadata. Examples include:

- Files of HathiTrust volume identifiers which can be downloaded.
- OAI at the University of Michigan
- HathiTrust Bibliographic API

The API accepts a request for a resource and returns XML, JSON or binary representations of the resource. The available representations depend on the resource in question.

The resources served by the API are partitioned into classes that have varying access policies and to which throttling applies by default. Refer to the section on resources for full details.

- **Metadata** resources - Accessible without restriction.
    - Examples: `type`, **`article/meta`**, **`volume/pagemeta`**, **`structure`**
- **Data** (content) resources - Access varies.
    - Examples: **`article`**, **`volume/pageimage`**, **`aggregate`**. Unrestricted data, such as public domain volumes digitized by Internet Archive, are available to

download.
    - Some forms of data are available for download only from an IP address approved by HathiTrust for access. Example: Google-digitized public domain volumes.
    - In-copyright data is only available through the API under special contract.

Uses

The API is meant for burst activities and not large-scale retrieval of content (e.g., for datasets). Retrieval of volume metadata and content subject to throttling can serve a variety of general purposes.

It is also possible to use the API for extended purposes that require an agreement with HathiTrust. Please contact us to determine the suitability of the API for intended uses. Partners have expressed the desire, and HathiTrust wishes to support, additional options that facilitate extended activities. Some examples of these activities include the following:

Preservation uses of in-copyright content:

1. A partner retrieves single pages of an in-copyright volume to insert into a physical volume
2. A partner retrieves a whole volume in order to make a print replacement copy

Validation of public domain and in-copyright content

1. A partner performs external validation of in-copyright and public domain archival packages (the full package for some, e.g., Google-digitized, content is not currently available through the API)

Development and some data retrieval purposes for publicly available content

1. Building and testing new interfaces to content
2. Identifying materials from a particular scanning source
3. Enabling "crawling" or other means of indexing the full-text of all or a large subset of public domain materials

API Details

URL scheme

This section describes the basic form of the API URL.

```
http[s]://babel.hathitrust.org/cgi/htd/:RESOURCE/:ID[/:FILEID|:SEQ]
      [?:QUERY_STRING]
```

The values of **`:RESOURCE`** (and supported `format` values based on the given resource) are listed below.

The **`:ID`** ranges over the all namespace-qualified barcodes or other logical identifiers for repository objects. Examples of namespaces are `mdp, miun, wu`.

The **`:SEQ`** variable is an integer starting at 1 and ranges up to the number of page images in a `volume`-type resource or supporting assets in an `article`-type resource. The **`:FILEID`** is an alternative to the **`:SEQ`** and can be obtained from the **`structure`** resource if not available out-of-band.

The **`:QUERY_STRING`** parameters are as follows.

N.B. This list does not include the OAuth URL parameters required when writing a program to access the API. Also, programmatic clients must specify the query parameter **`v=2`** in their request URL. The version 1 `alt` query parameter is replaced in version 2 by `format=json`.

- **`v`** - API version. required. currently only `v=2` is valid
- **`format`** – as indicated below by resource. optional.
- **`width`** – width in pixels for image derivatives. optional.
- **`height`** – height in pixels for image derivatives. optional.
    - Either `width` or `height` is sufficient. If both are supplied, the best fit is returned; the aspect ratio is not altered.
- **`res`** – alternative to width or height. optional. `0,2,4,8` where `0`=largest. default `0`.
- **`size`** - alternative to width or height. optional. a percentage of the maximum size.
- **`watermark`** – watermark the derivative. optional. `0 | 1`. default `1`.

The `width, height, size` and `res` query parameters are alternate ways of expressing the required size for a derivative page image. If none of these parameters are supplied, the default `size=100` is delivered. `width` and `height` are in pixels. `res` is the power of 2 resolution to extract from a `jp2`. Zero (`0`) is the highest resolution. Sizes for archival images in `tiff` or `jpeg` format are generated by emulating power of 2 resolution for `jp2` extraction. `size` is a percentage of the size of the maximum resolution available for the image.

When `application/xml` mimetype applies to a response, `format=json` requests the response in JSON.

Resource types

This section lists the resources currently available through both the API and API web client. Documentation for the web client is available in the API companion document on the API page at `http://www.hathitrust.org/data_api`.

The resources listed below can be retrieved from the API through your browser using the web client at `http://babel.hathitrust.org/cgi/htdc`.

N.B. The following API example URLs do not show the required additional OAuth URL parameters that a client program must supply. Refer to the section on writing a program to access the API for more information.

Common

The `volume`-type and `article`-type resources have these resources available in common.

- `aggregate` (zip file)
  - `format: none`
  - example
    - `http[s]://babel.hathitrust.org/cgi/htd/`**`aggregate`**`/:ID?v=2`
- `structure` (METS)
  - `format: xml | json` (default: `xml`)
  - example
    - `http[s]://babel.hathitrust.org/cgi/htd/`**`structure`**`/:ID?format=xml&v=2`

Volume-type resources

This resource type can be characterized as a bound volume like a monograph or multi-part work that has been digitized, consisting of page images and OCR text.

- `volume`
  - `format:` current: `ebm`, future: `pdf | epub`
  - (currently restricted to the Espressnet project, `format=ebm`)
  - example
    - `http[s]://babel.hathitrust.org/cgi/htd/`**`volume`**`/:ID?format=ebm&v=2`
- `volume/meta`
  - `format: xml | json` (default: `xml`)
  - example
    - `http[s]://babel.hathitrust.org/cgi/htd/`**`volume`**`/`**`meta`**`/:ID?format=json&v=2`
- `volume/pagemeta`
  - `format: xml | json` (default: `xml`)
  - example
    - `http[s]://babel.hathitrust.org/cgi/htd/`**`volume`**`/`**`pagemeta`**`/:ID/123?v=2`
- `volume/pageimage`
  - format: `raw | png | jpeg`
  - A watermarked derivative is the default `volume/pageimage` resource in either `png` or `jpeg` format derived from `tiff` or `jp2` archival images. The `raw` format requires higher authorization.
  - example (for **`:SEQ`** = 4)

        - `http[s]://babel.hathitrust.org/htd/`**`volume`**`/`**`pageimage`**`/:ID/4?format=jpeg&watermark=0&v=2`
- `volume/pageocr`
    - `format: none`
    - example (for `:`**`SEQ`** = 7)
        - `http[s]://babel.hathitrust.org/htd/`**`volume`**`/`**`pageocr`**`/:ID/7?v=2`
- `volume/pagecoordocr`
    - `format: none`
    - example (for `:`**`SEQ`** = 42)
        - `http[s]://babel.hathitrust.org/htd/`**`volume`**`/`**`pagecoordocr`**`/:ID/42?v=2`

Article-type resources [in development]

This resource type is characterized as a single, born-digital journal article in XML adhering to the JATS DTD.

- `article`
    - `format:` current: `xml`, in development: `pdf, epub`
    - example
        - `http[s]://babel.hathitrust.org/htd/`**`article`**`/:ID?format=pdf&v=2` (default: `xml`)
- `article/alternate`
    - `format: none`
    - example
        - `http[s]://babel.hathitrust.org/htd/`**`article`**`/`**`alternate`**`/:ID/1?v=2`
- `article/assets/embedded`
    - format: `none`
    - example
        - `http[s]://babel.hathitrust.org/htd/`**`article`**`/`**`assets`**`/`**`embedded`**`/:ID/12?v=2`
- `article/assets/supplementary`
    - format: `none`
    - example (using **`:FILEID`** obtained from the **`structure`** resource)
        - `http[s]://babel.hathitrust.org/htd/`**`article`**`/`**`assets/supplementary`**`/:ID/ASSETIMG_I1566?v=2`

Meta-resource

Version 2 syntax defines a meta-resource: **`type`** with the following syntax:

`http[s]://babel.hathitrust.org/cgi/htd/`**`type`**`/`**`:ID`**

- `type`
  - `format: xml | json (`default`: xml)`
  - the response to the request for the `type` resource is **`volume`** or **`article`.**

This meta-resource makes it possible for the API client to determine the resource type associated with an **`:ID`** which may not be available through out-of-band knowledge. Knowing the `type` is necessary to construct a valid resource request url, e.g. for resources such as `volume/pageimage` or `article/assets/embedded.`

Metadata Schema, Extension Elements and Values

The schema for the API metadata resources (`volume/meta, volume/pagemeta, article/meta`) is based on the Atom Syndication Format in the spirit of the response schema for a volume from the Google Book Search Data API.

XML responses are formatted as **`atom:entry`** elements in the default **`atom`** namespace. The required **`atom:id, atom:title, atom:updated`** elements are present. The API schema extends the Atom schema by defining and using the **`htd`** namespace.
Note that the use of the **`atom:entry`** element is adopted in the context of access to data and not of access to a feed.

The API is a *data* API that provides accompanying structural and administrative metadata. It is not a bibliographic metadata API. The **`atom:title`** element contains text that describes the entry and is *not* the title of the book. For example,

```
HathiTrust Repository Data API - single page metadata
```

The metadata schema employs a URI scheme for additional values of the **`atom:link[@rel]`** attribute. For resource identifiers we have:

- `http://schemas.hathitrust.org/htd/2009#volume_meta`
- `http://schemas.hathitrust.org/htd/2009#volume_pagemeta`
- `http://schemas.hathitrust.org/htd/2009#volume`
- `http://schemas.hathitrust.org/htd/2009#volume_pageimage`
- `http://schemas.hathitrust.org/htd/2009#volume_pageocr`
- `http://schemas.hathitrust.org/htd/2009#volume_pagecoordocr`
- `http://schemas.hathitrust.org/htd/2009#article_assets_embedded`
- `http://schemas.hathitrust.org/htd/2009#article_assets_supplementary`
- `http://schemas.hathitrust.org/htd/2009#article`
- `http://schemas.hathitrust.org/htd/2009#article_alternate`
- `http://schemas.hathitrust.org/htd/2009#structure`
- `http://schemas.hathitrust.org/htd/2009#aggregate`

The optional element **`atom:link`** appears with the **`rel=alternate`** and **`rel=self`** attributes.

- **`link[@rel="alternate"]`** - Generally taken to mean the permalink to the content pointed to by the entry. Currently this includes a link to the HathiTrust pageturner which is quasi-permanent and a link to the Handle Server for the given item. For example, `http://babel.hathitrust.org/cgi/pt?id=:ID[&seq=:SEQ]` and `http://hdl.handle.net/2027/:ID`
- **`link[@rel="self"]`** - This is the preferred URI for retrieving the entry itself. This value is important in scenarios where only the entry is available and not the location from which the entry was retrieved.

Extension Elements

Extension elements are in the **`htd`** namespace and vary with response. That are:

- `htd:version` - the version number of the API generating the response
- `htd:selected_seq` - the page sequence number requested. (`volume/pagemeta` resource only.)
- `htd:numpages` - the number of pages in a `volume`-type resource
- `htd:num_embedded_assets` - number of inline images in an `article`-type resource
- `htd:num_supplementary_assets` – number of associated external files for an `article`-type resource
- `htd:num_articles` - number of articles in an `article`-type resource
- `htd:num_article_alternates` - number of alternative representations for and `article`-type resource.
- `htd:access_use_statement` – the full text of the Access and Use statement stating the permitted uses and rights to access this item as determined by the `htd:rights/htd:attr` and `htd:rights/htd:source` values.
- `htd:access_use` - a URI equivalent to the `htd:access_use_statement` with one of the following values. Please refer to the Access and Use page for explanations of these values.
  - `http://schemas.hathitrust.org/htd/2009#pd`
  - `http://schemas.hathitrust.org/htd/2009#pd-google`
  - `http://schemas.hathitrust.org/htd/2009#pd-us`
  - `http://schemas.hathitrust.org/htd/2009#pd-us-google`
  - `http://schemas.hathitrust.org/htd/2009#oa`
  - `http://schemas.hathitrust.org/htd/2009#oa-google`
  - `http://schemas.hathitrust.org/htd/2009#section108`
  - `http://schemas.hathitrust.org/htd/2009#ic`
  - `http://schemas.hathitrust.org/htd/2009#cc-by`
  - `http://schemas.hathitrust.org/htd/2009#cc-by-nd`
  - `http://schemas.hathitrust.org/htd/2009#cc-by-nc-nd`
  - `http://schemas.hathitrust.org/htd/2009#cc-by-nc`

    - `http://schemas.hathitrust.org/htd/2009#cc-by-nc-sa`
    - `http://schemas.hathitrust.org/htd/2009#cc-by-sa`
    - `http://schemas.hathitrust.org/htd/2009#cc-zero`
    - `http://schemas.hathitrust.org/htd/2009#und-world`
- `htd:access[@resource]` - a URI that asserts the access level for downloading images, XML, PDF, OCR and zipped data. Restricted downloading access does not necessarily imply restricted viewability in certain contexts or via other HathiTrust web applications such as the HathiTrust PageTurner.
    - `http://schemas.hathitrust.org/htd/2009#open`
    - `http://schemas.hathitrust.org/htd/2009#restricted`
- `htd:rights` - container element for rights metadata child elements populated directly from the rights database. See HathiTrust rights database document: for documentation on these database fields.
    - `htd:namespace` - the namespace of the **`:ID`** (where `:ID` is the dotted concatenation of `htd:namespace` and `htd:id` values)
    - `htd:id`
    - `htd:attr`
    - `htd:reason`
    - `htd:source`
    - `htd:user`
    - `htd:time`
    - `htd:note`
- `htd:pgmap` - container element for page number to page sequence number map
    - `htd:pg[@pgnum]` - the mapping element. attribute is page number, content is page sequence number. one for each page number.
- `htd:seqmap` - container element for map of page sequence number to page number, feature, format.
    - `htd:seq[@pseq]` - attribute is the sequence number of the page, content is the page number
    - `htd:pnum` - the page number either printed or implicit (if available)
    - `htd:imgfmt` - format of the archival page image: `tiff` or `jp2` or `jpeg`
    - `htd:pfeat` - the page feature key (if available):
        - `CHAPTER_START`
        - `COPYRIGHT`
        - `FIRST_CONTENT_CHAPTER_START`
        - `FRONT_COVER`
        - `INDEX`
        - `REFERENCES`
        - `TABLE_OF_CONTENTS`
        - `TITLE`

Programmatic Access

This section describes the 2-legged OAuth mechanism that supports secure access to the API. The mechanism provides API clients with credentials the API uses to authenticate the client's identity and authorize its access to repository resources. The Key Generation Service (KGS) provides users with a registration point and OAuth key pair delivery. Refer to companion documentation at http://www.hathitrust.org/data_api regarding KGS.

Functional Elements

The API security implementation consists of these elements:

- A specification that describes exactly how a client should generate a signed API request URL.
- Generation and transmission of a public **`oauth_consumer_key`** (client ID / access key) and a **`oauth_consumer_secret`** shared between the API and its client. The Key Generation Service (KGS) supports the transmission of the keys to the user/developer.
- An authentication database that stores the `oauth_consumer_key` and `oauth_consumer_secret` and assorted client data.
- An authorization database that associates `oauth_consumer_key` with authorization to API resources.
- Logging, monitoring and reporting

Making a Signed API Request

The API client is required to sign the API request URL for all metadata and data resources. Possessing a registered `oauth_consumer_key` and `oauth_consumer_secret` do not automatically give greater access privileges. Greater privileges require a contractural agreement to be negotiated with HathiTrust.

A signed API request involves the construction and signing of the request URI and its authentication and authorization by the API.

The client program constructs a request URL from the basic request URL, adding the plain-text `oauth_consumer_key, oauth_signature, oauth_nonce, oauth_timestamp, oauth_version` and `oauth_signature_method` to the query parameters.

An example of a signed Data API resource request URI that obeys the OAuth 1.0 specification follows. N.B. this is only an example intended to illustrate a signed URL. The `oauth_*` parameter values are not valid for actual use.

```
http://babel.hathitrust.org/cgi/htd/volume/pagemeta/mdp.39015000000128
/12?oauth_consumer_key=23f9457e2&oauth_nonce=192ed4d53e27e5d2dcd1&oaut
h_signature=HIiQ13Vm0WuZeXKl6qxzgLqxmtI%3D&oauth_signature_method=HMAC
-SHA1&oauth_timestamp=1338838461&oauth_version=1.0
```

The good specification for constructing an OAuth 1.0 signed URL can be found at http://oauth.googlecode.com/svn/spec/ext/consumer_request/1.0/drafts/1/spec.html

The dialogue between client program and Data API proceeds as follows.

1. Client signs the constructed URL with the `oauth_consumer_secret` and adds the `oauth_signature` to the query parameters.
2. API receives the signed request.
3. API looks up the client's `oauth_consumer_secret` in the authorization database and uses it to sign the plain-text URI as the `oauth_signature`.
4. API compares its `oauth_signature` with the `oauth_signature` carried by the client's API request URL.
    a. If signatures match, API looks up the privileges associated with the oauth_consumer_key in the authorization database.
        i. If there is sufficient privilege for the requested resource it is delivered in the API response.
        ii. If insufficient privilege exists, API responds with authorization failure status.
    b. If signatures do not match, API responds with authentication failure status.

Data API Response Codes

Code	Explanation
`200 OK`	No error. The request to retrieve the resource was successful.
`400 BAD REQUEST`	Invalid request URI or HTTP header, or unsupported parameter.
`400 parameter_rejected`	One or more URL parameters in the QUERY_STRING did not correspond to the required OAuth parameter set or the optional Data API parameter set.
`303 SEE OTHER`	This redirect will be issued when a resource is restricted and the request protocol is not HTTPS. The redirect URL to repeat the request is returned in the HTTP header. It will not contain the required OAuth parameters. These must be regenerated on the client side based on the redirect URL.
`401 UNAUTHORIZED`	This code will be returned when the OAuth

	signature, timestamp or nonce are invalid or when one or more required OAuth parameters are missing.
`403 FORBIDDEN`	This code will be returned when access key (`oauth_consumer_key`) is not associated with an authorization level sufficient to grant access to the requested resource.
`404 NOT FOUND`	Resource identified by `:ID` or `:ID/{:FILEID|:SEQ}` not found.
`500 INTERNAL SERVER ERROR`	Internal error. This is the default code that is used for all unrecognized errors.
`503 SERVICE UNAVAILABLE`	Quota exceeded.

Client Implementation Details

All resource request URLs must be signed. This is mandatory as of 1 October, 2012.

Following registration with KGS, the developer's access key carries default authorization allowing access to resources categorized by the **`open`** access type. All metadata resources are categorized as **`open`**. See Appendix C regarding authorization levels.

The value of the **`htd:access[@resource=":RESOURCE"]`** element in the **`{volume|article}/meta`** and **`volume/pagemeta`** resources for a given resource, indicates the restrictions on that resource. Refer to Appendix C for details regarding these access values. Values vary from item to item, and across resource types, most often depending on the digitization source and limitations imposed on the distribution of data by the source. Example: Google-digitized volumes in the **aggregate** form are always "restricted," requiring special authorization. However, in individual **volume/pageimage** form, they may be "open" not requiring special authorization.

`htd:access` URN value	**Explanation**
`http://schemas.hathitrust.org/htd/2009#restricted`	Higher level of authorization is required, established through an out-of-band contract negotiation.
`http://schemas.hathitrust.org/htd/2009#open`	Open, default authorization is

	granted.

URLs requesting resources categorized as `restricted` values must use **`https`** protocol. API clients that request restricted resources over **`http`** must be prepared to handle a `303` response code and regenerate a signed URL from the URL returned in the `Location` field of the HTTP header. The simple approach is to always use **`https`**.

Signing

The signing code is based on Perl `OAuth::Lite::Consumer` available from CPAN at `http://search.cpan.org/dist/OAuth-Lite/`. URL signing uses the Hash-based Message Authentication Code HMAC-SHA1 algorithm. One source of code libraries in other languages that implement OAuth using HMAC-SHA1 to sign URLs can be found at `http://code.google.com/p/oauth/`. The key generation code is based on `OAuth::Lite::Util` also available from CPAN as above.

Sample Client and Server

See Appendix A for a code sample that implements a Perl CGI client that will invoke our test server. It will run out of the box after you install `OAuth::Lite::Consumer` and `OAuth::Lite::AuthMethod`.

The sample client can be invoked on our servers as:
`http://babel.hathitrust.org/cgi/htdc/dapiclient`

A sample server that the `dapiclient` talks to by default can also be invoked by your own client at this address:
`http://babel.hathitrust.org/cgi/htdc/dapiserver`

Appendices

Appendix A: Data API Sample Client

This is a sample client written in Perl that uses the `OAuth::Lite` package from CPAN. It can be run on our servers as `http://babel.hathitrust.org/cgi/htdc/dapiclient` or on your own server in a Perl environment with `OAuth::Lite` installed.

The sample client is provided only to suggest a model for developing a fully functional client. It is not designed to function as a fully functional client out of the box nor is the dapiserver capable of serving the resources available from the the full Data API server at http://babel.hathitrust.org/cgi/htd.

To use the sample client to make requests to the full Data API (ratther than the sample dapiserver), you would, minimally, edit the code to supply your actual key and secret, change the $request_url to talk to the full Data API at http://babel.hathitrust.org/cgi/htd and append a pathinfo string to specify the desired resource and its ID as documented in the body of this document. Examples of alternative client development paths also appear in the Perl `OAuth::Lite` CPAN package linked above.

```
#!/usr/bin/env perl

=head1 NAME

dapiclient

=head1 DESCRIPTION

This is an example perl 2-legged oauth client that shares the secret
key "PUBLIC_OAUTH_CONSUMER_SECRET" with dapiserver.  It is intended to
aid development of a fully function Data API client in Perl or other
languages that implement HMAC_SHA1 OAuth libraries.

=head1 SYNOPSIS

For example:

http://yourhost/path_to_client/dapiclient

=head1 OUTPUT

[CLIENT] sent this URL to server:

http://babel.hathitrust.org/cgi/htd/dapiserver?hello=world&oauth_consumer_key=PUBLIC_OAUTH_CONSUM
ER_KEY&oauth_nonce=47b8186be439110b4f98&oauth_signature=2cQYAM%2BYek%2BiOexZKMObM%2B3B2w4%3D&oaut
h_signature_method=HMAC-SHA1&oauth_timestamp=1332184191&oauth_version=1.0

[CLIENT] received this HTTP response from server:
  200 OK

[CLIENT] received this content response from server:

        [SERVER] received client request. echoing request parameters:

        hello
        world
        oauth_consumer_key
        PUBLIC_OAUTH_CONSUMER_KEY
        oauth_nonce
        47b8186be439110b4f98
        oauth_signature
        2cQYAM+Yek+iOexZKMObM+3B2w4=
        oauth_signature_method
        HMAC-SHA1
        oauth_timestamp
        1332184191
        oauth_version
        1.0
```

```
=cut

use strict;
use warnings;

use CGI;
use OAuth::Lite::Consumer;
use OAuth::Lite::AuthMethod;



my $access_key = 'PUBLIC_OAUTH_CONSUMER_KEY';
my $secret_key = 'PUBLIC_OAUTH_CONSUMER_SECRET';

my $request_url = 'http://babel.hathitrust.org/cgi/htd/dapiserver';

my $consumer = OAuth::Lite::Consumer->new
 (
  consumer_key    => $access_key,
  consumer_secret => $secret_key,
  auth_method     => OAuth::Lite::AuthMethod::URL_QUERY,
 );

my $response = $consumer->request
 (
  method  => 'GET',
  url   => $request_url,
  params  => {
               'hello' => 'world',
              },
 );

print CGI::header();

print "<p><b>[CLIENT] sent this URL to server:</b><br/>";
print $consumer->oauth_request->uri;

print "<p><b>[CLIENT] received this HTTP response from server:</b><br/>";
print $response->status_line;
if ($response->is_success) {
       print "<br/><b>[CLIENT] received this content response from server:</b><blockquote>" .
$response->content . "</blockquote>";
}

exit 0;
```

Appendix B: Items Determined to be in the Public Domain only in the U.S. or only outside the U.S.

Some data resources may be categorized as **`open`** because the item is in the public domain only in the U.S. or only outside the U.S. The API determines access rights in such cases based on the IP address of the requesting client. Clients should check the HTTP header for the `X-`

`HathiTrust-Notice` to be informed, and to inform third-party users, of obligations with regard to these items under their local copyright law.

The text of the notice in the HTTP header in each of these cases is available at the following links.

- Access and use policy statement for **“Public Domain only in the U.S.”**
- Access and use policy statement for **“Public Domain only outside the U.S.”**

Appendix C: Access and Authorization

The API provides access to repository data such as page images and to metadata resources derived from the METS file describing the object. Access is categorized along two dimensions: **`basic_access`** and **`extended_access`**.

There are two values along the **`basic_access`** dimension. They are a function of the rights attribute (copyright, license) and *access profile*. Access profiles are a replacement for sources and provide support for fine-grained access restrictions on certain items. See Database Layout section of the rights database document.

- **`free`**
  - Data (content) with rights in public-domain (may vary by geo-location), world, Creative Commons license, etc.
  - Metadata `{volume|article}/meta, volume/pagemeta` and `structure` resources.
- **`nonfree`**
  - Data (content) in-copyright or otherwise not freely available.

Within the **`free`** value of **`basic_access`**, some data may, nonetheless, have various restrictions. Example: Google-digitized volumes represented by the **`aggregate`** (zip package) resource are **`restricted`**, requiring higher than default authorization for access. However, individual page images represented by the **`volume/pageimage`** resource, may be **`open`**, not requiring special authorization.

Access to **`restricted`** resources, regardless of whether **`basic_access`** is **`free`** or **`nonfree`**, is modeled along a second dimension of values referred to as **`extended_access`** (refer also to Appendix B regarding the geo-location of the request).

`extended_access` values, orthogonal to **`basic_basic`**, support specific authorization for internal development, zip package download, un-watermarked Espresso Book Machine PDFs, archival images and un-watermarked image derivatives.

- **`allow_pdf_ebm`** -- 1 = allow access to un-watermarked EBM PDF. Default = 0.
- **`allow_raw_archival_data`** -- 1 = allow access to raw archival image (TIFF,

JP2000). Default = 0.

- **`allow_unwatermarked_derivatives`** -- 1 = allow derived image without a watermark. Default = 0.
- **`allow_zip`** -- 1 = allow download of Google zip. Default = 0.
- **`allow_nonfree --`** `a`llow access to `basic_access=nonfree.` Default = 0.