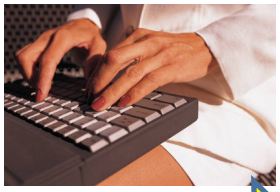
HTRC API Overview

Yiming Sun

HTRC Architecture



We are here

Direct programmatic access (by programs running on HTRC machines)

Portal access

Agent

- Application submission
- Collection building

Security (OAuth2)

Data API | Solr Proxy

Audit

Registry (WSO2)

- Token filter
- Sentence tokenizer
- Meandre flows
- Latent semantic analysis
- Word position
- Collections

Cassandra cluster volume store

Solr index shards

Compute resources

Storage resources

RESEARCH CENTER

HTRC RESTful Services

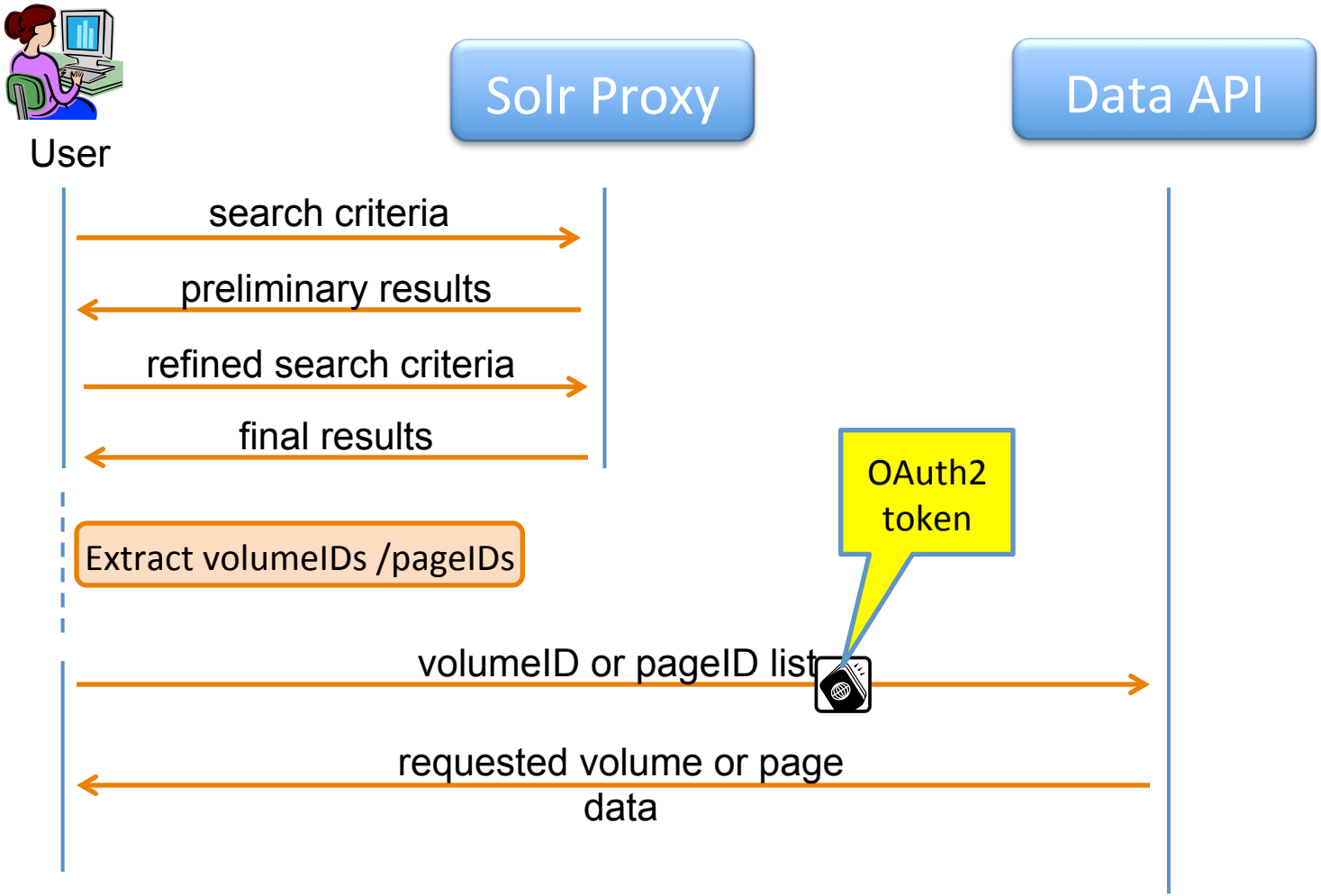
- Solr Proxy API
  - Search full text and metadata
  - Retrieve metadata
  - Pass through read-only Solr requests
  - Audit
- Data API
  - Retrieve volume and page data
  - Authorization via OAuth2
  - Communication over TLS (i.e. https)
  - Audit

RESEARCH CENTER

Why API? Why RESTful?

- API – Application Programming Interface
  - Interface stays the same while underlying implementation changes for improvement
  - Client to the API won't need to change
- RESTful – REpresentational State Transfer
  - A style of web service using HTTP
  - Light-weight with little to no dependencies
  - Language-agnostic

Typical Flow



RESEARCH CENTER

Search indexed fields in Solr Proxy API

Search for author named Dickens

URL:

http://chinkapin.pti.indiana.edu:9994/solr/select/?qt=sharding&q=author:dickens

chinkapin.pti.indiana.edu:9994 – solr proxy host and port number

solr – Solr proxy service

select - query

qt=sharding – query type must be sharding, as it is being used

q=author:dickens – query string

```
- <doc>
  - <arr name="publishDate">
      <str>1904</str>
    </arr>
    <str name="record_no">MIU01-008611879</str>
  - <arr name="countryOfPubStr">
      <str>United States</str>
    </arr>
  - <arr name="sdrnum">
      <str>sdr-nyp.b105718580</str>
    </arr>
  - <arr name="title">
      <str>Works</str>
      <str>The complete works.</str>
    </arr>
  - <arr name="publisher">
      <str> New York, Crowell [1904] </str>
    </arr>
    <str name="id">nyp.33433069279036</str>
  - <arr name="htsource">
      <str>New York Public Library</str>
    </arr>
  - <arr name="author">
      <str> Dickens, Charles, 1812-1870. </str>
    </arr>
    <str name="rights">1</str>
  </doc>
```

Specify returned fields in Solr Proxy API

Use “fl” to specify the fields to return.

e.g.

Return title, volumeID, and author of books whose author is Shakespeare

URL:

http://chinkapin.pti.indiana.edu:9994/solr/select/?

qt=sharding&q=author:shakespeare&fl=title,id,author

fl=title,id,author – return title, id, and author fields

RESEARCH CENTER

```
−<doc>
  −<arr name="title">
      <str>The First part of King Henry VI.</str>
      <str>Century Shakespeare</str>
    </arr>
    <str name="id">njp.32101074878446</str>
  −<arr name="author">
      <str> Shakespeare, William, 1564-1616. </str>
    </arr>
  </doc>
```

RESEARCH CENTER

Specify number of rows returned from Solr Proxy API

By default, Solr returns only 10 entries. Use “rows” to specify a different number:

e.g.

Return max of 20 entries for books whose title contains the word canterbury

URL:

http://chinkapin.pti.indiana.edu:9994/solr/select/?
qt=sharding&q=title:canterbury&rows=20

rows=20 – return a maximum of 20 entries

Some common indexed fields*

Field name	Indexed	Stored	Explanation
id	Y	Y	Field for volume ID
ocr	Y	N	Ocr field for full text search
title	Y	Y	Field for book title
author	Y	Y	Author for a volume
isbn	Y	Y	The International Standard Book Number (ISBN) for a volume
issn	Y	Y	The International Standard Serial Number (ISSN) for a volume
callnumber	Y	Y	The call number for a volume
language	Y	N	Field for the languages in which a volume is written
topic	Y	N	Topic of the volume
genre	Y	N	Genre the volume belongs to
publishDate	Y	Y	Publish date of the volume
publisher	Y	Y	publisher of the volume

*complete list at the end of the slide deck

Remember to URL encode query string

Web browsers and tools such as Blacklight perform URL encoding on your Solr query behind the scene so you may enter special characters such as colons and spaces directly:

URL:

http://chinkapin.pti.indiana.edu:9994/solr/select/?qt=sharding&q=title:canterbury AND author:chaucer

If you are writing code to query Solr using http connection, you must URL encode your query strings. Most languages include URL encoding libraries.

URL:

http://chinkapin.pti.indiana.edu:9994/solr/select/?qt=sharding&q=title%3Acanterbury%20AND%20author%3Achaucer

Retrieve MARC records for specified volumes

URL:

http://chinkapin.pti.indiana.edu:9994/solr/MARC/?volumeIDs=<volumeID-list>

MARC – retrieve MARC records

volumeIDs – parameter for volume ID list

<volumeID-list> - list of volume IDs. Individual volume IDs are joined together using "|" (bar, pipe)

volumeIDs=uc2.ark:/13960/t9668bb2t|uc2.ark:/13960/t8kd1sn38

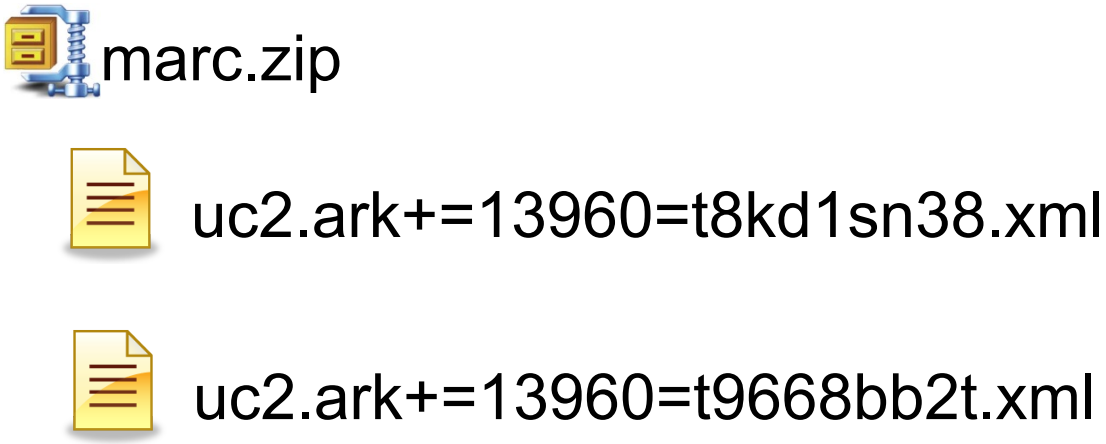
Again, query string needs to be URL encoded:

http://chinkapin.pti.indiana.edu:9994/solr/MARC/? volumeIDs=uc2.ark%3A%2F13960%2Ft9668bb2t%7Cuc2.ark%3A%2F13960%2Ft8kd1sn38

RESEARCH CENTER

What's returned

Streams back a ZIP file containing one MARC XML file for each volume



RESEARCH CENTER

Retrieve volumes from Data API

2 request headers must be present:

- Content-type: application/x-www-form-urlencoded
- Authorization: Bearer oauth2-token

Request method:

- POST

Retrieve volumes from Data API

URL:

https://silvermaple.pti.indiana.edu:25443/data-api/volumes

silvermaple.pti.indiana.edu:25443 – service host and port number

data-api – Data API service

volumes – request for volumes

Request body:

volumeIDs=<volumeID-list>

volumeIDs – parameter to list requested volumes

<volumeID-list> - volumeIDs joined by "|"

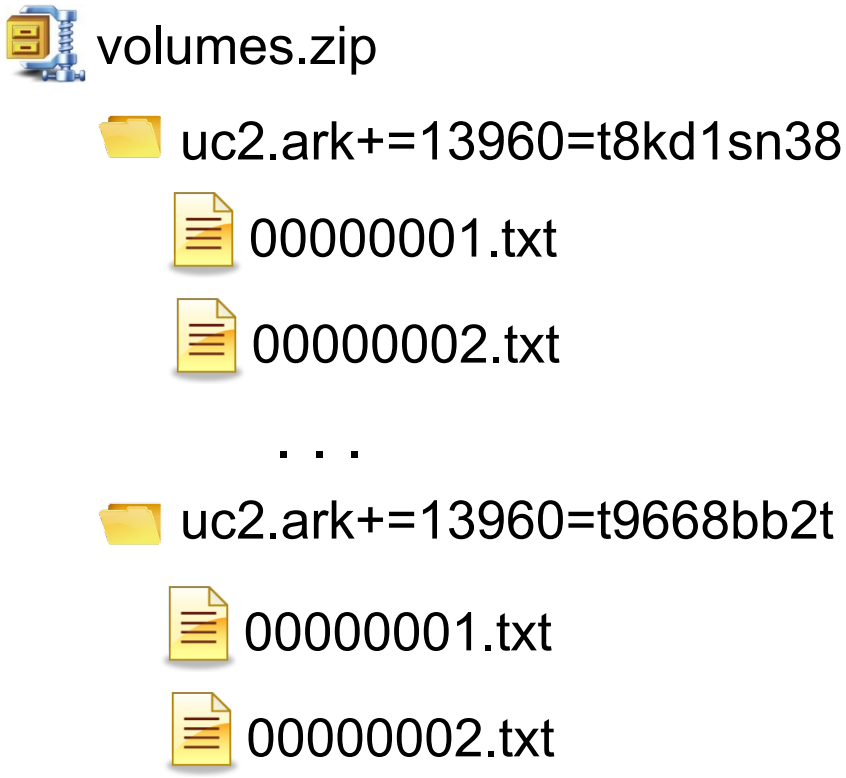
uc2.ark:/13960/t9668bb2t|uc2.ark:/13960/t8kd1sn38

volumeIDs=uc2.ark%3A%2F13960%2Ft9668bb2t%7Cuc2.ark%3A%2F13960%2Ft8kd1sn38

RESEARCH CENTER

What does Data API return?

Streams back a ZIP file where each volume is a directory, and each page in the volume is a text file named with a 8-digit page sequence number in the directory.



RESEARCH CENTER

Retrieve volumes with pages concatenated

URL:

https://silvermaple.pti.indiana.edu:25443/data-api/volumes

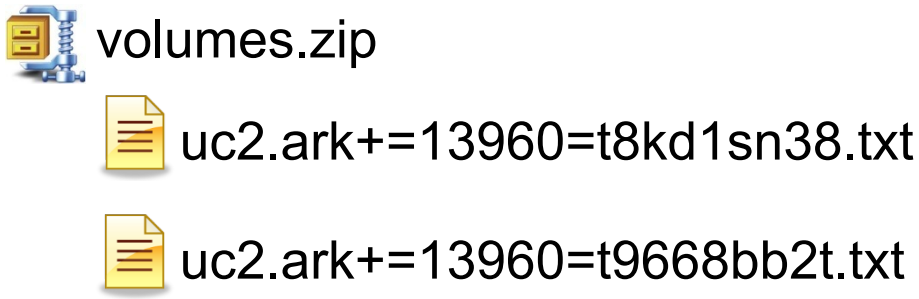
Request body:

volumeIDs=uc2.ark%3A%2F13960%2Ft9668bb2t%7Cuc2.ark%3A%2F13960%2Ft8kd1sn38&concat=true

concat=true – all pages of a volume are concatenated into a single text file

What does Data API return?

Streams back a ZIP file with each volume being a single text file and the content of the text file is all pages concatenated together.



RESEARCH CENTER

Retrieving pages

Enables retrieval of select pages instead of entire volumes

Page ID is volume ID followed by a comma separated list of page sequences enclosed in square brackets:

uc2.ark:/13960/t9668bb2t[2,4]

Page IDs are also joined by "|" to form a pageID list

RESEARCH CENTER

Retrieve pages from multiple volumes

Page ID list:

uc2.ark:/13960/t9668bb2t[2,4]|uc2.ark:/13960/t8kd1sn38[5,3,1]

URL:

https://silvermaple.pti.indiana.edu:25443/data-api/pages

Request body:

pageIDs= uc2.ark%3A%2F13960%2Ft9668bb2t%5B2%2C4%5D
%7Cuc2.ark%3A%2F13960%2Ft8kd1sn38%5B5%2C3%2C1%5D

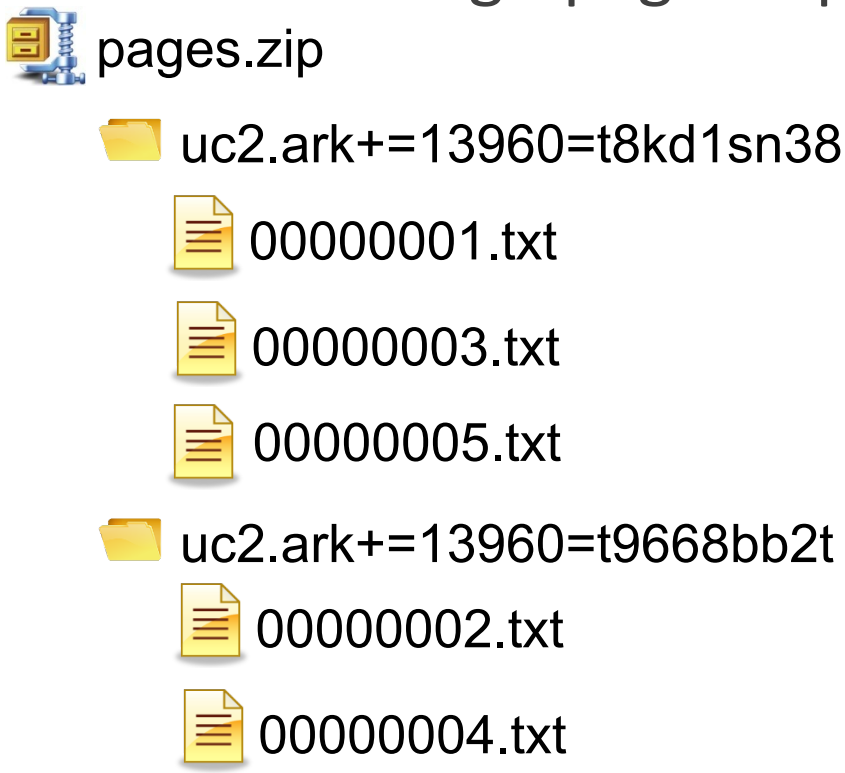
pages – retrieve pages

pageIDs – parameter for pageID list

RESEARCH CENTER

What does Data API return?

Streams back a ZIP file where each volume is a directory, and each requested page is a separate text file named with the 8-digit page sequence number.



RESEARCH CENTER

Retrieve pages as a bag of words

URL:

https://silvermaple.pti.indiana.edu:25443/data-api/pages

Request body:

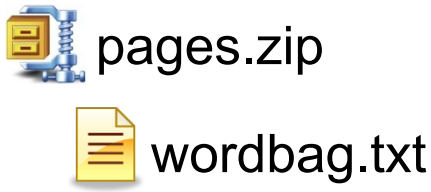
pageIDs= uc2.ark%3A%2F13960%2Ft9668bb2t%5B2%2C4%5D%7Cuc2.ark%3A%2F13960%2Ft8kd1sn38%5B5%2C3%2C1%5D&concat=true

concat=true – all requested pages are concatenated into a single file

RESEARCH CENTER

What does Data API return?

Streams back a ZIP file containing a single text file named wordbag.txt which is the concatenation of all requested pages



RESEARCH CENTER

What is ERROR.err?

If Data API encounters an error *before* it has started streaming back data, the client will receive an HTTP error response status code (e.g. 404, 500) and an explanation of the error in response body.

But if Data API encounters an error *after* it has started streaming back data as ZIP, it injects an ERROR.err file in the ZIP as the very last entry. This file contains the information of the very first error encountered.

If ERROR.err is present, one or more requested items are missing from the ZIP


pages.zip

. . .


ERROR.err

RESEARCH CENTER

When and why do I get an error?

For several possible reasons:

- Requested item does not exist (e.g. result of a deletion notice)
- Requested more items than the policy would allow
- Request is invalid (e.g. malformed volumeID list)
- Backend server overloaded or crashed

* list of possible errors at the end of the slide deck

OAuth2 Security Token

- To allow only authorized users to access HTRC data and resources
- To enforce auditing of activities
- Every request to Data API must bear a valid token
  - As "Authorization" HTTP request header to Data API

URL:

https://silvermaple.pti.indiana.edu:25443/oauth2/grant_type=client_credentials&client_id=<id>&client_secret=<secret>

oauth2 – OAuth2 token provider

grant_type=client_credentials – the only grant type we support at the moment

client_id – your assigned client ID, must be URL encoded

client_secret – your assigned client secret, must be URL encoded

RESEARCH CENTER

What does OAuth2 return and what do I do with the token?

Our OAuth2 token provider returns a JSON string looks something like the following:

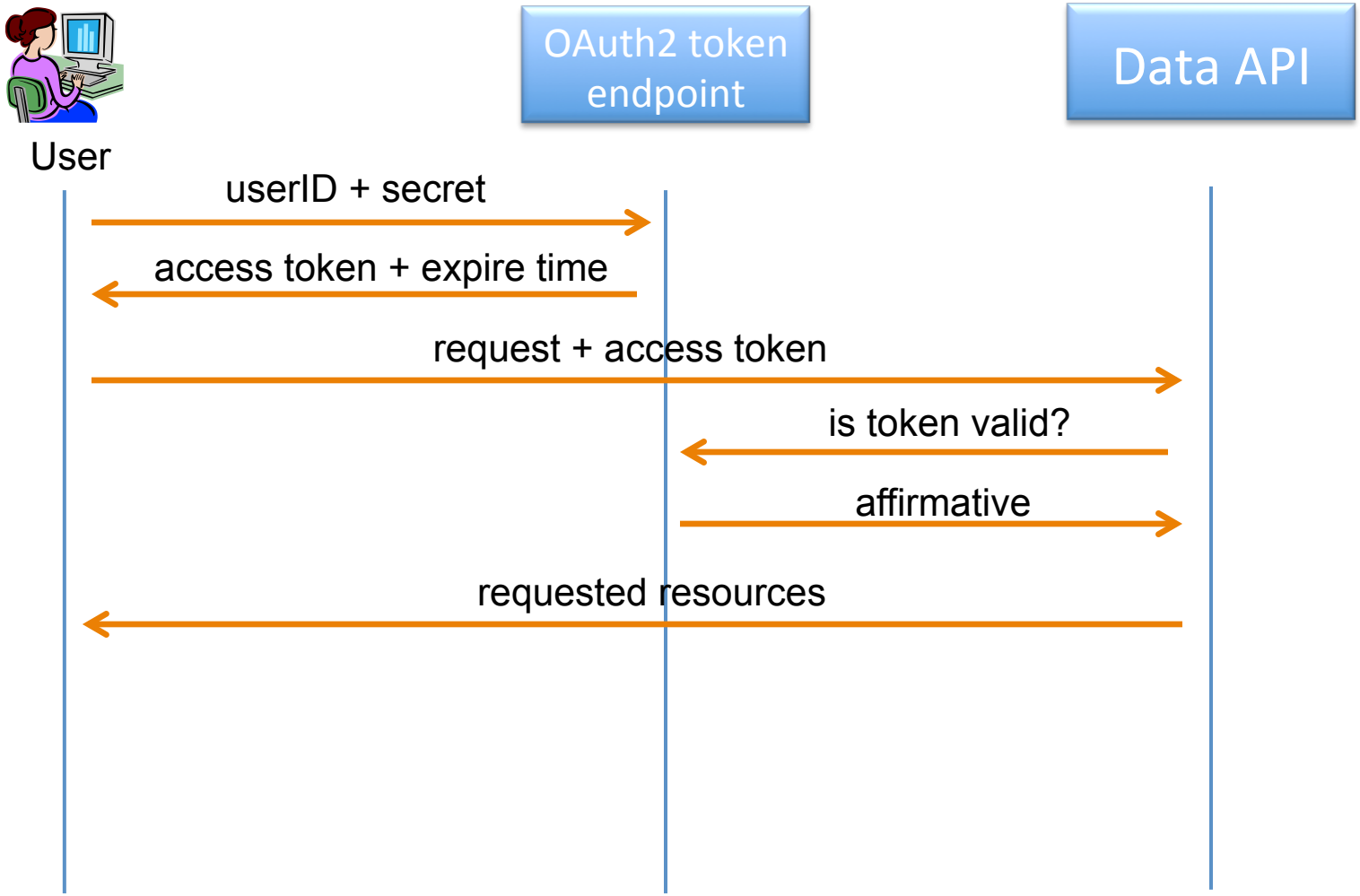
{“access_token” : “aaabbbccdd”, “expires_in” : “3600”}

The token must be set as “Authorization” header in the http request to Data API:

Authorization: Bearer aaabbbccdd

RESEARCH CENTER

Flow of obtaining and using OAuth2 tokens



RESEARCH CENTER

Summary

- Solr Proxy
  - Filters Solr queries to allow read-only requests
  - Always use qt=sharding
  - Augmented functionality: retrieve MARC records
  - All requests are HTTP GET requests
- Data API
  - Retrieve volumes and pages as ZIP
  - Requires valid OAuth2 token
  - All requests are HTTP POST requests
  - Request parameters go into request body
- OAuth2 token endpoint
  - Currently only support grant_type=client_credentials
  - Returns JSON
- URL Encoding your parameters!

References

- Solr 3.6 tutorial:
  - http://lucene.apache.org/solr/api-3_6_1/doc-files/tutorial.html
- Pairtree Specification
  - https://wiki.ucop.edu/display/Curation/PairTree
- OAuth2
  - http://oauth.net/2/

Questions?

Indexed and stored Solr fields

Field name	Indexed	Stored	Explanation
id	Y	Y	Field for volume ID
ocr	Y	N	Ocr field for full text search
title	Y	Y	Field for book title
rights	Y	Y	Copyright for a volume. e.g. "1" for public domain. please refer to http://www.hathitrust.org/rights_database#RightsAssignment
author	Y	Y	Author for a volume
authorStr	Y	Y	Author string not tokenized to facilitate facet on author
oclc	Y	Y	The OCLC Control Number
rptnum	Y	Y	
sdrnum	Y	Y	
isbn	Y	Y	The International Standard Book Number (ISBN) for a volume
issn	Y	Y	The International Standard Serial Number (ISSN) for a volume
callnumber	Y	Y	The call number for a volum
sudoc	Y	Y	Superintendent of Documents number for a volume. Please refer to http://www.fdlp.gov/cataloging/856-sudoc-classification-scheme?start=3
language	Y	N	Field for the languages in which a volume is written
htsource	Y	Y	Field for the sources of a volume, e.g. "Indiana University"
era	Y	N	The era of the volume
geographic	Y	N	Brief geographic information for a volume, e.g."pennsylvania"
country_of_pub	Y	N	The publication country of the book
countryOfPubStr	Y	Y	Country names not tokenized for better facet string on it
topic	Y	N	Topic of the volume
topicStr	Y	Y	Topic string not tokenized for better facet string on it
genre	Y	N	Genre the volume belongs to
publishDate	Y	Y	Publish date of the volume
publisher	Y	Y	publisher of the volume

Data API error responses

Response Status	Response Body	Reason
400 Bad Request	Missing required parameter volumeIDs	request for volumes does not have volumeIDs query parameter
400 Bad Request	Missing required parameter pageIDs	request for pages does not have pageIDs query parameter
400 Bad Request	Malformed volume ID list. Offending token: xxx	volumeID list contain tokens that are not valid volumeIDs and cannot be parsed
400 Bad Request	Malformed page ID list. Offending token: xxx	pageID list contains tokens that are not valid pageIDs and cannot be parsed
400 Bad Request	Request too greedy. Request violates Max Volumes Allowed xxx. Offending ID: zzz	request would touch and retrieve more volumes than allowed by the policy. In the response body, xxx is the limit set by the policy, and zzz is the first volumeID that exceeds the limit. Applicable if the policy is set on the server.
400 Bad Request	Request too greedy. Request violates Max Total Pages Allowed xxx. Offending ID: zzz	request would touch and retrieve more pages than allowed by the policy. In the response body, xxx is the limit set by the policy, and zzz is the first pageID that exceeds the limit. Applicable if the policy is set on the server.
400 Bad Request	Request too greedy. Request violates Max Pages Per Volume Allowed xxx. Offending ID: zzz	request would touch and retrieve more pages per volume than allowed by the policy. In the response body, xxx is the limit set by the policy, and zzz is the first ID that exceeds the limit. Applicable if the policy is set on the server
404 Not Found	Key not found. Offending key: xxx	request asks for a non-existent volumeID, or the request asks for a non-existent page sequence of a valid volume (e.g. asking for page 100 of a volume with only 90 pages).
500 Internal Server Error	Server too busy.	Data API gets a timed out exception while trying to retrieve data.
500 Internal Server Error	Internal server error.	other exceptions occurred when Data API tries to retrieve data.