HTRC API Overview
HTRC Architecture

Portal access

Agent
- Application submission
- Collection building

Direct programmatic access (by programs running on HTRC machines)

Security (OAuth2)

Registry (WSO2)
- Token filter
- Sentence tokenizer
- Word position
- Meandre flows
- Collections

Data API

Solr Proxy

Audit

Compute resources

Storage resources

Cassandra cluster
- Volume store

Solr index shards
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
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

User → Solr Proxy → Data API

- search criteria
- preliminary results
- refined search criteria
- final results

Extract volumeIDs / pageIDs

volumeID or pageID list

requested volume or page data

OAuth2 token
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
- `<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>`
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*

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

*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/?volumelDs=<volumelD-list>

MARC – retrieve MARC records
volumelDs – parameter for volume ID list
<volumelD-list> - list of volume IDs. Individual volume IDs are joined together using “|” (bar, pipe)

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

Again, query string needs to be URL encoded:

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

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

 marc.zip

 uc2.ark+=13960=t8kd1sn38.xml

 uc2.ark+=13960=t9668bb2t.xml
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
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.

volumes.zip

- `uc2.ark+=13960=t8kd1sn38`
- `00000001.txt`
- `00000002.txt`

...
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.

volumes.zip
uc2.ark+=13960=t8kd1sn38.txt
uc2.ark+=13960=t9668bb2t.txt
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
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 %7C uc2.ark%3A%2F13960%2Ft8kd1sn38%5B5%2C3%2C1%5D
cu pages – retrieve pages

cu pageIDs – parameter for pageID list
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.

```
pages.zip
```

- `uc2.ark+=13960=t8kd1sn38`
  - `00000001.txt`
  - `00000003.txt`
  - `00000005.txt`

- `uc2.ark+=13960=t9668bb2t`
  - `00000002.txt`
  - `00000004.txt`
Retrieve pages as a bag of words

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

Request body:
pagelDs=uc2.ark%3A%2F13960%2Ft9668bb2t%5B5B2%2C4%5D%7Cuc2.ark%3A%2F13960%2Ft8kd1sn38%5B5B5%2C3%2C1%5D&concat=true

concat=true – all requested pages are concatenated into a single file
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.
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](attachment:pages.zip)

...
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
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" : "aaabbbccddd", "expires_in" : "3600"}
```

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

Authorization: Bearer aaabbbccddd
Flow of obtaining and using OAuth2 tokens

User

OAuth2 token endpoint

Data API

userID + secret

access token + expire time

request + access token

is token valid?

affirmative

requested resources
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:

• Pairtree Specification
  – https://wiki.ucop.edu/display/Curation/PairTree

• OAuth2
  – http://oauth.net/2/
Questions?
## Indexed and stored Solr fields

<table>
<thead>
<tr>
<th>Field name</th>
<th>Indexed</th>
<th>Stored</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Y</td>
<td>Y</td>
<td>Field for volume ID</td>
</tr>
<tr>
<td>ocr</td>
<td>Y</td>
<td>N</td>
<td>Ocr field for full text search</td>
</tr>
<tr>
<td>title</td>
<td>Y</td>
<td>Y</td>
<td>Field for book title</td>
</tr>
<tr>
<td>rights</td>
<td>Y</td>
<td>Y</td>
<td>Copyright for a volume. e.g. &quot;1&quot; for public domain. please refer to <a href="http://www.hathitrust.org/rights_database#RightsAssignment">http://www.hathitrust.org/rights_database#RightsAssignment</a></td>
</tr>
<tr>
<td>author</td>
<td>Y</td>
<td>Y</td>
<td>Author for a volume</td>
</tr>
<tr>
<td>authorStr</td>
<td>Y</td>
<td>Y</td>
<td>Author string not tokenized to facilitate facet on author</td>
</tr>
<tr>
<td>oclc</td>
<td>Y</td>
<td>Y</td>
<td>The OCLC Control Number</td>
</tr>
<tr>
<td>rptnum</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>sdrnum</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>isbn</td>
<td>Y</td>
<td>Y</td>
<td>The International Standard Book Number (ISBN) for a volume</td>
</tr>
<tr>
<td>issn</td>
<td>Y</td>
<td>Y</td>
<td>The International Standard Serial Number (ISSN) for a volume</td>
</tr>
<tr>
<td>callnumber</td>
<td>Y</td>
<td>Y</td>
<td>The call number for a volume</td>
</tr>
<tr>
<td>language</td>
<td>Y</td>
<td>N</td>
<td>Field for the languages in which a volume is written</td>
</tr>
<tr>
<td>htsource</td>
<td>Y</td>
<td>Y</td>
<td>Field for the sources of a volume, e.g. &quot;Indiana University&quot;</td>
</tr>
<tr>
<td>era</td>
<td>Y</td>
<td>N</td>
<td>The era of the volume</td>
</tr>
<tr>
<td>geographic</td>
<td>Y</td>
<td>N</td>
<td>Brief geographic information for a volume, e.g.&quot;pennsylvania&quot;</td>
</tr>
<tr>
<td>country_of_pub</td>
<td>Y</td>
<td>N</td>
<td>The publication country of the book</td>
</tr>
<tr>
<td>countryOfPubStr</td>
<td>Y</td>
<td>Y</td>
<td>Country names not tokenized for better facet string on it</td>
</tr>
<tr>
<td>topic</td>
<td>Y</td>
<td>N</td>
<td>Topic of the volume</td>
</tr>
<tr>
<td>topicStr</td>
<td>Y</td>
<td>Y</td>
<td>Topic string not tokenized for better facet string on it</td>
</tr>
<tr>
<td>genre</td>
<td>Y</td>
<td>N</td>
<td>Genre the volume belongs to</td>
</tr>
<tr>
<td>publishDate</td>
<td>Y</td>
<td>Y</td>
<td>Publish date of the volume</td>
</tr>
<tr>
<td>publisher</td>
<td>Y</td>
<td>Y</td>
<td>Publisher of the volume</td>
</tr>
<tr>
<td>Response Status</td>
<td>Response Body</td>
<td>Reason</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>Missing required parameter volumeIDs</td>
<td>request for volumes does not have volumeIDs query parameter</td>
<td></td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>Missing required parameter pageIDs</td>
<td>request for pages does not have pageIDs query parameter</td>
<td></td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>Malformed volume ID list. Offending token: xxx</td>
<td>volumeID list contain tokens that are not valid volumeIDs and cannot be parsed</td>
<td></td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>Malformed page ID list. Offending token: xxx</td>
<td>pageID list contains tokens that are not valid pageIDs and cannot be parsed</td>
<td></td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>Request too greedy. Request violates Max Volumes Allowed xxx. Offending ID: zzz</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>Request too greedy. Request violates Max Total Pages Allowed xxx. Offending ID: zzz</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>Request too greedy. Request violates Max Pages Per Volume Allowed xxx. Offending ID: zzz</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>404 Not Found</td>
<td>Key not found. Offending key: xxx</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>500 Internal Server Error</td>
<td>Server too busy.</td>
<td>Data API gets a timed out exception while trying to retrieve data.</td>
<td></td>
</tr>
<tr>
<td>500 Internal Server Error</td>
<td>Internal server error.</td>
<td>other exceptions occurred when Data API tries to retrieve data.</td>
<td></td>
</tr>
</tbody>
</table>