Proposal for Persistent & Unique Entity Identifiers

Jacob Jett, Guangchen Ruan, Leena Unnikrishnan, Colleen Fallaw, Chistopher Maden, Tim Cole
UNIVERSITY OF ILLINOIS GSLIS | INDIANA UNIVERISTY

Executive Summary

This proposal argues for the establishment of persistent and unique identifiers for page level content. The page is a key conceptual entity within the HathiTrust Research Center (HTRC) framework. Volumes are composed of pages and pages are the size of the portions of data that the HTRC's analytics modules consume and execute algorithms across. The need for infrastructure that supports persistent and unique identity for is best described by seven use cases:

1. **Persistent Citability:** Scholars engaging in the analysis of HTRC resources have a clear need to cite those resources in a persistent manner independent of those resources' relative positions within other entities.
2. **Point-in-time Citability:** Scholars engaging in the analysis of HTRC resources have a clear need to cite resources in an unambiguous way that is persistent with respect to time.
3. **Reproducibility:** Scholars need methods by which the resources that they cite can be shared so that their work conforms to the norms of peer-review and reproducibility of results.
4. **Supporting "non-consumptive" Usage:** Anonymizing page-level content by disassociating it from the volumes that it is conceptually a part of increases the difficulty of leveraging HTRC analytics modules for the direct reproduction of HathiTrust (HT) content.
5. **Improved Granularity:** Since many features that scholars are interested in exist at the conceptual level of a page rather than at the level of a volume, unique page-level entities expand the types of methods by which worksets can be gathered and by which analytics modules can be constructed.
6. **Expanded Workset Membership:** In the near future we would like to empower scholars with options for creating worksets from arbitrary resources at arbitrary levels of granularity, including constructing worksets from collections of arbitrary pages.
7. **Supporting Graph Representations:** Unique identifiers for page-level content facilitate the creation of more conceptually accurate and functional graph representations of the HT corpus.

There several ways in which persistent and unique identifiers for page-level content can be implemented. The solutions range from Handle or ARK Identifier servers providing unique resource resolution services on the Web, to the internal use of URN schemes for unique and persistent internal reference. Since page-level content within the HTRC context is not meant to be "consumable" there is no need for it to be uniquely identified in a manner that is resolvable with respect to ordinary Web browsers. Therefore, the adoption of a URN scheme makes the most sense for the HTRC's needs. Of the possible URN schemes, the UUID scheme provides both a well-documented solution, there are algorithms for generating UUIDs in the libraries of most software languages, and a highly scalable solution, the UUID scheme provides 3.4×10^{38} possible unique identifiers.

Page-level Identifiers Recommendation

It is our recommendation that infrastructure supporting the creation and management of UUIDs for page-level content be adopted. In addition, as a simple means of providing support for citability

down to the level of the exact representations used in analyses, we propose extending the scheme to include the individual file objects, the OCR text and jpeg / tiff image files, so that each file object has a globally unique and persistent reference within the HTRC context.

Our estimate of the activities needed for such an implementation, including the human effort, are:

Design and implement database

Task	Time
Design schemas for RDBMS and graph database, generate and ingest dummy data.	1 week
Performance evaluation between RDBMS and graph database.	1.5 weeks
Decision making on which database to use, schema refine, performance tuning.	1 week

Design and implement identifier minting system

Task	Time
Modify ***Volume***/***Page*** ingestion; implement detection algorithm that detects changes in METs; implement JDBC or graph DB operations that insert corresponding records into database. Perform debug and functionality test.	2.5 weeks
Implement code that mints identifiers for existing ***Volumes*** in Cassandra; run the code and insert corresponding records into RDBMS or graph DB.	2 weeks

Extend existing DataAPI

Task	Time
Design and implement new RESTful calls to fulfill user queries that contain point-in-time info in resource request; and calls that request identifiers, e.g. query minted id for a ***Page*** based on a sequence number.	2.5 weeks
Design other RESTful calls that retrieve, query and insert records for other HTRC data assets, e.g. ***Workset*** and software packages.	2 weeks

Adoption of Identifiers for additional entities within the HTRC context

In addition to adopting persistent and unique identifiers for page-level content, our group was asked to consider the adoption of identifier schemes for additional entities with the HTRC context. Since volumes are already adequately identified by the HT's Handle service and the Internet Archive's ARK Identifier server, no further action is required for volumes.

Worksets and analytics modules were the other two entities considered. Since both of these entities are either already encapsulated within a Web facing service (i.e., the workset builder) or are under consideration for migration to a Web facing service (i.e. Git) it does not make sense to simply fold them into the UUID services recommended above. Rather, a gentler identification scheme of workset / analytics module name + author / user name can be exploited.

Our estimate of the activities needed for such an implementation, including the human effort, are:

Implement version control for Worksets[1]

Task	Time
Modify settings in the WSO2 Governance Registry to turn on versioning for ***Worksets***, including testing and evaluating storage implications.	2 days

Migrate Analytics Modules into GitHub

Task	Time
Check all Analytic Modules and other HTRC Software Projects into GitHub	2 days

Layered Services

As noted above, extending the UUIDs to encompass the file objects that store and provide the representations of page-level content can be used as a means to achieve version control of resources within the HTRC context. There are alternate means of achieving this through layered services. We considered a number of layered services that could be exploited for content negotiation, version control and for support of an HTRC-based page turner client. Adoption of any of the layered services suggested in this summary and the proposal document should not be considered a recommendation but rather exists to articulate a number of stretch goals that can be realized through the implementation of persistent and unique identifiers.

Our estimate of the activities needed for such an implementation, including the human effort, are:

Extending the Data API to include negotiation services

Task	Time
Extensions to the Data API including negotiated services.	3 weeks

Implementation of name resolver services for Volumes, Pages, and File Objects

Task	Time
Determine the type of database to use: RDBMS or graph database, design and implementation of the same.	3 weeks

Implementation of name resolver services for Worksets and Analytics Modules

Task	Time
Tool to add identifiers for existing ***Worksets***.	1.5 weeks
Software tool to update database with identifiers for analytic modules and other software projects committed to github.	2 weeks

Other Services

We also considered the implementation of a Handle or ARK Identifier service and server in place of many of the name resolver services listed above. However, any dedicated server would only

[1] Note that versioning has already been provided for in the Workset xsd document.

have limited utility within the HTRC context, essentially being a name resolution service only for those Worksets and Analytics Modules which are made openly accessible to users outside of the HTRC. Combined with the equipment expenses and additional time in labor (listed below) we do not recommend implementing a Handle or ARK Identifier server until such a time as HTRC resources are to be made more widely available to the Web-going public.

Design & Implementation of a Handle Server for additional Web-based name resolution services

Task	Time
Application and receipt for a unique institutional identifier	1 week
Set up and configuration of Handle server, including implementation of additional customized API services	4 weeks
Updates to existing layered services to operate with Handle server, implementation of local Handle service including reading and interpreting data from the table, interpreting JSON/XML/other access information, directing to layered service for date-based identifiers.	2 weeks
Modify the ingest process to update the database appropriately for ***Pages***, files, and ***Volumes***: add new items, handle deletes by updating the "end date column", create access information in expected JSON/XML/other format etc.	2 days
Add above procedures to ***Workset*** builder for creation and deletion of ***Worksets***.	1.5 weeks

Finally, we considered future growth scenarios. If the proposals for both page-level and workset / analytics modules are adopted then the HTRC will be well-positioned to naturally grow into experimentation with graph or RDF-based infrastructures. Any graph-based representation will be able to fully exploit the HTRC's identifiers. Should additional identifiers specific to particular graph nodes become necessary then the adoption of additional identifier schemas, or the reuse of the UUID scheme under a new context, can be explored more fully. Ideally, the good use of existing RDF-based resources combined with best practice usages of blank nodes, data / content literals, and appropriate predicates should ensure that graph-based representations and infrastructure provide scalable and agile services.

Conclusion

The primary risk that HTRC faces by not adopting some form of globally persistent identifier scheme for page-level content is the creation of a gulf between the functionality desired by scholarly users and the functionality that the existing infrastructure can support. Such a gulf will ultimately stifle the kinds of research questions that scholars can develop to exploit the HTRC's services to resolve. Globally persistent identifiers at every level of conceptual entity directly ameliorate these risks. By adopting a persistent and unique means for identifying page-level content the HTRC can both better meet the needs of its scholarly users and more fully prepare for future endeavors.

Context

The HathiTrust Research Center (HTRC) was the result of a call for proposals by the HathiTrust (HT) Executive Committee to establish a research center that would facilitate leveraging the millions of in-copyright works owned by HathiTrust member institutions in "non-consumptive" computational research. To this end research teams at Indiana University (IU) and the University of Illinois at Urbana-Champaign (UIUC) combined their efforts in a joint development venture to realize the HTRC concept. As the HTRC continues to grow and evolve so to have its computational and system needs. Specifically in the context of this proposal, the need for a persistent and unique method of identifying ***Page***-level content is discussed at length and a proposal for its implementation put forth. Adoption of persistent and unique identifiers for other entities within the HTRC's computational analysis milieu are also discussed and included within the confines of the overall implementation proposal. The design and implementation of layered services, including the implementation of a Handle server for sharing HTRC entities on the Web, are also discussed and are included in the proposal in the form of stretch goals.

Definitions

Within the HTRC context there are several key entities that exist to support various HTRC functions and initiatives. The important entities for this proposal are: ***Worksets***, ***Volumes***, ***Pages***, ***File Objects***, and ***Analytics Modules***.

Workset

The core functionality of the HTRC's system is that researchers can gather together an arbitrary grouping of research materials and run selected ***Analytics Modules*** on their content, generating various publishable research results. In all ways, an HTRC ***Workset*** is analogous to a scholar's research collection. At this time, ***Workset*** membership is limited to ***Volumes***. Therefore, within the context of HTRC we define a ***Workset*** as:

> ***Workset***: a collection of ***Volumes***

Unfortunately, ***Volumes*** do not correspond to actual analytic granules and so this definition does not actually meet researcher needs with regards to the specificity of the data actually being analyzed. In the future a broader definition for ***Workset*** that can include resources at the ***Page*** and ***File Object*** levels (or even graphs) will facilitate growth in both the kinds of ***Analytics Modules*** that can be used on a ***Workset*** and the claims that can be made based on the results of analyses.

Volume

One of the most basic entities within the HTRC's current data model is that of ***Volume***. From a conceptual point of view a ***Volume*** is a generic digital representation of a corresponding physical book. However, as an artifact of the digitization process, physical books are not digitized whole cloth but are instead digitized on a ***Page*** by ***Page*** basis. For practical purposes then, within the context of the HTRC we define a ***Volume*** as:

Volume*:** a collection of ***Pages

Within the HTRC context, ***Volumes*** are identified by identifiers that are assigned to them by their owning institution. By HT convention all ***Volume*** identifiers are either Handles assigned by the HT or are ARK identifiers assigned by the Internet Archive. ***Volumes*** are described by METS files which denote both what ***File Objects*** are a part of a ***Volume*** and in what order (or "sequence") ***Page*** level content (in the form of ***File Objects)*** should be arranged in for client software such as the HT ***Page***-turner.

Page

A ***Page*** is the minimum content granule within the HT and HTRC systems. It corresponds to the content of one side of one leaf of a physical book. In the context of the HTRC we define a ***Page*** as:

> ***Page*:** digitized content that directly corresponds to real-world content appearing on a physical ***Page*** (i.e. on one side of a physical leaf) in a physical book

From the end user's point of view a ***Page*** is a specific resource they wish to interact with. Because there are multiple methods to digitize real-world content and thereby multiple digital representations of digitized content, a ***Page*** is a conceptual container used by the system to group various representations of digital content together (e.g., to group various versions of ***File Objects*** or ***File Object*** mimeTypes together). Within the current context of the HTRC, content at this level is identified by a ***Volume*** id in combination with the sequence of the ***Page*** within that ***Volume***. ***Pages*** do not inherently have an order (or sequence value) but rather, this is a relationship asserted by the ***Volume*** (through the metadata contained within the METS file that describes it) within which they are a constituent of.

File Object

The ***File Object*** is the actual thing consumed by a client software package (e.g., the HT ***Page***-turner) or by an ***Analytics Module*** such as the HTRC's feature extraction algorithms. In the context of the HTRC we define a ***File Object*** as:

> ***File Object*:** a file containing a format specific representation of some ***Page*** level content (e.g. an OCR text representation of a ***Page*** or the corresponding *tiff* or *jpeg* image representation of that same ***Page***)

From the system point of view, a ***File Object*** is the thing which must be parsed or rendered by some ***Analytics Module*** or software client. In the current HTRC context, ***File Objects*** are identified by a combination of ***Volume*** id and file name. There is no method for detecting changes in ***File Objects*** between one ingest and the next. The METS files that describe ***Volumes*** also contain some minimum metadata describing ***File Objects*** in the form of checksums but no means of leveraging that data for version control has been implemented yet.

Analytics Module

An ***Analytics Module*** describes the various groups of algorithms that process various parts of the HT corpus. These include feature extraction algorithms, SEASR algorithms, and various R and Python clients developed by the HTRC and scholarly researchers. As a part of the HTRC's research workflows, an ***Analytics Module*** consumes ***File Objects*** and produces some analytical results that can be further leveraged by HTRC staff or researchers. Conceptually an ***Analytics Module*** consumes ***Page*** level content through the intermediaries of ***Worksets*** (which are comprised of ***Volumes***) and ***Volumes*** (which in turn are comprised of ***Pages***). In the context of the HTRC then, we define an ***Analytics Module*** as:

> ***Analytics Module*:** any arbitrary grouping of software that is designed to specifically consume a ***Workset*** or similar sub-collection of the HT corpus and return analytics results (e.g. extracted features, statistically significant pattern analysis results, etc.)

Currently an ***Analytics Module*** registers with HTRC systems via configuration files specifying its deployment details. An ***Analytics Module*** has a name specified in the configuration file, and this name can be used to identify it in the context of the HTRC system. There is currently no uniform method for storing, modifying, or referring to the particular instance of an ***Analytics Module*** used in an analysis.

Proposal for Persistent & Unique *Page* Identifiers

The primary goal of this document is to propose the adoption of a persistent and unique method for referring to ***Page*** level content. Two existing methods for identifying ***Page*** level content already exist:

1. ***Page*** level content is identified in the context of the HT Page-turner client by the combination of a ***Volume*** identifier and the *order* value assigned to the ***File Objects*** that contain various representations of that ***Page*'*s*** content.
2. ***Page*** level content is identified in the context of the HTRC ingestion workflow by the combination of a ***Volume*** identifier and the *order* value assigned to the ***File Objects*** that contain various representations of that ***Page*'*s*** content. Currently in HTRC, the ***Volume*** identifier is used as the file name of a *zip* ***File Object*** that contains the ***File Objects*** containing the OCR text representations of the ***Page*'*s*** content. HTRC anticipates that other types of representations in addition to OCR text, such as images or TEI text, will also be included in the *zip* ***File Object*** that represents the ***Volume***. These other representations will need to be identified distinctly.

The primary problem with both of the ***Volume*** identifier plus *order* schemes is that they do not provide a persistent means of identifying ***Page*** level content. They are not persistent because HT's member institutions periodically rerun OCR processes providing new text ***File Objects*** in the process. Sometimes ***Volumes*** are completely rescanned and additional ***Pages*** are added necessitating changes in the relative *order* values associated with each ***Page***. Finally, it is occasionally the case that entire ***Volumes*** are withdrawn from the HT corpus due to copyright concerns.

The two schemes described above are also inadequate with regards to uniquely identifying ***Page*** level content. This is because the unique portion of any arbitrary ***Page*** identifier is the identifier for the ***Volume*** it is a part of. This leaves the infrastructure vulnerable to various collisions between ***Volumes*** during the rsync/ingestion, query response, and analytics workflows. If for some reason the ***Pages*** from two or more ***Volumes*** were to be merged together it would be impossible to disambiguate them.

There are clear benefits that can be realized by adopting a persistent and unique identifier scheme as well as risks for the project should a scheme not be adopted. These are detailed below as *Objective-specific Use Cases* and *Future-Functionality Use Cases*.

Objective-specific Use Cases

Objective-specific use cases all revolve around the HTRC's primary goal of supporting research that exploits the HT corpus via "non-consumptive" means. In order for the research results to be considered valid, the scholars analyzing the corpus need reliable methods for citing their data so that the results are reproducible. A recent user study by members of the Illinois team demonstrated that citability and reproducibility are core requirements for scholarly uptake, with one interviewee noting:

> "'[I]f you just say, I have a corpus and nobody is allowed to see it but wonderful things come out of it… That's not really research. That is a problem here, I think, for us, because we are just starting with this kind of work. We are trying to get accountability for the kind of work we are doing. And it's important for us to show the basis our work' (P13)." — excerpted from Fenlon et al, 2014.

Persistent Citability: A scholar should be able to refer to content within a ***Volume*** in a way that will survive changes to the sequence or content details. For example, "the poem on page 53" might become "the poem on page 55" if new ***Pages*** are added to the ***Volume***; there should be a way to cite the ***Page*** with the poem that survives these changes that is independent of its relative position inside of the ***Volume***.

- **Benefits:** Adoption of a persistent and unique identifier scheme would allow ***Page*** level content to be decoupled from its position within a ***Volume*** allowing citations of specific portions of content to be portable across multiple versions of a ***Volume***. Failure to adopt a reliable means of citing content that is critical to a scholar's research will create a barricade towards scholarly uptake and usage of the HTRC as a means of analyzing the HT corpus.

Point-in-time Citability: A scholar should be able to refer to the *specific* text or image that was the subject of analysis. If older versions of files are discarded, such references may become unresolvable, but the reference should still be unambiguous. This can be as simple as the system noting in a log somewhere that a specific resource has been withdrawn from the corpus.

- **Benefits:** Adoption of a persistent and unique identifier scheme here directly supports the *reproducibility* use case below by providing the HTRC system with a means of identifying content that has updated, replaced, or withdrawn. Failure to adopt a reliable means of referencing specific analytic granules at the point-in-time at which they were analyzed

directly impacts the reproducibility of the outcomes of research processes. No claims of reproducibility will be able to be asserted as the data being analyzed is liable to be volatile.

Reproducibility: A scholar should be able to share references to the content he or she analyzed so that others can reproduce their work. As long as the content in question is available, it should be retrieved unambiguously; if the content is discarded, that should be clear that it has been discarded, rather than having slightly different content silently substituted.

- **Benefits:** Adoption of a persistent and unique identifier scheme supports the ability of scholars to reuse the specific components of other scholars' analyses at a variety of granularities, including the ***Workset***, ***Volume***, and ***Page*** levels in order to verify and validate research outcomes through the peer review process. Failure to adopt a reliable means for scholars to share their data and analytic workflows will serve as a barricade towards peer review of research results ultimately impeding the uptake and usage of the HTRC as a means of analyzing the HT corpus.

Supporting "Non-consumption" Usage: It has been pointed out that there is a very small but present risk that a clever computer scientist may be able to combine the data provided through analyses like the feature extraction ***Analytics Module*** and a sophisticatedly configured n-gram viewer to reconstruct the content of a ***Volume***. While there is no method that can act as a panacea and completely ameliorate this potential risk, adoption of a methodology that moves away from leveraging information contained in the *order* value to identify both ***Page*** level content within a ***Volume*** and ***Volume***-specific chunks of the feature extraction output improves the situation.

- **Benefits:** Adoption of a persistent and unique identifier scheme further reduces the risk that analytics results might be leveraged to reconstruct entire ***Volumes*** of content from the HT corpus by making ***Pages*** more anonymous and less directly linked to the ***Volume*** entity that contextualizes them. While failure to adopt a reliable means of decoupling ***Page*** level content from its relative position within a ***Volume*** only maintains the status quo with regards to the risk that a researcher might reconstruct an entire ***Volume's*** content, it does continue to overload conceptual notions of ***Page*** with that of ***Volume***.

Future-functionality Use Cases

We anticipate that the adoption of a persistent and unique identifier scheme will greatly facilitate the development of workflows and infrastructure that support new or expanded capabilities within the HTRC context. In particular, granularity of ***Workset*** members is an important consideration of scholars seeking to use the HTRC's analytical services (Fenlon et al, 2014).

Improved Granularity: We would ideally like to use the identifiers to point at the minimum size content blocks within the HTRC system. Since we know that the ***File Objects*** that provide text or image representations of a ***Volume*** are chunked at the ***Page*** level, it would be advantageous to be able to point directly at ***Pages*** and ***File Objects***, rather than going through the indirection of ***Volume*** membership.

- **Benefits:** Adoption of a persistent and unique identifier scheme can increase the efficiency and functionality of HTRC ***Analytics Modules*** by facilitating both the construction of

queries that can directly interrogate ***Page*** level content and the connection of researchers' annotations to specific ***Page*** level content. Failure to adopt means of reliably identifying smaller granules of content than the ***Volume*** level will limit in the kinds of queries that researchers can employ during the process of ***Workset*** creation, artificially constraining the kinds of research questions that they can ask with regards to the HT corpus.

Expanded *Workset* Membership: Expanded entity granularity, especially at the ***Page*** level, directly supports a number of additional ***Workset*** membership use cases and empowers researchers to build ***Worksets*** that consist of ***Pages*** containing specific types of content, such as poems.

- **Benefits:** Adoption of a persistent and unique identifier scheme directly supports scholarly requirements for ***Worksets*** consisting of ***Pages*** containing specific types of content, expanding the kinds of research questions that the HTRC's analytics services can support. Failure to adopt a means of identifying more granular content will lock ***Workset*** and collection entities into a membership definition that excludes any kind of resource other than ***Volumes***, ultimately stifling the kinds of analyses that researchers can carry out.

Supporting Graph Representations of HT Corpus Entities: Researchers at Illinois are currently exploring graph representations of ***Worksets*** and other sub-divisions of the HT corpus through the Workset Creation for Scholarly Analysis (WCSA) initiative. A National Science Foundation (NSF) BIGDATA grant proposal has recently been submitted that will continue and expand upon the results of the WCSA initiative. Persistent and unique identifiers for ***Page*** level content directly support the activities of both initiatives.

- **Benefits:** Adoption of a persistent and unique means of identifying ***Page*** level content directly supports both initiatives by facilitating the development of nuanced representations of ***Worksets***, similar collection entities, ***Volumes***, and ***Pages***. Failure to adopt will complicate and slow the work of both initiatives as they will require additional infrastructure to map conceptual entities within their frameworks to actual entities in the existing HTRC infrastructure. The HTRC will also be poorly positioned to leverage research outcomes from the initiatives.

Use Case Scenarios

Persistent Page Citability

In the course of discussing the results of an analysis, a scholar might want to highlight particular examples. In the status quo, the scholar only has the option of a traditional citation (which a reader might then use to identify the text manually), or URLs in the HathiTrust Page Turner, which rely on the volatile page sequence numbers. For example, a scholar might cite:

> Albert Mérat, "Prologue," in L'idole (Paris: Lemerre, 1869), 2–3.

Or, the scholar might offer the URLs
https://babel.hathitrust.org/shcgi/pt?id=hvd.hnx8ey;view=1up;seq=16 and
https://babel.hathitrust.org/shcgi/pt?id=hvd.hnx8ey;view=1up;seq=17, accessed 18 August 2014.

With persistent Page citability, a scholar might be able to make a citation like one of these:

> Albert Mérat, "Prologue," in L'idole (Paris: Lemerre, 1869), 2–3, https://babel.hathitrust.org/shcgi/pt?id=hvd.hnx8ey;page=12345678-9abc-def0-1234-56789abcdef0.
>
> or:
>
> Albert Mérat, "Prologue," in L'idole (Paris: Lemerre, 1869), 2–3, http://www.htrc.org/hdl/12345678-9abc-def0-1234-56789abcdef0.

These citations would refer to the content on the Page in question, even if the leaf were rescanned, OCR were re-run, or leaves were inserted or deleted, changing the sequence number.

Point-in-time Workset/Analytics Module Citability

When a scholar publishes an analysis, it would be desirable to be able to refer to the specific content and specific analysis that was performed. The scholar may have used an Analytics Module published by someone else, and subject to subsequent change; likewise, the content of the Workset may shift e.g. due to rescanning or removal of Volumes.

The scholar might then write something like, "We ran the Data Capsules analysis (http://github.com/htrc/HTRC-Data-Capsules/commit/894ca) on the Ancient Greek corpus (http://www.htrc.org/workset/AncientGreek@miao/version/3)." A reader of the publication would be able either to retrieve the specific point-in-time Workset and Analytics Module and reproduce the results, or would be able to understand that the content and/or analysis had changed, possibly explaining variance in the results.

Persistent File Object Citability

A scholar may wish to comment on specific versions of the digital representation of Pages, e.g. to analyze OCR errors. In that case, references to specific File Objects are required, e.g. "In the scan of page 5 (http://www.htrc.org/content/mdp.1234567/abcd.tiff), the insufficient resolution leads to persistent rn/m recognition errors, as seen in the resulting text (http://www.htrc.org/content/mdp.1234567/hjkl.txt)." The File Object URIs may not be resolvable to the general public reading the publication, but an HTRC-authorized scholar may be able to access the image and see an n-gram of the text page, confirming the first scholar's analysis.

Persistent & Unique Identifier Schemes

There are a number of schemes and specifications that might be adopted to institute the creation of URIs for additional assets within the HTRC context. With regards to URIs there are two routes to be chosen depending on the types of end usage expected (Berners-Lee, Fielding, & Masinter, 2005). For those entities which we expect end users to interact with via browsers, the use of URLs might make the most sense. However, as most of our use cases concern backend consumption of

objects and conceptual entities by the HTRC's services and ***Analytics Modules***, it is the URN that seems to be the most likely candidate for adoption.

Uniform Resource Locators

We considered the following two URL standards: California Digital Library's ARK (Archival Resource Key) Identifiers (Kunze & Rodgers, 2013) and the Handle System Protocol (Sun, Lannom, & Boesch, 2003; Sun et al, 2003). Both provide recipes for minting unique identifiers for a variety of entities but other than some requirements that identifier strings should be opaque and should not be based on changeable attributes of an entity neither system prescribes the exact means by which entity-level identifiers should be constructed. In our context the requirement that entity-level identifiers be opaque would refer to the *order* value that both the HT ***Page***-turner and the Cassandra store leverage to keep track of ***Page*** level entities. Since *order* is an attribute that has already been demonstrated to be changeable, it clearly doesn't meet the requirements of either specification. Despite this requirement neither protocol is focused specifically on identifying entities in a context-free manner, rather both protocols are designed around assigning authoritative identifiers at the organizational level and managing the resolution of resource requests by Web browsers.

Implementing either protocol would be costly, as a dedicated web server would need to be maintained to resolve requests made for HTRC resources by other web agents. Additionally, under the terms of "non-consumptive" research, providing a service that would actually serve ***Page*** level content or the ***File Objects*** themselves out onto the web is undesirable under most circumstances. Finally HTRC would still be left with developing an identifier scheme that opaquely identifies each of the entities for which ARK identifiers or Handles would be minted. Neither of these services truly meet the needs listed above, which are methodologies for ensuring reproducibility of research results and improving the granularity at which HTRC provides services to scholars.

Uniform Resource Names

There are a very large number of URN schemes used worldwide today. They range from ISBNs and ISSNs, through UUIDs, to more esoteric and context dependent schemes such as the mpeg-7 and the European Union's Lex URN namespaces. Of all of the possible URN schema that the HTRC could adopt, UUID appears to be the best fit. It provides a very large range of possible identifiers (3.4×10^{38} possible combinations; Leach, Mealling, & Salz, 2005) which should scale very well to the billions of ***Page*** level entities within the HT corpus and ***File Objects*** that provide representations of them.

Recommendation for ***Page*** level Entities

It is our recommendation that infrastructure for minting and leveraging UUIDs that persistently and uniquely identify both ***Page*** level content and their underlying ***File Objects*** be adopted and implemented. Creating an entity that conceptually represents ***Page*** level content regardless of whether it is represented to the end user agent (which in the HTRC context will frequently be a software agent, i.e., a ***Analytics Module***) as a rendered image file or a text file increases the granularity at which scholars can ask research questions and provides opportunities for the

development of new HTRC services capable of exploiting the expanded conceptual model (see Fig. 1).

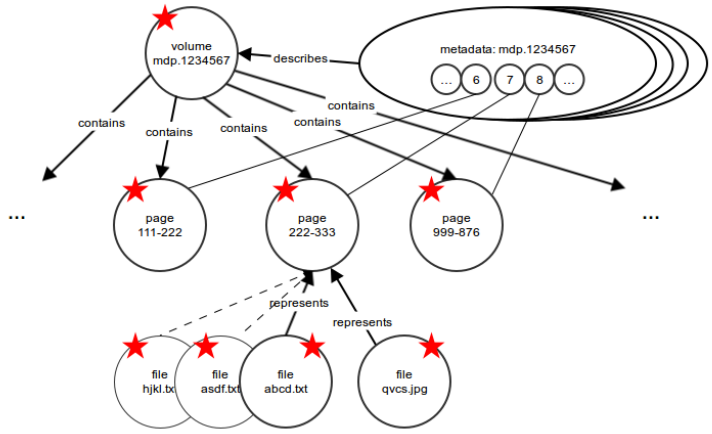

Figure 1: Expanded Conceptual Model for *Volumes*

Overall design principles

As Cassandra only keeps the latest version of ***Page***–level content but, we want to provide *Point-in-time citability*, even if the version in question may no longer be available, it is better to set up another data store (e.g., a RDBMS or graph database) to keep track of persistent identifiers while leaving Cassandra intact. The basic idea is to separate the storage of actual ***Volume*** content from the information necessary to construct identifiers (i.e. to separate ***Page*** level content from that content's relative position within a ***Volume***). No changes to the Cassandra schema design are necessary.

Another rationale is that the query for *Point-in-time citability* needs to be flexible and Cassandra, as a NoSQL store, has very limited support for that (though Cassandra's CQL provides query capability to some extent, it is still far from what we need). An RDBMS can answer query requests more efficiently. However, we do have scalability and performance concerns regarding an RDBMS store. Since we are going to only store identifier data rather than actual content in the RDBMS, then robust indexing can be pre-built to speed up queries. We would only have a moderate number of tables (i.e., three tables for representing the conceptual data model and another two or three tables for maintaining persistent identifiers for HTRC data assets – e.g., ***Worksets*** and ***Analytics Modules***). We expect an RDBMS be able to satisfy our performance requirements. To get a quantified performance measure, we intend to use Neo4j as an alternative and conduct performance comparison between the two.

UUID generation and database records insertion during data ingestion

For the RDBMS, we have the following three tables that are used to represent the conceptual data model as well as keeping track of how a ***Page*** evolves over time, e.g., when a ***Page*** is added, deleted, or rescanned.

RDBMS schema design

Table 1: RDBMS table that stores *Volume* metadata, i.e. various METS files

volumeID	revisionDate	originalMETS	updatedMETs
mdp.1234567	03-08-2014	text blob	text blob
mdp.1234567	04-15-2014	...	...
mdp.1234567	05-1-2014	...	...
mdp.1234567	06-18-2014	...	...

Where 'volumeID' is HT's ID for a ***Volume***, 'revisionDate' is the date when we detect that a revision has occurred through an updated METS, normally this is the date when an rsync is performed and a new METS file is obtained. The column 'originalMETS' stores the original METS XML file that was rsynced from HT and the 'updatedMETS' column stores the METS files that have been supplemented persistent identifiers for ***Pages*** and ***File Objects***. Below is an example of an 'updatedMETS',

```
<METS:structMap>
    <METS:div ORDER="110" LABEL="UNTYPICAL_PAGE" TYPE="page" ORDERLABEL="100">
      <METS:fptr FILEID="TXT00000110"/>
      <METS:fptr FILEID="IMG00000110"/>
    </METS:div>
 </METS:structMap>
```

We could replace the above with something like this:

```
<METS:structMap>
        <METS:div ORDER="110" LABEL="UNTYPICAL_PAGE" TYPE="page" ORDERLABEL="100"
PAGEPID="PAGE_PERSISTENT_ID">
      <METS:fptr FILEID="TXT00000110" FILEPID="FOO"/>
      <METS:fptr FILEID="IMG00000110" FILEPID="BAR"/>
    </METS:div>
 </METS:structMap>
```

We add an extra attribute named 'PAGEPID' indicating the persistent ***Page*** identifier of the ***Page***. Similarly, for each fptr element (scanned image, OCR plain text, etc), we add an extra attribute called "FILEPID" indicating the persistent file identifier. Here we assume that any ***Page***/***File Object*** change (i.e., any addition, deletion, or update) will yield a new METS file when we perform an rsync from HT. We note that out "updated METS" files have a different schema and cannot be parsed by the parser intended for the METS files delivered during rsync. We need to generate a new XSD for the "updated METS" files. In Table 1, 'volumeID' and 'revisionDate' together serve as the primary key.

Table 2: RDBMS table that stores *Pages* and their ordering

volumeID	pageID	order	revisionDate
mdp.1234567	111-222	5	03-08-2014
mdp.1234567	111-222	5	04-15-2014
mdp.1234567	111-222	6	05-01-2014
mdp.1234567	111-222	null	06-18-2014
mdp.1234567	222-333	10	03-08-2014
mdp.1234567	222-333	10	04-15-2014
mdp.1234567	222-333	10	05-01-2014
mdp.1234567	222-333	10	06-18-2014

Where 'pageID' is the minted persistent ID for a ***Page*** and the *order* attribute's value is its sequence number imposed by a METS file. The columns 'volumeID' and 'revisionDate' together serve as the foreign key which allow us to retrieve the corresponding METS file in Table 1. Note that it is possible that a ***Page*** has no order-mapping in METS due to deletion. We can use a special value (e.g., 'null') to indicate this case. We also note that the *order* attribute can actually be derived from the METS file. However, we still store it explicitly in a separate column to speed up the query as we think *order* may be an important attribute to query. This strategy trades space (i.e., it has an extra column) for efficiency. In this table, 'pageID' and 'revisionDate' together serve as the primary key.

Table 3: RDBMS table that stores persistent IDs of *File Objects*

pageID	fileID	accessionDate	deaccessionDate
222-333	5678	03-08-2014	05-01-2014
222-333	6789	06-18-2014	06-18-2014

Where 'fileID' is the persistent ID minted for a ***File Object*** (e.g. a scanned image or an OCR plain text). The column 'accessionDate' is the date of this ***File Object***'s accession, i.e. has a corresponding record within the METS. The column 'deaccessionDate' is the last known revision date that this ***File Object*** still existed. In other words, 'deaccessionDate' records when a ***File Object*** is superseded or deleted, i.e. the date that the ***File Object*** disappeared from the METS file. Under the current workflow, when a ***File Object's*** content is updated, HT will silently supersede the obsolete ***File Object*** by directly writing the new ***File Object*** over the old one, retaining the same filename. We will need to check the MD5 field of a ***Page*** in order to tell whether its content has been changed. In this table, 'pageID' and 'fileID' together serve as the primary key.

We note that even though in Table 3 we only keep the 'accessionDate' and the 'deaccessionDate' of a ***File Object***, by joining Tables 1, 2 and 3 and specifying the time interval starting from

'accessionDate' and ending with 'deaccessionDate', we are able to retrieve all METS that has records for a specific ***File Object***.

We also note that apart from above tables, we need to create tables that keep persistent identifiers for other HTRC data assets, e.g., ***Workset*** and ***Analytics Modules***.

UUID generation

On ingest of a completely new *Volume*, we have following tasks:
Pages must be identified and UUIDs minted for each. On initial ingest, this appears to be as simple as creating one ***Page*** identifier for each div element containing fptr elements in a ***Volume's*** corresponding METS file. Moreover, we associate the ***File Objects*** (e.g. image and plain text representations) named in the fptr elements to the corresponding ***Page*** identifier and mint UUIDs for each ***File Object***. As noted in Table 2 above, the corresponding *order* value is stored and mapped to the corresponding ***Page*** entity.

On ingest of a new version of an existing *Volume*, we have following tasks:
The nature of any changes to a ***Volume*** must be determined. The fact that the ***File Objects*** related to a ***Volume*** have changed may indicate things as trivial as that the METS generation software was updated, or it could indicate more substantial changes have occurred such as that completely new scans or OCR content is available, that the ordering of ***Pages*** has changed, or that any combination of the preceding changes has occurred. Checksums must be considered to determine the exact nature of the changes.

New ***File Object*** identifiers must be minted as needed. Any previously-unknown file must undergo the same process as for initial ingest. Associations between ***File Objects*** and ***Pages*** must be updated. If new ***File Objects*** representing the scanning of newly identified physical pages appear, then new ***Page*** entities within the HTRC context must be created with new unique identifiers. If new ***File Objects*** represent additional or superseded content of previously-known ***Pages***, then the ***Page-File Object*** associations must be updated accordingly. It may not be possible to determine this with 100% reliability; e.g. if all of the ***File Objects*** associated with a ***Page*** that happened to have an *order* value 5 are replaced by new ones, it will be difficult to tell if the old ***Page's*** content was completely removed and replaced with entirely different content, or if new scans and OCR text was simply provided. However, reasonable heuristics seem achievable.

Sequence number (i.e., *order* value) mapping to ***Pages*** must be updated. If it is believed that a ***Page*** has been added or deleted, or that ***Pages*** have been reordered, then that should be noted. Such a change should be detectable by the values contained with the corresponding file element's *checksum* attribute in the METS file but, see the note about heuristics immediately above.

To fulfill these tasks, corresponding records are inserted into aforementioned tables and along the way UUIDs are minted. Using Tables 1, 2 and 3 as an example, on 03-08-2014, a new ***Volume*** 'mdp.1234567' is ingested into Cassandra and persistent identifiers for ***Pages*** and ***File Objects*** are minted and corresponding records are inserted into RDBMS. A ***File Object*** 'hjkl.txt' representing

a ***Page*** (whose assigned id is '222-333') within this ***Volume*** is assigned id '5678' and a record of ['222-333', '5678', '03-08-2014', '03-08-2014'] (in the order of 'pageID', 'fileID', 'accessionDate' and 'deaccessionDate') is inserted into Table 3. On dates '04-15-2014' and '05-1-2014', we perform rsync respectively and get a new METS file. However, by checking the MD5 field within the METS for ***Page*** '5678' or by other auxiliary means, we are assured that ***Page*** '5678' doesn't change. The corresponding record is updated to ['222-333', '5678', '03-08-2014', '04-15-2014'] and ['222-333', '5678', '03-08-2014', '05-01-2014'], respectively. On '06-18-2014', we perform yet another rsync and detect that the checksum of ***Page*** '5678' has changed, though the filename is still 'hjkl.txt'. Accordingly, we mint a persistent ID '6789' for this new file and record ['222-333', '6789', '06-18-2014', '06-18-2014'] is inserted into table 3.

Estimate of person time

Design and implement database

People Hours

Task	Time
Design schemas for RDBMS and graph database, generate and ingest dummy data.	1 week
Performance evaluation between RDBMS and graph database.	1.5 weeks
Decision making on which database to use, schema refine, performance tuning.	1 week

Design and implement identifier minting system

People Hours

Task	Time
Modify ***Volume***/***Page*** ingestion; implement detection algorithm that detects changes in METs; implement JDBC or graph DB operations that insert corresponding records into database. Perform debug and functionality test.	2.5 weeks
Implement code that mints identifiers for existing ***Volumes*** in Cassandra; run the code and insert corresponding records into RDBMS or graph DB.	2 weeks

Extend existing DataAPI

People Hours

Task	Time
Design and implement new RESTful calls to fulfill user queries that contain point-in-time info in resource request; and calls that request identifiers, e.g., query minted id for a ***Page*** based on a sequence number.	2.5 weeks
Design other RESTful calls that retrieve, query and insert records for other HTRC data assets, e.g. ***Workset*** and software packages.	2 weeks

Considering Persistent & Unique Identifiers for Additional HTRC Entities

Within the HTRC milieu there are two additional entities that are of great import to both scholars and the HTRC's internal infrastructure: ***Worksets*** and ***Analytics Modules***. Both of these entities are very different in nature from the ***Pages*** and ***File Objects*** discussed above. While it is tempting to include them within the same workflows that mint UUIDs for both ***Pages*** and ***File Objects*** it is

important to keep in mind that both entities have their own distinct workflows and storage infrastructure. Rather than try to shoehorn ***Worksets*** and ***Analytics Modules*** into the ingestion workflows described above, we recommend the creation of a pair of identifier schema that are specific to these two entities within the HTRC context.

Identifiers for Worksets

Worksets are identified by [workset_name]@[author_username]/version/[version_number] For example, AncientGreek@miao/version/1. To refer dynamically to the most recent version, the version element can be omitted, as in AncientGreek@miao. This could be further abstracted with a handle service, as an optional layer.

Estimate of person time

Implement version controls for *Worksets*

People Hours

Task	Time
Modify settings in the WSO2 Governance Registry to turn on versioning for ***Worksets***, including testing and evaluating storage implications.	2 days

Identifiers for Analytic Modules and other Software Assets

All software that is considered to be an HTRC data asset and is to be given an HTRC identifier is located in HTRC's central, authoritative source control repository system – https://github.com/htrc. The identifier scheme then would be service/software block name/version. For example,
https://github.com/htrc/HTRC-Data-Capsules/commit/123abc456 identifies the HTRC Data Capsule software at a particular point in time. To refer dynamically to the most recent version, the commit element can be omitted, as in https://github.com/htrc/HTRC-Data-Capsules. This could be further abstracted with a handle service, as an optional layer

Estimate of person time

Extend GitHub implementation to also manage *Analytics Modules*

People Hours

Task	Time
Check all Analytic Modules and other HTRC Software Projects into GitHub	2 days

HTRC Service Layers

Data API & Other Layered Services

It is clear that the adoption of all of the measures described above will necessitate changes be made to the HTRC's current Data API service. At present the Data API returns entire ***Volumes*** from Cassandra or specific ***Pages*** given their sequence numbers. The Data API has to be extended to be able to access ***Volume Pages***, given identifiers for ***Pages*** or text/image files.

In addition, the Data API is extended to provide the following services.

- **Content negotiation:** Given a ***Page*** identifier, one might request a specific file by asking for that ***Page*** with a custom content type request, such as ocrText\base, pageImage\base, pageImage\methodX, or ocrText\methodX. That should make the resource available (subject to security considerations, of course), and should also make the direct identifier evident. The current default is effectively something like ocrText\base.
- **Sequence or *Page*-flipping:** Given a ***Volume***, a user might navigate to a ***Page*** by its sequence number. It should be possible for a user to retrieve a persistent ***Page*** identifier for a sequence-located ***Page***.
- **Time-based negotiation:** Given a ***Volume*** or a ***Page***, it should be feasible for the user to request resources relative to a point in time, using negotiation such as Memento. Just as with specific identifiers, a service might report that the resource from the requested time is no longer available, or might refer a user to another service where the resource can be found. If the resource is available, its persistent identifier should be available to the user.

All of these services are facilitated by Tables 1, 2, and 3 above. For example, time-based negotiation for a ***Page*** identifier translates into a lookup in Table 2. If the version of the ***Page*** at the specified time is also part of the current ***Volume*** version in Cassandra, then its contents can be returned, otherwise we notify the client that it is no longer available. Time-based negotiation for ***Volumes*** is achieved as follows: given a ***Volume*** V and a timestamp t, obtain the list L of ***Page*** identifiers for the ***Pages*** of the desired version of V, say the latest version before time t, from Table 2; use time-based negotiation for ***Pages*** in L to construct ***Volume*** V at time t.

Estimate of person time

Extend the Data API

People Hours

Task	Time
Extensions to the Data API including negotiated services.	3 weeks

Resolution and Maintenance of HTRC Identifiers

Associating persistent identifiers with different HTRC data assets requires the maintenance of information needed to look up such identifiers. Given an identifier, one should be able to look it up and determine (a) if it refers to a valid HTRC entity that is available, and how to access this entity, (b) if it refers to a valid HTRC entity that is no longer available, or (c) if it is not a valid identifier. The following table might be used to maintain such information.

Table 4: RDBMS table supporting HTRC identifier resolution service

Id	Resource type	Start date	End date	Access information
AncientGreek@miao	***Workset***	02/17/2014, 12:24:07GMT		<Registry, server htrc2.pti.indiana.edu, path /htrc/miao/***Worksets***/AncientGreek>
AncientGreek@miao.v1	***Workset***	02/17/2014, 12:24:07GMT		<Registry, server htrc2.pti.indiana.edu, path /htrc/miao/***Worksets***/AncientGreek, version 1>
EarlyDickens@userx	***Workset***	02/05/2014, 10:10:34GMT	02/27/2014, 09:31:24GMT	<Registry, server htrc2.pti.indiana.edu, path /htrc/userx/***Worksets***/EarlyDickens>
github.com/htrc/HTRC-Data-Capsules	Software	02/01/2014, 12:20:00GMT		<https://github.com, path htrc/HTRC-Data-Capsules>
github.com/htrc/HTRC-Data-Capsules/commit/894ca...	Software	02/01/2014, 12:20:00GMT		<https://github.com, path htrc/HTRC-Data-Capsules/commit/894ca...>

The access information column contains data in JSON or XML or any other suitable format, describing where HTRC data resources are located and how they may be accessed. Note that different resources may have different methods of access; also, we might want to restrict access to the actual data resource and return only metadata for the resource. The services that write into this database include the workset builder and tools to gather information on software commits into GitHub.

The information in Table 4 is used to efficiently resolve identifiers of HTRC data assets, including those that are no longer available. Identifiers for ***Volumes*** may be resolved either through Cassandra or the HathiTrust handle service for ***Volumes***, Identifiers for pages (logical leaves), and text/image files of pages are resolved through tables 2 and 3.

In addition to efficient identifier resolution, table 4 also ensures uniqueness of identifiers. This is done by maintaining information for every identifier created in HTRC, including those that refer to resources that might have been deleted or are no longer available, e.g., a user ***Workset*** that was deleted after some period of use. Any attempt to reuse an existing identifier is easily spotted and prevented.

The above information could be maintained in an RDBMS table as shown above or as nodes in a graph database. The latter extends easily to use cases involving graph representations of logical groupings of texts or other entities (described in the NSF BIGDATA grant proposal), or ***Worksets*** containing ***Volumes***, ***Pages***, and ***File Objects***. We would need to add identifiers for ***Volumes***, ***Pages***, and ***Files*** as needed to the database. An example is shown below in Figure 2. Initially, the database contains only nodes representing HTRC data assets. Edges are added as logical groupings of entities are discovered.

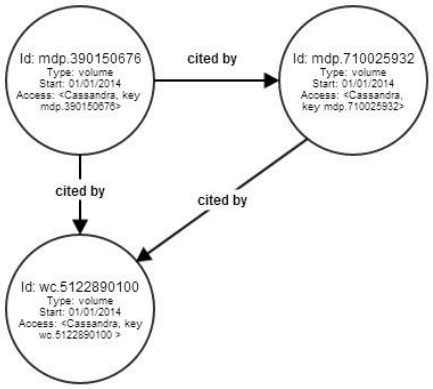

Figure 2: Abstract graph representation of citation mappings

The information in the above RDBMS table or graph database is used by the HTRC local handle service to efficiently resolve identifiers of HTRC data assets, including those that are no longer available. In addition to efficient identifier resolution, this table also ensures that identifiers are not reused by maintaining information about identifiers that refer to assets that might have existed at some earlier point in time but have been deleted since, e.g., a user ***Workset*** that was deleted after some period of use.

Resolution of ingest-date-based versions of ***Volumes*** and ***Pages*** is performed by layered services for time-based negotiation for the same. For example, if the HTRC local handle service were to receive "mdp.123456789/date/2014-01-01", it would first ensure that mdp.123456789 is a valid identifier by looking it up in the above database. If it is a valid identifier, then a call is made to the time-based layered service for ***Volumes***, with arguments mdp.123456789, 2014-01-01, to try and obtain the version of the ***Volume*** corresponding to the given ingest date. For ***Volumes***, if we store only the latest versions in Cassandra, reconstruction of old ***Volume*** versions is feasible if subsequent changes only involve addition or reordering of ***Pages***, and not deletion or rescan or re-OCR of ***Pages***.

Different HTRC services write to the aforementioned database. The initial ingest into Cassandra that creates ***Page*** identifiers and ***File Object*** identifiers, also adds these identifiers and ***Volume*** identifiers to the database. Most writes will occur in the initial ingest phase. Subsequent ingests may have modifications, deletions for limited numbers of ***Pages*** and files, implying limited writes into the database. Other services such as the ***Workset*** builder and tools to gather information on software commits into GitHub also write to the database. But these writes will be fewer in number and sporadic.

Estimate of person time

Implementation of name resolver services for *Worksets*, *Analytic Modules* and other software assets

People Hours

Task	Time
Determine the type of database to use: RDBMS or graph database, design and implementation of the same.	3 weeks
Tool to add identifiers for existing ***Worksets***.	1.5 weeks
Software tool to update database with identifiers for analytic modules and other software projects committed to github.	2 weeks

Handle / ARK Identifier server

A more ambitious alternative to the name resolver service described above would be to implement a Handle or ARK Identifier server to respond to resolution requests for HTRC resources. However, as discussed above, not all HTRC resources are going to be appropriate for the kinds of RESTful name resolver services that a Handle or ARK Identifier server provides. As such we do not recommend the implementation of such a server at this time. We do provide an estimate of the steps necessary and effort involved in implementing a Handle server. This estimate does not include the actual material costs of the server hardware.

Estimate of person time

Design & Implementation of a Handle Server for additional Web-based name resolution services

People Hours

Task	Time
Application and receipt for a unique institutional identifier	1 week
Set up and configuration of Handle server, including implementation of additional customized API services	4 weeks
Updates to existing layered services to operate with Handle server, implementation of local Handle service including reading and interpreting data from the table, interpreting JSON/XML/other access information, directing to layered service for date-based identifiers.	2 weeks
Modify the ingest process to update the database appropriately for ***Pages***, files, and ***Volumes***: add new items, handle deletes by updating the "end date column", create access information in expected JSON/XML/other format etc.	2 days
Add above procedures to ***Workset*** builder for creation and deletion of ***Worksets***.	1.5 weeks

Graphs and RDF-compliant Models

In the future use cases we briefly discuss experiments that envision much of the HTRC infrastructure in the forms of graphs and ontologies. Partially this is because using graphs to model hierarchical relationships, such as those between ***Worksets*** and ***Volumes***, allow them to be much more easily illustrated. The other driving reason for this is that, despite the relatively low uptake Library Linked Open Data, and similar RDF-based data sharing initiatives, there is an increasing

need to experiment with new technologies so that we may realize new functionalities and efficiencies within our ever-evolving technical infrastructures.

The proposals above provide a plethora of identifiers for many of the entities that any graph-based model would likely contain. We can easily adopt additional schema, or even reuse the UUID schema in a new context, to provide additional identifiers for graphs and graph nodes. Careful consideration for where and when to use things such as blank nodes, data and content literals, and management of node identity through good predicate usage will ensure that any graph or RDF-based representation scales with the rest of the HTRC infrastructure.

Conclusion

The current HTRC infrastructure presents humanities scholars a number of opportunities for carrying out novel research tasks. The number of opportunities created for and presented to scholars can be increased a great deal through the adoption of persistent and unique identifiers at the page level. Identifiers supporting ***Page***-level entities directly facilitate scholars' abilities to cite their data sources and provide reproducible results for peer review. Such identifiers also partially ameliorate some security concerns by obfuscating the direct linkages between ***Page***-level analytics results and the ***Volumes*** that they are derived from, e.g., as in the case of feature extraction. Finally persistent and unique ***Page***-level identifiers will support the growth of both new functionality and new arenas of research within the HTRC infrastructure.

The team tasked with formulating this proposal also examined the issues of crafting persistent and unique identifiers for additional conceptual entities within the HTRC milieu, specifically ***Worksets*** and ***Analytics Modules***. In both cases change management is a key factor in minting persistent identifiers for these entities. Because both entities exist in distinct branches of the HTRC's architecture, no overarching benefits can be realized by reusing the same identifier minting strategy proposed for ***Page***-level entities. Because of this a number of alternate methods can be considered and implemented at separate and appropriate times during the HTRC's natural growth processes.

We were also asked to briefly examine the question of version control. The term itself, "version control," is too strong of a descriptor for the actual needs of HTRC's scholarly users. A better notion is that the HTRC's architecture should be "change aware." Change awareness is highly purposeful in that it directly supports the scholarly claim and peer review cycles. For a scholar's research to be reproducible it is not a necessary requirement that another scholar be able to perform the exact same experiment and receive the exact same results. The problem is primarily with how we commonly conceptualize "exact same results" and the expectations that frequently extend from that conceptualization. In the course of both humanistic and scientific research, peers should never have an expectation for receiving the "exact same results." Rather, there should always be the expectation that results will vary and that systems and architectures, along with a discipline's methodological norms, provide ready and useful means for explaining observed variations in results. A change aware architecture plays a vital role in this process by providing researchers information regarding when OCR text has been exchanged for different OCR text, when pages

have been rescanned, or even when entire ***Volumes*** of content have been withdrawn due to copyright concerns.

Ultimately, the best course of action is not to adopt all of the measures proposed here whole cloth. By using the contents of this report to inform long-term planning, various facets and aspects of this proposal can slowly be implemented as an integrated part of the HTRC's strategic growth. This allows for the components of the proposal to be further developed through additional scrutiny and consideration and will maintain a better sense of service continuity for HTRC users. Taking a gradual, step-by-step approach towards the construction of a more robust, change aware architecture will allow all involved in the HTRC project – architects, researchers, and scholarly users – to realize the most benefits of engaging with the HT corpus. Adoption of persistent and unique ***Page***-level identifiers is a necessary and foundational first step towards this future.

References

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *RFC 3986: Uniform resource identifier (URI): Generic syntax*. Network Working Group, The Internet Engineering Taskforce. Retrieved from http://tools.ietf.org/html/rfc3986 on 1-Aug. 2014.

Fenlon, K., Senseney, M., Green, H., Battacharyya, S., Willis, C., & Downie, J. S. (2014). Scholar-built collections: A study of user requirements for research in large-scale digital libraries. Paper to be presented at *The 77th ASIS&T Annual Meeting*. 31-Oct. – 5-Nov. 2014, Seattle, WA.

Kunze, J. & Rodgers, R. (2013). *The ARK identifier scheme*. Network Working Group, The Internet Engineering Taskforce. Retrieved from http://tools.ietf.org/html/draft-kunze-ark-18 on 1-Aug. 2014.

Leach, P., Mealling, M., & Salz, R. (2005). *RFC 4122: A universally unique identifier (UUID) URN namespace*. Network Working Group, The Internet Engineering Taskforce. Retrieved from http://tools.ietf.org/html/rfc4122 on 1-Aug. 2014.

Sun, S., Lannom, L., & Boesch, B. (2003). *RFC 3650: Handle system overview*. Network Working Group, The Internet Society. Retrieved from http://www.handle.net/rfc/rfc3650.html on 1-Aug. 2014.

Sun, S., Reilly, S., Lannom, L., & Petrone, J. (2003). *RFC 3652: Handle system protocol (ver 2.1) specification*. Network Working Group, The Internet Society. Retrieved from https://www.ietf.org/rfc/rfc3652.txt on 1-Aug. 2014.