Roadmapping Session Notes

Notetaker: Adam Jazairi & Sara Brumfield

Presentation slides:
https://drive.google.com/open?id=1k9Tjm9m9w-iR3l97LwtLAFqfzNoJzGwBBh_tgWFEPlk

Top priority is 3.0 release of Image and Presentation APIs.

2 APIs in draft: Change discovery 0.3 -> 0.4 soon (https://iiif.io/api/discovery). Big Question: INCLUDE NOTIFICATIONS or not? Will affect timeline

Content State API 0.2 -- could land in 6-12 months if there is willingness and effort for implementation

Unstarted: Auth and Search -- needs updates to match Image and Prezi 3.
Auth -- complex, always.

- How browsers handle 3rd party cookies has changed since the spec was released, makes in problematic.
- Wide acceptance of the need for a probe service, but not in any spec.
- Need auth support around annotations. Not covered in the current auth specs.
- Needs a TSG, Auth spec was the hardest to write, and won't get easier.

Search

- 2 options, both viable:
    - Formal TSG, look at issues, etc
    - Postpone formal TSG, just bring it up to date with the existing 3.0 specs

What else do we need to be thinking about:

- MMatienzo: Auth support for content search. Material accessible via login, but you can't search that without a login. Simeon: same restrictions as the annotations; isn't there an issue on this?
    - ~~TODO: Open an issue, or check for one.~~
        - https://github.com/IIIF/api/issues/1890
    - Who else is this relevant to:
        - JSTOR is thinking about this.
        - MMcG: FWIW, I think we (Digirati) have the same issue on some projects.
- What about 3D? Is it ready for a TSG?
    - No 3D folks at the working group meeting

    - They are still at the “how are people doing 3D?”
    - Question to ask them: “When do they think there is something that can be done in one of the specs that would make it better than it is today, where they would get a few implementations for both client and server?”
    - When we need to address things in a Z axis? That would drive the need for API 4.0 (image? Or both image and prezi?)
    - Nowhere in IIIF to attach a 3D rendering yet
    - Maybe we should make a recipe for 2D content of a 3D object (??) -- a step towards supporting 3D. (And link to 3D rendering in the manifest). Gives you something to look at in any IIIF viewer, but clients that are looking for 3D data could display it.
    - TODO: Open issue for a recipe for accompanying canvas (if you don’t display, you aren’t missing out — optional additional content), real canvas (the thing being modelled), and rendering (= extra bonus stuff).
- Content negotiation
    - Content negotiation between 2 and 3 -- need a recipe for this. TODO: open an issue for the recipe. (Not in the spec because we don’t want to force folks to use content negotiation). You don’t have to have a 2.0 manifest, you could just publish a 3.0 one (and not need content negotiation). Might be a Level 0 implementation.
    - Content negotiation could be a stumbling block for folks.
    - Level 0 implementations are really good for training
    - Content negotiation is not a requirement, and would add undue complexity to the spec. As a producer of content, I could decide to just flip the switch from 2.0 to 3.0 once I thought everyone was ready for it.
    - Should be more discussion of this topic on a technical community call
- What’s going to drive migration for current publishers? A/V support, better annotation ecosystem. Web Annotation is more widely adopted beyond IIIF, and 2.x spec explicitly uses Open Annotation.

Discovery TSG:
- Going well, will be easy to transfer from TSG to a Spec.
- Change Discovery API
    - Several implementations from client and publisher (OCLC, Glen, Digirati). What’s there is implementable and useful. Discussion of what else we could put in. E.G. what happens if the metadata file goes away?
    - Timeline & Roadmap: When do we want to deal with push notifications from publisher to a client? Very related to Change Discovery spec, but a lot more work. Do we need to do it for 1.0? Should be in the same spec, but different protocol.
    - Some interdependence on the Discovery For Humans CG (to be formed); “profile identities” belongs to the CG/was pushed to them.

    - We are at 0.4; would publish 0.9 when we decide it's "done" to get early adopters and implementors; probably 6 months after that we'd release 1.0
- Content State API
    - A way of agreeing to "states on viewers of particular content" (??).
    - Drag & drop is an example of this.
    - It would be basically a way for users to capture the current view and send it to another user
    - Viewers would have to have a way to export/import content state
    - Initial UV to open on a particular page
    - The basic idea is that the *viewer* (person) needs a way to communicate their state -- i.e. zoom these 3 paintings to this detail on each of them, add this annotations -- to different viewers, for example via a URL. INDEPENDENT OF THE VIEWER. Stored as a web annotation?
    - There's an early draft: https://iiif.io/api/content-state/0.2/
    - How do I stick it in a query string, how do I put it in a HTML XXX.
    - Glen has implemented some patterns
    - Richard Higgins at Durham has implemented some Content State patterns.
    - Does it interact with other activities? We don't believe so, only Presentation 3.0. This needs to pre-date the search API work, because this will tell you how to open viewers at the search result.

What are people's view of the priority of Auth vs Search?

If we're doing auth support for json documents, then we're going to need to do auth and search together. (Rob's assertion: auth and search are intertwined and should be done together)

Auth is very important to archivists, user restrictions are part of any solution to drive more use. (And search is more use?)

How many people have implemented auth? (Stanford, Digirati, Getty, others?)
Search? (Stanford, Digirati, Indiana, Getty, others?)
Similar numbers, half overlapping.

Newspaper CG would be very supportive of a Search TSG rather than a quick update.
Cost is more about cost to reimplement, not about cost to bring the current API up to spec.
Perhaps no one else new would invest in a up-to-spec update to the Search API; If we update to include more use cases then that would drive more implementations.
Text Granularity group looked at how big annotations could get.
There may be people who are ready to implement.

Consensus: a quick update to Search is not worth it. Need to think about TSG or TSGs.

Should we do a joint TSG for Search and Auth? Some in favor some against. Auth is going to be big/hard.

Image and Presentation 3.0 APIs
2 issues for 3.0 remain.

To get to 3.0 “for real”:

- Momentum there for A/V
- Less momentum for Image 3.0
- For images, need 3.0 support for OSD and Leaflet.
- Relatively easy to add 3.0 support to most image servers
- Clients will also need to upgrade OSD/Leaflet (there are more clients than UV and Mirador!! :))
- TODO: Go through the feature implementation list, and see if you are implementing any of the features;
    - Per the community policy, need 2 server side implementations (at least one in production) and 1 client side implementations of any new features before they are approved.
- TODO: IIIF API docs have a great big warning about 3.0 -- can we get them changed/lessened?
    - Yes! → Have created https://github.com/IIIF/api/issues/1891 to track this
- There are a few IIIF 3.0 implementations that are in production (e.g., Avalon at IU/NU, BL) that could demonstrate implementation of certain features
- Recipes don’t count as implemented features. (Because we don’t want to burden the spec with things no one uses.) Recipes prove it’s feasible, but not practical.
- Differentiate to proof of concepts, which do count.

If we say there won’t be more than 1 breaking change per year per spec, and the specs interact, can we actually achieve that. Yes: because we won’t release anything formally fast enough for the other specs not to catch up. Coordination between specs is important for development.

Discovery Spec Notes:

Level 0: URI of changes

Level 1: adds Date/time of changes,

Level 2: Adds type of activity -- 1st creation, update to manifest, deletion, move

Crawlers would retrieve the overall collection (same pattern as annotation collection/pages); crawls through the set of pages, backwards. Those pages contain activities that refer to the manifest that has been changed. Creates a search Index.
Links metadata, so crawler hopefully goes and indexes the metadata (format/standard agnostic). Metadata profile is recommended.
Then you need a search interface that interfaces with the index. I.e. Europeana
Going from that interface to the Viewer UI. (This is where the content state API comes in)
Search ui informs viewer of content change
Viewer reloads manifest

Ecosystem: IIIF registry (proposed) & a crawler layered on top of this. IIIF registry would essentially be a list of links to IIIF collections, from which crawler can fetch references.

I.e. EAD, archives group suggests we use namespace for EAD for a profile to be a document that describes a thing/provide metadata. SeeAlso can point to a lot of different types of formats. (See UCD for early examples of this).

Idea is that communities of practice would evolve, i.e. manuscripts community might want a index that would let you search for “French manuscripts from 1500s”, which a more generic IIIF search index would not let you do. Would expect correlation between community groups and domains.

No generic search API here. (Hasn’t worked in the past, probably wouldn’t work here due to breadth of domains.) Just a way to publish additional data for indexing.
Lightweight way to move search results around (Content state API for that).

DPLA comes up as an example of how challenging this is multiple times. This pattern goes from the manifest to the metadata, IIIF community “controls” the manifest, but if you are harvesting metadata (i.e. DPLA) it would be up to the metadata spec/writer/publisher to figure out how to point back to the IIIF manifest.

Discovery TSG has participation from Europeana (Nuno Freire). In their context they are looking at IIIF manifests at an aggregator level.

Should there be an activity stream about the metadata, separate from the manifest? Current consensus is no, but they’ve engineered the spec so this would be trivial to achieve.

Because items include things like books, a search index would/could index the entire text of the book. To send someone to the particular part of the book that has the search term, we need “content state” -- i.e. “this page at this x,y has the following string” so that search results can link to the exact content state that shows what you searched for.

Link to a page in the viewer that has an annotation as a param in the query string. E.g.:

```
<a
href='viewer.html?iiif-content={"id":"https://example.org/alice/canva
s77#xywh=1000,2000,1000,2000","type":"Canvas","partOf":[{"id":"https:
//example.org/alice/manifest","type":"Manifest"}]}'>
```

Very early on, a drag and drop pattern so you can drag iiif logo from UV to mirador. (It's hard, though! Mouse movements/tabs/dropping in. Good for demos, bad in practice.) This was the starting point for content state.
UV has a way to open with a bunch of URL parameters -- for pages to open to, search terms to highlight.

Tried to find other options for sharing IIIF content:

Copy & Paste pattern -- same as above, but easier to use.

Link through pattern — adds a parameter "iiif-content" and puts the annotation at the end of the URL. QUESTION: why does it have to be encoded?

Download pattern -- download a JSON file that is basically the manifest + annotation state to open in a viewer.

Citation stores -- click "store the citation"; it stores the content state json blob so it could be opened by others.

Why do we need to encode the parameters? Curly braces, etc. are fragile in transit, likely to get corrupted/mangled when sent via email, passed around different web servers.

"URL safe" base 64 encoded. Base64 encoded URL comes at the expense of readability/hackability, but safer/better for robustness in transit.
Robustness requirement -- browser won't accept super long URLs; some characters can't be in URLs (non ascii) QUESTION: does base 64 encoding shorten strings? (I guess so?)

At its simplest, a content state could be `iiif-content="manifest_uri"` Like accepting a manifest on the query string (most viewers are doing that now)

Purpose of the API is to get you from some context of discovery (citation, search results, tweet, email) to an interactive state to view/interact with that object

Further clarification on IIIF registry:
Not a registration of manifests, but a set of URIs to the (hopefully) per-institution collections(?)

Not very heavy weight. A JSON file in github that you can submit a PR to add your collection to. Conceptually similar to Ryan Bauman's IIIF Universe: https://github.com/ryanfb/iiif-universe/ (If you remember PontIIIF, that was an early POC of this based off of iiif-universe.)

TODO: respect "don't crawl me" in robots.txt (Roger Espinosa is opening issue)

What to do when you run into access controls? (Covered in spec). Crawler may not have access and won't self-destruct if it doesn't. Expected scenario. Crawler *may* have the key to access the content. Search creators can decide if you index and show restricted material in search results or not.

OCLC built a PoC(?) of activity streams in their contentDM sites. (Ask Jeff Mixter for more info.) Only publically accessible material)

3 places we have collections:
Item/IIIF collections
Annotations collections
Activity stream collections

These are all the same thing: activity stream collections. The difference is what they collect (items, annotations, activities).

Scalability. 2500 institutions in ContentDM. Consortial customers that have their own collections they might want to index/search, which would result in a massive JSON file. May have to scale the infrastructure to a db to accommodate this, but we'd welcome the challenge as this would be a very positive development!

Existing implementations of activity stream collections: ContentDM, Matt McGrattan, UToronto

Further reading on activities in the Change Discovery API:
https://iiif.io/api/discovery/0.4/#activities

IIIF registry -- unclear what the purpose is, but discovery for humans group will clarify this. The idea (I think) is that if we have a registry than anyone building a IIIF search can pick and choose which available collections they want to include. (i.e. Europeana, DPLA, our mythical medieval manuscript search index, etc)

OCLC demo by Jeff Mixter
ContentDM site-level activity stream:
https://researchworks.oclc.org/digital/activity-stream/site/15/705

Paged. First page has ordered list of manifests, from which you get the references to the manifests themselves.

Search UI for this:
https://researchworks.oclc.org/iiif-explorer
Supports keyword searching across ContentDM sites. Not using content state.
Can create your own collection manifest, derived from search, which you can load in the viewer of your choice.

Lists of content states = just an annotation list.

Is it assumed that the publisher maintains a list of all activities from creation date? Can they truncate old activity?
Total items is total number of activities included in the list you're publishing. Would be ideal to include full list of activities, but you can optionally truncate it.
Crawler would process activities until it hits something an activity it's seen before, then would stop processing.

Activity is around 5 lines of json, so not a lot of storage requirements. OCLC 32M total activities. They just store the data needed to generate the activity streams in an ElasticSearch data store (about 1.2 GB)

What happens for deleted items? There's a "delete" activity that refers to something that you can't get. 410 GONE if it's completely gone or 404 if it's just not there. Returning a 410 is not a requirement in the spec.

FromThePage publishes transcripts as annotations and presents states (how finished something is). Would they use activities to present states of transcripts as they get updated? One thing the TSG has discussed is what happens when something that's not the manifest is updated. But if you don't control the manifest, then this spec doesn't apply.

Where does notification fall in all this?
Optimization: Subscribe to Changes
Notification to the crawler.
Pub/Sub model -- "I'm interested in changes of this type, please tell me when changes occur"; publisher pushes it to the subscriber.
Lowers the latency for updating to the index; doesn't have to wait until the activity stream is crawled.
It doesn't really make sense to crawl an index constantly unless you're working at a particular scale (e.g., aggregators working with 150M objects).

Takes a lot more effort in the Crawler -- needs a webserver that can receive announcements/pubs; content publisher has to publish changes.

Do people think pub/sub protocol is a good idea, or is it overkill? The yeas have it, but many abstained.

You have to support basic activity stream discovery before you can do pub/sub. We could release 1.0 without pub/sub, and then a later version could support pub/sub.

How good is pub/sub now?
Pub: LD notification system is pretty good. Basic inbox system, accepts JSON-LD, is REST. What it doesn't have is the sub part, so we'd be on the hook to figure that out.
Sub: WebSub doesn't work with LD notification, so we'd have to figure that out.

Use Case: When you don't have control of the manifest, but you are a trusted annotator of it. (Jeff Witt's case)

Current Discovery spec already allows for diffs, so is current pub/sub model performance-driven (ping instead of pull)? Yes.

What sort of technology stack would be required to support pub/sub architecture? Rails server + web sockets? Seems like it would be difficult to implement in a trustworthy way. Would be better to release 1.0 without pub/sub, then discuss adding it later.

Adding pub/sub to a 1.0 doesn't feel like it would be a breaking change. Just an additional section. Maybe adds services to the activity stream, but wouldn't be required.

Pub/sub still seems like a good idea and should stay in the charter, but infrastructure challenges require further discussion.

How many requests would it take when say HathiTrust publishes a u
Go to register, get the list of things,
Makes a request to the top level manifest, 10 pages with manifests
10 requests
Each activity stream has
(I'm lost…) (me too) (seem like an order of magnitude of Q requests by X many manifests, but I don't see what that is exactly)

Is there concern about how specialized these patterns are? This is noted in the charter. Some reasons why existing patterns (sitemaps, schema.org, etc.) don't work:
Link to metadata
Ordering of getting changes so you can stop crawling (don't get that from sitemaps)
Looked at schema.org and sitemaps
Go read the notes in the google drive, especially from the TSG discussion in Toronto.
TSG Drive folder: https://drive.google.com/drive/folders/0B_Alni5J8UNITzlpYW1MdnFpSlU
(can't find Toronto meeting folder — might be included in TSG drive above.)

Whew!!! 9+ pages of notes.