Date
11 Oct 2022

Attendees

- Curtis Mirzi, Dave Vieglais, Karen Hanson, Greg Janee, Roxana Maurer, Tom Creighton

Goals

- NISO, resolver, persistence descriptor updates

Discussion items

<table>
<thead>
<tr>
<th>Item</th>
<th>Who</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Announcements</td>
<td></td>
<td>dv: tried converting the ARK spec to markdown format suitable for IETF publication; if this works it can be a much easier way to manage the text than the XML format ACTION: jk and dv to meet to do the conversion to new spec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>jk: The DeSci.com folks are quite interested in ARKs as a citation friendly (shortish), possibly mutable objects that reference parts of IPFS and IPNS; interested in endorsing ARKA all: general agreement that we’re interested in desci’s explorations in this area and about keeping up awareness rm: I’d support mentioning them on the website ACTION: jk: I’ll ask them to fill out the expression of interest form, which we already have procedures for</td>
</tr>
<tr>
<td>Calls for papers, submission deadlines, upcoming meetings: Calendar of events</td>
<td>jk: CNI panel proposal (for December 12-13) was submitted</td>
<td></td>
</tr>
<tr>
<td>persistence descriptors</td>
<td></td>
<td>jk: will meet with FamilySearch; please speak up if you’re interested in joining rm: BnL (National Library of Luxembourg) uses these terms persistence: [ // Persistence statements: <a href="https://datascience.codata.org/articles/10.5334/dsj-2017-039/">https://datascience.codata.org/articles/10.5334/dsj-2017-039/</a> contentVariance: { // how an object’s content will change over time type: String, enum: [ &quot;frozen&quot;, // The bit stream representing previously recorded content will not change. &quot;keeping&quot;, // Previously recorded content will not change, but character, compression, and markup encodings may change during a format migration, and high-priority security concerns will be acted upon (e.g., software virus decontamination, security patching). ] } ]</td>
</tr>
<tr>
<td>Any news items we should blog about?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
"fixing", // Previously recorded content may be corrected at any time, in addition to any change under "keeping".

"rising", // Previously recorded content may be improved at any time, for example, with better metadata (datasets), new features (software), or new insights (pre- and post-prints). This encompasses any change under "fixing".

"molting", //Previously recorded content may be entirely overwritten at any time with content that preserves thematic continuity (homepages).

"waxing" // Change that is limited to appending content in a way that does not in itself disrupt or displace previously recorded content. Examples of waxing objects include live sensor-based data feeds, citation databases, and serial publications.

}

},

objectAvailability: { // the period of time during which the provider expects to keep the object available
  type: String,
  enum: [
    "finite", // Availability is expected to end on or around a given date (e.g., limited support for software versions not marked "long term stable") or trigger event (e.g., single-use link).
    "indefinite", // The provider has no particular commitment to the object.
    "lifetime", // The object is expected to be available as long as the provider exists.
    "subinfinite" // Due to succession arrangements, the object is expected to be available beyond the provider organization's lifetime.
  ]
},

nonReassignment: { // (NR) Once assigned and made public, the identifier will not be reassigned.
  type: Boolean

BnL example: https://persist.lu/ark:70795/m4bk6v?info returns
Spec transition, ongoing collection of ARK spec update use cases

New resolver update from Dave V:

- The N2T resolver replacement is being implemented using the Python FastAPI asynchronous web service framework. Goals of the implementation include minimal resource requirements, maintainable codebase, multiple parallel deployment support, emphasis on API access, and compliance with applicable specifications.

  The purposes of the implementation include:
  - Determine the applicable resolver service given an identifier or portion thereof.
  - Provide information about the resolver(s) available for a given identifier.
  - Periodically evaluate the operation of advertised resolver services.

  Challenges that need to be addressed include dealing with the ARK inflection requests and concocting a resolver service metadata spec.
  - For example, a double ‘?’ (“??”) can be reliably passed through to the web application, appearing as a single ‘?’ since technically the second ‘?’ is the query string. However a single ‘?’ may be consumed at the client, the proxy, the load balancer, the web server, the web server application environment, or the web application framework with no hint to the web application since any of those components may perform some url normalization that removes the single ‘?’ since it is a no-op under HTTP.

New resolver update from Dave V:

- The N2T resolver replacement is being implemented using the Python FastAPI asynchronous web service framework. Goals of the implementation include minimal resource requirements, maintainable codebase, multiple parallel deployment support, emphasis on API access, and compliance with applicable specifications.

  The purposes of the implementation include:
  - Determine the applicable resolver service given an identifier or portion thereof.
  - Provide information about the resolver(s) available for a given identifier.
  - Periodically evaluate the operation of advertised resolver services.

  Challenges that need to be addressed include dealing with the ARK inflection requests and concocting a resolver service metadata spec.
  - For example, a double ‘?’ (“??”) can be reliably passed through to the web application, appearing as a single ‘?’ since technically the second ‘?’ is the query string. However a single ‘?’ may be consumed at the client, the proxy, the load balancer, the web server, the web server application environment, or the web application framework with no hint to the web application since any of those components may perform some url normalization that removes the single ‘?’ since it is a no-op under HTTP.

Spec transition, ongoing collection of ARK spec update use cases

New resolver update from Dave V:

- The N2T resolver replacement is being implemented using the Python FastAPI asynchronous web service framework. Goals of the implementation include minimal resource requirements, maintainable codebase, multiple parallel deployment support, emphasis on API access, and compliance with applicable specifications.

  The purposes of the implementation include:
  - Determine the applicable resolver service given an identifier or portion thereof.
  - Provide information about the resolver(s) available for a given identifier.
  - Periodically evaluate the operation of advertised resolver services.

  Challenges that need to be addressed include dealing with the ARK inflection requests and concocting a resolver service metadata spec.
  - For example, a double ‘?’ (“??”) can be reliably passed through to the web application, appearing as a single ‘?’ since technically the second ‘?’ is the query string. However a single ‘?’ may be consumed at the client, the proxy, the load balancer, the web server, the web server application environment, or the web application framework with no hint to the web application since any of those components may perform some url normalization that removes the single ‘?’ since it is a no-op under HTTP.

New resolver update from Dave V:

- The N2T resolver replacement is being implemented using the Python FastAPI asynchronous web service framework. Goals of the implementation include minimal resource requirements, maintainable codebase, multiple parallel deployment support, emphasis on API access, and compliance with applicable specifications.

  The purposes of the implementation include:
  - Determine the applicable resolver service given an identifier or portion thereof.
  - Provide information about the resolver(s) available for a given identifier.
  - Periodically evaluate the operation of advertised resolver services.

  Challenges that need to be addressed include dealing with the ARK inflection requests and concocting a resolver service metadata spec.
  - For example, a double ‘?’ (“??”) can be reliably passed through to the web application, appearing as a single ‘?’ since technically the second ‘?’ is the query string. However a single ‘?’ may be consumed at the client, the proxy, the load balancer, the web server, the web server application environment, or the web application framework with no hint to the web application since any of those components may perform some url normalization that removes the single ‘?’ since it is a no-op under HTTP.

New resolver update from Dave V:

- The N2T resolver replacement is being implemented using the Python FastAPI asynchronous web service framework. Goals of the implementation include minimal resource requirements, maintainable codebase, multiple parallel deployment support, emphasis on API access, and compliance with applicable specifications.

  The purposes of the implementation include:
  - Determine the applicable resolver service given an identifier or portion thereof.
  - Provide information about the resolver(s) available for a given identifier.
  - Periodically evaluate the operation of advertised resolver services.

  Challenges that need to be addressed include dealing with the ARK inflection requests and concocting a resolver service metadata spec.
  - For example, a double ‘?’ (“??”) can be reliably passed through to the web application, appearing as a single ‘?’ since technically the second ‘?’ is the query string. However a single ‘?’ may be consumed at the client, the proxy, the load balancer, the web server, the web server application environment, or the web application framework with no hint to the web application since any of those components may perform some url normalization that removes the single ‘?’ since it is a no-op under HTTP.

NISO update

- No word from NISO yet
docker minting tool update

- Need to fix minter state
- Added limitations section
- Set expectation of no further (fundamental) development

SYNOPSIS

arknoid - tool to create ARK (Archival Resource Key) identifiers

QUICK START

If your organization doesn't have a NAAN, request one here:

https://n2t.net/e/naan_request

After you install docker on your host, build the arknoid container:

$ docker run -it -d --rm --name arknoid jakkbl/arknoid

Initialize the system with your organization's 5-digit NAAN:

$ docker exec -it arknoid arknoid init 12345

Mint one ARK string with

$ docker exec -it arknoid arknoid mint 1
ark:12345/h74x54g19

Mint enough unique ARK strings to assign to your 35865 objects with

$ docker exec -it arknoid arknoid mint 36000 > MyFirstARKs

You can mint over 70 million ARKs with one minter, and make as many minters as you want whenever you want. Your ARKs will be unique across all minters.

Action items

- John Kunze meet with Dave Vieglais to convert to new markdown spec
- John Kunze move docker-arknoid to arks-org repo, and away from jakkbl dockerhub, and also fix the lost state problem