JHU API-X Use Case: Provenance stream

Title (Goal)	As a data management specialist, I want to see the provenance stream of an object in order to be assured of the integrity and authenticity of an object.
Primary Actor	Data management specialist
Scope	
Level	
Author	Elliot Metsger
Story (A paragraph or two describing what happens)	In order to meet preservation goals, I should be able to ask the repository for provenance information for an object.

As a data management specialist, I want to see the provenance stream of an object in order to be assured of the integrity and authenticity of an object. As a data management specialist, I am comfortable with using URIs to identify objects, and interpreting XML or JSON documents. Knowing the identifier or URI of an object in the repository, I would like to request the the provenance stream that is associated with the object.

Events in the provenance stream may have been generated by the repository itself, or the events may have been generated by external processes that have operated on the object (e.g. a service that performed a format migration). I would expect the provenance stream (especially the events generated by the repository) to conform to a known standard such as Prov-O or PREMIS. The kinds of events I would expect in the provenance stream include (but aren't limited to):

- fixity verification (e.g. a service (internal or external to the repository) successfully verified the checksum of a file)
- model validation (e.g. a service validated the conformance of the object to some model)
- virus checking (e.g. a service scanned the contents of a file for known viruses)
- accession information (who deposited the object, when, by what method, etc.)
- preservation actions (e.g. a service which targets the object for a preservation action)
- object modifications (e.g. correcting descriptive metadata of the object)

Use Case Evaluation

This use case evaluation presumes that the information necessary for generating a provenance stream has already been collected, and the role of the extension is to produce a web resource containing all the desired provenance events. For the sake of discussion, let us assume that we want to produce a provenance stream based on information collected from the fcrepo Audit Service, which can collect internal and external audit events, and represent them as repository resources or triples in a triple store.

Web Resources and Interactions

- This extension defines a URI path segment `ext:provenance` such that for a given repository resource with path `/path/to/resource`, requests to `/path/to/resource/ext:provenance` will be directed to the extension.
- An HTTP GET to `http://host:port/path/to/resource/ext:provenance` will return a representation of the provenance stream.

Preconditions

- Provenance information is presumed to have been collected and available to the extension implementation.
  - The fcrepo audit service can be configured to place audit records in a triple store, or as resources in a specified container
  - 'Available' means that the extension implementation may access it, either through queries to a triple store, or requests to Fedora
  - The extension has been configured to query the appropriate source of events for the provenance stream
- An HTTP request is made to `http://host:port/path/to/resource/ext:provenance` for any repository resource with path `/path/to/resource`

Deployment or Implementation notes

- If the extension is deployed outside of Fedora (e.g. in an external karaf container), without access to an implementation of fcrepo-kernel-api, then the only plausible modes of retrieving the audit events for creating the stream are (a) using the Fedora HTTP APIs, if the events are persisted with Fedora, (b) SPARQL query if the events are persisted in a triple store, or (c) populating a custom index or cache of provenance events, and querying that.
- If the extension is implemented using requests to fcrepo-kernel-api, then it may be required to be deployed into the same container as Fedora.
- Open question: Can the fcrepo WebAC effort be used to secure the ext:provenance endpoints?
- This extension could plausibly be written as a JAX-RS resource and baked into the fcrepo war, if API-X did not exist.
- This extension could plausibly be written as a Camel route which extracts the identity of a Fedora resource from the request, issues appropriate requests to the audit store (fedora repository, triple store, etc), and filters/transforms/joins the results to produce the desired representation of a provenance stream.

Proposed Requirements

- The API-X framework must be able to expose an endpoint on all resources (ext:provenance), for which all HTTP requests are directed to the extension
- It must be possible to restrict access to the provenance stream, preferrably through an existing piece of infrastructure such as WebAC
- API-X should link to ext:provenance for each resource, in order to make it discoverable.

API-X Value Proposition

- Provides discoverabllity of provenance stream resources
- Deployment flexibility, if extension does not need to be co-located in same container as Fedora
- Ease of deployment (compare to fcrepo webapp plus, where it is necessary to compile in extensions)