Open Annotation Collaboration Phase II: Annotation Demonstration Experiments

Developing an annotation framework for Fedora using OAC: Final Report

Primary Investigator:  
Andrew Ashton

Contributors:  
Julia Flanders  
Elli Mylonas  
Michael Park

Center for Digital Scholarship, Brown University Library
**Executive Summary**

Brown University Library’s Center for Digital Scholarship developed an Open Annotation “plugin” for the Fedora Commons repository system. The plugin is a standalone web service that allows client developers to create, query, retrieve, and serialize a variety of annotation types using Fedora’s backend architecture to manage annotations as first-class objects. It was designed to be repository and content agnostic; it allows the creation of annotations of any content type in Fedora repositories using any model or framework (e.g., Islandora, Hydra, etc.). The software and documentation is publicly available under an open source license at [http://brown-university-library.github.com/oac_web_service/](http://brown-university-library.github.com/oac_web_service/).

In order to test the utility of the plugin, we created a demonstration application that can serve as the basis for projects using the Open Annotation standard to work with text collections in research and coursework at Brown. Using this tool, we demonstrate how to annotate different features of TEI texts (sections, strings, marked up data), and how those annotations are created using the Fedora plugin.

**Use Cases and Context**

Brown University is home to a number of prominent collections of texts encoded using TEI. They include the Women Writers Project, the Virtual Humanities Lab, as well as a variety of other documentary collections, such as the US Epigraphy and Inscriptions of Israel and Palestine. This project continued an effort to expand the stable of tools and infrastructure for enabling scholarly interaction with these collections.

Prior to this project, the library’s Center for Digital Scholarship had already developed prototypes of annotation tools in support of faculty research, and had begun to explore the application of OAC to these projects. The earlier prototype work focused on using AtomPub as a protocol for creating annotations of TEI-encoded editions of several works from the Italian Renaissance. These works include Giovanni Pico della Mirandola’s Oration on Human Dignity (1486), as presented in its first printed edition (Bologna, 1496), and Pico’s Conclusiones Nongentae Disputandae(1486), or 900 Theses. In this earlier version of our text annotation tool, an Atom XML document - which contained metadata about the annotation, the content of the annotation, and a URI that defines the portion of the text that the annotation addresses – was sent via HTTP POST to an AtomPub web service, which ingested it into a digital repository. The annotation was thus stored as a new, distinct object in the annotation repository.

These experiments highlighted several key benefits to treating annotations as distinct resources within the web context. For example, a Mexican Pico scholar created annotations containing Spanish translations of each of Pico’s 900 Theses. The Pico Project interface then aggregated that class of annotations into a new representation of the complete work—a Spanish translation of Pico’s text (Figure 1).

---

1 [http://www.tei-c.org](http://www.tei-c.org)
2 [Women Writers Project: http://www.wwp.brown.edu](http://www.wwp.brown.edu)
 [Virtual Humanities Lab: http://www.brown.edu/Departments/Italian_Studies/vhl_new/](http://www.brown.edu/Departments/Italian_Studies/vhl_new/)
Figure 1. Creating a full translation via individual annotations

The Brown Digital Repository, based on Fedora Commons, is emerging as the long-term repository for objects of enduring scholarly interest at Brown. It can interact with a variety of software tools designed to enable meaningful interactions with primary digital objects, and was used in its early form to support a collaborative project between faculty in Brown’s Computer Science and Italian Studies departments. In Spring 2011, students in an Italian Studies seminar, “Garibaldi and the Risorgimento,” used an experimental visual interface developed by the Computer Science department to arrange, annotate, and establish relationships between digital objects housed in the Brown Digital Repository and from other sources (Figure 2).

Figure 2. Developing interfaces for multimedia annotation
In this pedagogical example, the ability to capture, save, and reuse students’ annotations of objects introduces a new dimension to classroom dialogue. Although digital repositories are often equated with published and curated works, annotation and commentary are becoming more prominent in the rhetorical landscape of the digital scholarship. Storing these objects in a persistent and multipurpose store will enhance the potential for comparison and collaboration between students and between classes over time. The Open Annotation plugin was designed to use Fedora as a platform to enable the development of custom interfaces that could be flexible enough to apply to many kinds of scholarly activities, while using a single data model to capture and reuse the products of those activities as first class digital objects.

**Process**

Initially, the Open Annotation Collaboration beta specification raised several compelling questions about approaches to targeting portions of TEI documents at different levels of granularity. 3 Because these questions were germane to the research and scholarship described above, we developed this experiment to examine to what extent could XML fragments be expressed via URIs? What syntax for fragment identification would be the most easily reusable? Would such an approach allow for targeting individual characters? And, how easily would OAC apply to the annotation of encoded semantic data within a corpus? The beta spec relied heavily on the use of fragment identifiers within URIs to specify annotation targets, but offered the possibility of more complex annotation targets via the use of oac:Constraints.

After examining how the OAC beta RDF would be constructed, we determined that Fedora’s built-in Mulgara resource index appeared suited to the task of parsing and indexing OAC RDF. Fedora’s content model includes two reserved datastreams for RDF about an object’s relationships: RELS-EXT (Relationships-External) is an RDF datastream that describes relationships to other objects, and RELS-INT (Relationships-Internal) describes relationships between datastreams of the object in question. The plan was to model the Fedora implementation literally, following the OAC basic model, with an annotation object containing RELS-EXT RDF linking a body object to a target object. Early experiments with this approach revealed a number of challenges, most notably that the stock Mulgara index and RELS-EXT/INT datastreams are constrained in what kinds of RDF statements they can handle. The Mulgara index only listens for content in a RELS-EXT/INT datastream – meaning it was not possible to include arbitrary RDF and have it indexed for the purposes of query and retrieval.4 Furthermore, RELS-EXT/INT only allows triples that have the parent object as the subject, which prohibited these early experiments from implementing all of the OAC relationships in the pre-existing Fedora Content Model.5

The publication of the core specification in May 2012 fundamentally changed the equation for attempting to use Fedora as an out-of-the-box OAC platform. The new data model introduced another layer of abstraction, in that it broke out several pieces of data (e.g., selector strings, specific targets, etc.) into new resources. This affected our ability to map Open Annotation data to existing data structures within Fedora.

---

4 See [https://wiki.duraspace.org/display/FEDORA34/Resource+Index](https://wiki.duraspace.org/display/FEDORA34/Resource+Index) for more on Mulgara in Fedora.
After examining the changes and conferring with colleagues at MITH and in the OA Community Group, we decided to start fresh with a new web service that both sought to adhere more closely to the resource-centric spirit of Open Annotation and addressed several of the issues that arose when attempting to pair OA with Fedora. The results of these changes were two-fold: first, because the changes meant that Fedora could not provide much of the required functionality itself (e.g., indexing and querying the annotation data store), the resulting web service needed to be more robust and fully featured than the one initially proposed. Second, because the new data model both discouraged the use of URLs loaded with information about target fragments and removed the notion of OAC Constraints as a mechanism for addressing different representations of a text, the specific investigation into approaches to targeting TEI via URLs became less central and oac:Constraints became irrelevant. Rather than identifying and supporting a specific set of approaches to TEI annotation, the new software design delegated those decisions to the client – allowing for Fedora to serve as a platform for a wider variety of use cases and enhancing the potential reusability of the resulting tools for the scholarly community.

**Software overview**

The software and documentation for the Open Annotation plugin for Fedora are available at: [http://brown-university-library.github.com/oac_web_service/](http://brown-university-library.github.com/oac_web_service/).

As we set out to develop the second (core specification) version of the web service, we identified several principles that would guide the software development:

1. The software should require no expertise beyond what is required to run Fedora (i.e., a Java web application).
2. The software should require no modifications to Fedora. It should work with Fedora out of the box.
3. Any functions that could be handled by Fedora’s existing APIs should not be recreated (i.e., object creation, editing, deletion, etc.).
4. The software should not require any specific approach to content modeling.

The resulting software acts as an add-on for Fedora, and runs as a separate web application. It allows the creation, querying, and retrieval of Open Annotation core spec compliant annotations as persistent objects in a Fedora repository. Objects can be enriched with further datastreams (e.g., metadata, non-OA RDF, etc.) using the Fedora APIs. Due to the mapping issues outlined previously, the application does not rely on Fedora’s Mulgara index, but instead includes its own triplestore based on Apache Jena for indexing and querying annotation objects in Fedora. In order to keep the triplestore in sync with Fedora, an optional Java Messaging consumer is included to update the OA plugin triplestore anytime an associated Fedora object is modified.

The software created though this experiment requires:

- Instance of [Fedora Commons 3.5](http://fedora-commons.sourceforge.net) running
- Java Servlet container (tested on Tomcat 6/7) in which to install this plugin

It includes a Jython web service with the following methods:
/create

Uses HTTP Basic-Auth. A POST method that creates an Annotation based on the following submitted parameters:

Required Parameters:
- `source_uri`: The URI for the whole target object
- `dc_title`: Dublin Core title associated with the annotation
- `body_inline`: Plain text string to store as the body
  OR
- `(body_content`: Content of the body for a new Fedora body object
  AND
  `body_mimetype`: Mimetype of the body_content)
  OR
- `body_uri`: URI pointing to the body of the annotation

Optional Parameters:
- `annotator`: A string identifying a user (0 or more)
- `generator`: A string identifying the application that generated the annotation (0 or more)
- `oax_style_uri`: A URI for a XSLT stylesheet used to render the whole target object. (0 or 1)
- `oa_selector`: A string with the selector value (0 or 1)
- `oa_selector_type_uri`: Required if an oa_selector is passed in, e.g. 'oa:Fragment'
- `fragment_type`: URI describing the oa_selector type. Optional and only used if an oa_selector is passed in. e.g. 'http://www.w3.org/TR/xpath/
- `body_content_model`: A string representing the body's content model in Fedora

A create request will create an `oa:Body` object in Fedora if `body_content` is provided, and will then use the resulting identifier to create an `oa:Annotation` object. If no `body_content` is provided, only the `oa:Annotation` object will be created. The resulting `oa:Annotation` RDF will be automatically indexed in the included triplestore and immediately accessible via the `/sparql` method (see below).

/flush

Uses HTTP Basic-Auth. A GET method that will serialize all Annotation objects in Fedora. An optional 'format' parameter outputs the serialization in different formats.

Format options are:
- `nt` (default)
- `rdf/xml` or `xml`
- `rdf/json` or `json`
- `turtle` or `ttl`
- `n3`
/rebuild

Uses HTTP Basic-Auth. A GET method that will rebuild the internal TDB index with all of the Annotation objects in Fedora

/show

A GET method that takes a comma separated list of Annotation (A-1) PIDs to serialize as the 'pid' parameter. An optional 'format' parameter to output the serialization in different formats.

Format options are:
- rdf/xml or xml (default)
- rdf/json or json
- turtle or ttl
- nt
- n3

/sparql

A POST or GET method to query the Open Annotations index with a SPARQL Query. This method has READ access to the Annotations index.

Required parameters:
- query: The SPARQL query to execute

Fedora annotation objects created with this service have a default oa-annotation content model (configurable in the application settings). The content model includes the following datastreams:
- Dublin Core
- annotation (rdf/xml)
- specifictarget (rdf/xml)
- selector (rdf/xml)
- inlinebody (rdf/xml) (optional)

Additionally, the code includes Service Definition (SDef) and Service Deployment (SDep) objects for convenient serialization of oa-annotation objects. The service creates annotations with the bindings outlined in Figure 3.
Finally, the web service offers a listener for use with Fedora’s ActiveMQ Java Messaging Service (JMS). When enabled in Fedora, the JMS will notify the listener that a related object has been updated, and the listener will update the triplestore with the newly updated data from an annotation object. Since enabling the JMS in Fedora requires configuration by an administrator, the listener is disabled by default. Information on enabling and configuring the listener can be found in the documentation on GitHub.
Using the plugin

Figure 4. Three types of TEI annotation.

Brown University Library hopes to use this service to support the development of more robust and preservation-oriented interfaces for capturing and reusing student and faculty commentary about texts in the library’s digital collections. A proof-of-concept web application was developed, using a sampling of texts from the Virtual Humanities Lab and the Women Writers Project as test cases. The web application uses the web service to create and query annotations of these TEI texts in three ways: by targeting sections of text as marked up in the TEI using an XPath selector, by targeting spans of characters using TEI’s string-range convention, and by targeting semantic markup in the TEI and linking it to data sources via URIs. Figure 4 shows each kind of annotation applied to one of Pico’s 900 theses.

Example 1 is an annotation of a section of TEI text as rendered by an XSLT stylesheet stored in the Fedora repository as a separate object. In this example, the section is identified by an XPath, which is stored in the annotation as a string in the selector resource. The body of the annotation is first submitted to the repository as a TEI object...
(test:1085), and then the annotation object is created (test:1086). The serialized RDF for Example 1 is:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix fedora-model: <info:fedora/fedora-system:def/model#> .
@prefix oa: <http://www.w3.org/ns/openannotation/core/> .
@prefix oax: <http://www.w3.org/ns/openannotation/extension/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<info:fedora/test:1086> a <oa:Annotation>;
    oa:annotator "aashton@brown.edu";
    oa:generator "OAC TEI Demo web application";
    oa:hasBody <info:fedora/test:1085>;
    oa:hasTarget <info:fedora/test:1086/SpecificTarget>;

<info:fedora/test:1085> a <oa:Body>;
    dc:format "text/xml" .

<info:fedora/test:1086/SpecificTarget> a <oa:SpecificResource>;
    oa:hasSelector <info:fedora/test:1086(selector)>;
    oa:hasSource <info:fedora/test:99>;
    oax:hasStyle <info:fedora/test:1000010118> .

<info:fedora/test:1086(selector) a <http://dbpedia.org/resource/XPath>,
    <oa:FragmentSelector>;
```

**Example 2** is an annotation of a string within the TEI text. The string is targeted using TEI's string-range syntax. In this example, the URL for the TEI string-range specification is provided to identify the type of syntax used in the fragment selector. While this was not part of the core spec in the May 2012 release, it was necessary to include for client development. This is an optional parameter in the creation of the annotation. The serialized RDF for Example 2 is:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix fedora-model: <info:fedora/fedora-system:def/model#> .
@prefix oa: <http://www.w3.org/ns/openannotation/core/> .
@prefix oax: <http://www.w3.org/ns/openannotation/extension/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<info:fedora/test:1088> a <oa:Annotation>;
    oa:annotated "2012-09-07T15:25:24.210999Z";
    oa:annotator "aashton@brown.edu";
    oa:generated "2012-09-07T15:25:24.346999Z";
    oa:generator "OAC TEI Demo web application";
```

---

Example 3 is an annotation that is targeting semantic markup, in this case a personal name, and annotating it with a resource that represents that person. The client can then opt to pull data from that resource for display. In this example the personal name is annotated with a link to the DBPedia resource for that person, and the client is pulling the abstract for display. The RDF for Example 3 is:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix fedora-model: <info:fedora/fedora-system:def/model#> .
@prefix oa: <http://www.w3.org/ns/openannotation/core/> .
@prefix oax: <http://www.w3.org/ns/openannotation/extension/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<info:fedora/test:1084> a <oa:Annotation>;
  oa:annotated "2012-09-07T15:01:32.397000Z";
  oa:annotator "aashton@brown.edu";
  oa:generated "2012-09-07T15:01:32.484999Z";
  oa:generator "OAC TEI Demo web application";
  oa:hasBody <http://dbpedia.org/resource/Avicenna>;
  oa:hasTarget <info:fedora/test:1084/SpecificTarget>;

<http://dbpedia.org/resource/Avicenna> a <oa:Body>;
  dc:format "text/plain" .

<info:fedora/test:1084/SpecificTarget> a <oa:SpecificResource>;
  oa:hasSelector <info:fedora/test:1084/selector>;
  oa:hasSource <info:fedora/test:99>;
  oax:hasStyle <info:fedora/test:1000010118> .
```

7 The resource used in this example is http://dbpedia.org/resource/Avicenna
A fourth type of annotation, which is not represented in the demo example, is **annotation with an inline body**. Examples 1 and 2 create TEI documents as new objects in Fedora, which are then referenced in the RDF to create the body of the annotation. Alternatively, a client can create bodies as cnt:ContentAsText and embed the annotation within the RDF.⁸ An example of this kind of annotation is serialized here:

```xml
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix oa: <http://www.w3.org/ns/openannotation/core/> .
@prefix oax: <http://www.w3.org/ns/openannotation/extension/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<info:fedora/test:955> a <oa:Annotation>;
    oa:annotated "2012-08-31T17:35:08.779999Z";
    oa:annotator "aashton@brown.edu";
    oa:generated "2012-08-31T17:35:08.898999Z";
    oa:generator "OAC TEI Demo web application";
    oa:hasBody <info:fedora/test:955/inlinebody>;
    oa:hasTarget <info:fedora/test:955/SpecificTarget>;
    oa:hasModel <http://www.openannotation.org/spec/core/20120509.html>;

<info:fedora/test:955/SpecificTarget> a <oa:SpecificResource>;
    oa:hasSelector <info:fedora/test:955/selector>;
    oa:hasSource <info:fedora/changeme:35>;
    oax:hasStyle <info:fedora/test:955/inlinebody> a <cnt:ContentAsText>,
        <oa:Body>;
        dc:format "text/plain";
        cnt:chars "some inline body text has been changed" .

<info:fedora/test:955/selector> a <http://dbpedia.org/resource/XPath>,
    <oa:FragmentSelector>;
    rdf:value "/TEI[1]/text[1]/body[1]/div[1]/div[1]/p[10]/name[1]" .
```

**Issues and implications for the Open Annotation specification**

The software developed in this experiment can help to encourage a broader adoption of the Open Annotation specification by aligning it closely to a widely adopted software platform. However, throughout the development process we discovered that Fedora’s

---

⁸ http://www.w3.org/TR/Content-in-RDF
design prevents a precise mapping of the specification to Fedora without significant additional development work. The issues described in this section may be representative of the kinds of challenges that developers seeking to deploy Open Annotation will encounter as they integrate it with existing systems.

**Support for UUIDs**

By default, Fedora does not use UUIDs to name resources such as datastreams. The OA model recommends using UUIDs to support web-scale exchange of annotations. Instead, Fedora uses the object PID (Persistent ID) assigned by Fedora, followed by the datastream name (e.g., *test:123/annotation*). Some Fedora developers, particularly Ben O’Steen at Oxford University, have successfully demonstrated an approach for creating UUIDs and having Fedora assign the UUID to the PID value. However, this approach requires that Fedora be modified to create PIDs using a UUID scheme – the kind of modification that we wanted to avoid in order to create the most widely applicable tool. This creates a problematic trade-off between ease of deployment in local repositories and the overall intention of using Open Annotation as a truly interoperable, web scale standard. Further development of the Fedora plugin could focus on creating UUIDs during indexing and serialization, and offer some handler service to map IDs back to the content in the local repository. This development was considered out of scope for this phase of the project, and more investigation of the approaches to this problem is required.

**Allowing multiple targets**

A similar issue to the lack of UUID support is the issue of locating targets in Fedora datastreams, which are identified using the PID/Datastream-name convention described above. This practice results in the inability to feasibly support multiple targets for a single annotation, since the datastream name needs to be unique. Creating an annotation with a target of *test:123/specific target* is acceptable, but adding a second target that is part of the same annotation object (i.e., *test:123*) would require creating some convention for datastream naming, such as *test:123/specific target1, test:123/specific target2*, etc. This approach was deemed too problematic, and therefore more investigation is required to identify a better approach to supporting multiple targets.

**Dealing with changing resources**

The Fedora plugin does not implement the *oa:State* class for dealing with issues such as time-dependency. This was considered beyond the scope of the project at the time that the specification was revised to include *oa:State*. As an alternative approach, clients querying the SPARQL endpoint for annotations can federate their queries to compare the *oa:annotated* value with the *oa:Body* and *oa:Target* object metadata in Fedora’s SPARQL endpoint, and optionally retrieve the versioned copy of the relevant resource even if the resource has changed. In the TEI examples used for this experiment, this feature was determined to be critical, since changes in the TEI could render XPath selectors completely irrelevant.

**Fragment syntax identification**

One feature found to be missing from the Open Annotation specification was a requirement to define the type of selector syntax being used to target a fragment of a resource. This information is necessary for clients to parse the data in order to properly

---

determine the target of the annotation. The web service accepts an optional parameter (URI) that defines the type of selector submitted. Example:

```xml
<info:fedora/test:955/selector> a
<http://dbpedia.org/resource/XPath>,
```

The problem with this approach is that there is no canonical source for identifiers for this kind of data. Potentially, a resource such as the Unified Digital Formats Registry\(^\text{10}\) could be extended to include selector types, but currently it does not include things like XPath or less standardized syntaxes.

**Clarification of styles extensions**

The *oax:XsltStyle* extension is used in this software to denote the XSLT stylesheet used to render the target object. The Open Annotation specification also defines *oax:CssStyleValue*, which defines the CSS used to render a target object. While both values are useful additions to Open Annotation, they are quite different in the potential impact that they have on an annotation. For example, a change to a XSLT resource could mean that annotated nodes do not display, or that new nodes are displayed and therefore render any XPath selectors obsolete. A change to CSS may have little impact on the integrity of an annotation, or, an annotation targeted at content of a specific color or font could be affected if the selector used references the CSS. Additionally, other types of algorithms, scripts, or transforms could be essential to identifying the proper rendering for a target for an annotation. The Open Annotation specification would benefit from a more generalized approach to identifying resources that must be applied to a resource in order to have it serve as a valid target for a particular annotation. This is an approach that was implied in the obsolete *oac:Constraint* mechanism, but which is still essential for recreating annotations accurately.

**Challenges in deploying the standard**

The development of a flexible and widely used standard for annotation is critical for the long-term viability of scholarly projects in many domains. Because many such projects will have already established a technical architecture, the rate of adoption of Open Annotation will depend somewhat on the ease with which it can be deployed to work with existing technologies. In developing the Fedora plugin, we have attempted to create a tool that would be of use to many projects using Fedora Commons as a platform. The resulting tool successfully supports many kinds of annotations of digital repository objects, and adheres closely to the OA model in function and intent. However, as described above, we – perhaps inevitably - found several areas in which we were forced to compromise in order to deliver a tool that is easy to use and of broad appeal. In some cases, we were unable, at the outset, to support some valid uses of OA, such as annotations with multiple targets. This feature may be essential for some projects, and so additional development may be needed, but the lack of support for multiple targets does not create risks for the interoperability of these annotations. Other compromises, such as the lack of support for UUIDs, do create risks for annotation interoperability. The approach we use – naming some resources with Fedora URIs – risks the possibility of clashing or outdated resources being linked to annotations. It should be possible through prefixing and querying to ensure that the correct resource is retrieved, but that is problematic when annotations begin to be exchanged across the web.

These examples point to a larger need, as the specification approaches a wider release, for implementation and client-development guidelines within the OA community. The OA

\(^{10}\) http://udfr.org/
specification attempts to address many different use cases for annotation, but not all aspects or features of the specification will be needed for every implementation. In order to encourage developers to use OA in a way that will ensure that the fundamental goals of the project are preserved (interoperability, scalability), it would be useful for the OA community to identify what features of OA are baseline requirements for a new OA client to support. A “compliance standard” such as this could be based on the ontology constraints outlined in the specification, but might be packaged in a way that would give developers a clearer sense of which parts of the specification are essential to creating a viable ecosystem of annotations.