## Objective

Provide a mechanism to allow users to manage "simple notes" related to specified sections of an object that currently exists in the repository. Provide playback of Flash videos with annotations. Management is defined as the creation, editing and deletion of these "simple notes."

## WMS, dlr/EDIT, Search and Other Application/Configuration Impact and Implications

The introduction of the annotation tool will impact not only the repository, but many other applications that interface with the repository as well.

1) *Workflow Management System (WMS) Impact* - Annotation objects will have their own content model. Creation and editing of annotation will take place in the annotation tool, annotation objects will not need to be managed from the WMS application.
2) *Search and Display Impact* – When a resource is discovered that has an annotation object associated with it the interface needs to express that the annotation exists and allows the user to access and view the annotation while using the resource. In a brief result list annotations will not display in the results list. When visiting a resource full record, if any annotations exist access will be granted at that point.
3) *dlr/EDIT* – SMAP's for the annotation objects will need to be editable in dlr/EDIT. The record display that dlr/EDIT supplies after visiting a handle will need to be updated. When an object is accessed in this view, if annotations exist, the user needs to be supplied with the ability to traverse to those records and view them with the resource that was annotated. Without this the annotation might not be able to be viewed inside the context of the resource it is annotating. Inline XACML policies need to be editable in dlr/EDIT.
4) *Indexing* - Search index creation needs to be updated. If annotations of a resource exist when that resources search object is created any annotation metadata needs to be added to that resources search object to facilitate discovery. Do not index annotation objects individually.
5) *XACML* – Policy creation will need to support the creation of policies with eduPersonTargetedID information for unpublished annotation access control.
6) *Shibboleth* – Passing of EPPN (eduPersonTargetedID) needs to be enabled to properly identify users.

## Assumptions

1. The annotation tool will use either a progressive (release 5.2) or streaming (release 5.3) video datastreams to create the appropriate time markers for viewing the annotated video.
   - Supporting both video datastream types (progressive or streaming) will account for licensed/restricted videos that will not have a progressive video datastream.
   - It is understood that if a streaming video datastream is used the "scrubbing" during the creation of the annotation will be limited, if not non-existent.
   - If both video datastream types are available, for a particular video, the Annotation Tool will preference the progressive type, thinking the user experience of "scrubbing" along the video will be the best of the two.
   - We will not always have permission to use all licensed/commercial videos for annotation creation. This licensing restriction will be noted and managed in the XACML policy of the commercial/licensed video itself.
   - Created annotations will be viewable using either progressive or streaming datastreams by the end user.
   - Future support, release 5.3, for MP4 formatted video files which can be either streaming or progressively downloaded.

2. There will be forward and backward pointers between the annotation object and the annotated object using the appropriate annotation ontology (hasAnnotation & isAnnotation) that is delivered with Fedora.
3. Relationships will be represented with the Fedora RELS-EXT capability. We concluded that we will not need to implement a relationship management layer to support these relationships, in part because it is a very simple relationship (see next bullet also).
4. Annotations will have states associated with them for which we need to have a well defined vocabulary. An "annotation in progress" (AiP) is one in which the person creating the annotation may work on it over a span of several days or even weeks. An AiP implies that the annotation is not published. An annotation can also be published or unpublished. Once an annotation is complete, the author of the annotation may decide to publish it. As an example, an annotation may be public for the duration of a single semester course and then the faculty member may decide to unpublished the annotation and perhaps re-publish when he/she offers the course several years later.
5. We recommend that AiPs be ingested into Fedora and retrieved from Fedora for further work, as opposed to continuing the work in a WMS type environment (external database).
6. The underlying mechanism for controlling AiPs and publish/unpublished can be the XACML policy datastream.
7. We recommended that annotations be treated like other archived objects, i.e. they are not deleted from the repository. This approach preserves citation integrity and also minimizes management overhead and the risk of introducing a dangling RELS-EXT reference.
8. Once an annotation is marked published it cannot be edited, even if its state is changed to unpublished. This preserves the integrity of the information in the annotation in the event it was cited by an external source. In a future release it would be advantageous to explore the possibility of allowing the creator to create an addendum of a previously published annotation object.
9. Commercial vendors that do not allow their content to be annotated might supply their own annotations as a supplement. Investigation into importing those vendor supplied annotations/segments into the repository will need to be investigate. A user would then typically find this vendor supplied annotation, which can be considered published and not a AiP. At this point a user will need the capability to add their own note to this published annotation. This could be considered an addendum as well, reference assumption #8.
10. In our first release of the annotation capability, progressive video datastreams will be supported. Streaming video datastreams will be investigated in the first release as well.

## R5.2 Capability Overview

1. Provide login functionality; Shibboleth, LDAP, local?
2. Retrieve attributes about logged in user for use in metadata and management of resource annotations.
3. Search and retrieve existing videos with a Flash(FLV)/progressive datastream in the repository. Streaming video will be addressed in a future release.
4. Ability to select a video from results and apply one or many resource annotations to it.
5. Associate users' resource annotations with a repository collection.
6. Generate a resource annotation object in the repository.
7. Enforce controls on to limit its access and search indexing, creation of XACML policies.
8. Create relationships between a resource annotation object and a source video object.
9. Edit existing resource annotations provided they were created by the users initiating the editing.
10. Create a Structure Map (SMAP) for a resource annotation object.
11. Purge existing private resource annotations and related relationships by the user that created them.
12. Provide Flash (FLV) file playback for annotating video.

13. Provide Flash (FLV) file playback of video with annotations in the public search interface.
14. Architect tool for possible open source availability.

## R5.2 Capability Close-up

### 1) Provide login functionality; Shibboleth, LDAP, local?

It is assumed that authentication and authorization will occur through some centralized Shibboleth/LDAP service provided for the NJVid grant. The value *eduPersonTargetedID* will need to be passed in order to associate a user with the annotation he/she is creating; this will be used as the unique identifier. In the event a member of NJVid is not comfortable passing that information that groups user community will not be able to use the annotation tool. An example of *eduPersonTargetedID* for a successfully logged in user would be *cmmills@rci.rutgers.edu*.

Along those lines the possibility that a prospective user might not have those credentials but still needs to annotate objects. In this case a local A/A will exist with the requisite registration capability. All users' classes (faculty/student/etc.) will be to access and create annotations in the Annotation Tool.

### 2) Retrieve attributes about logged in user for use in metadata and management of resource annotations.

After logging in the Annotation Tool will need to be able to access information pertaining to the user that will be needed during the creation of the annotation object. The specific information needed is not available at this time but at least full name, organization, unique ID, role and email will be needed. Additional input from NJVid is being requested.

### 3) Search and retrieve existing videos with a Flash(FLV)/progressive datastream in the repository.

Using the Open Search Service developed for R5.1 the Annotation Tool will need to be able to query for existing objects with FLV datastreams. Only video objects with FLV files will be allowed to be annotated as they have been deemed as having the appropriate copyright. Searching capability will be simple, like a Quick Search, with one text input box and no additional parameters available.

### 4) Ability to select video from results and apply one or many resource annotations to it.

One or more than one resource annotations of a video will occur and the interface for the end-user and the underlying architecture need to accommodate this.

### 5) Associate users' resource annotations with a repository collection.

Annotations will be stored in the collection of the resource that they annotated. The annotation will contain the same mods:relatedItem metadata that was found in the resource being annotated.

### 6) Create a resource annotation objects in the repository.

The Annotation Tool needs to create and ingest a resource annotation object into the repository.

### 7) Enforce controls to limit its access and search indexing, creation of XACML policies.

a) XACML policies are needed to enforce access, read-only, by only the resource annotation creator, only the creator's institution or the entire repository community.
b) XACML policies are needed to restrict an annotation from being indexed by the search engine or not. Would it be more advantageous to index all annotations and rely on XACML to handle access restrictions?
c) XACML policies are needed to support editing of an annotation by the creator only.
d) XACML policies will be stored in line with the annotation object.

See Resource Annotation Content Model specification for more details:
http://rucore.libraries.rutgers.edu/collab/ref/ctm_sawg_annotation.pdf

Possible XACML attributes/values include the following:

Published and Private – Status of annotation

Author (eppn) – Unique user ID to limit access to a specific author

Institution – The affiliation of the creator of the annotation, to restrict access to an institution

## 8) Create relationships between a resource annotation object and a source video object.

The Annotation Tool needs to create and manage relationships specified in the Resource Annotation Content Model.

## 9) Edit existing resource annotations provided they were created by the users initiating the editing.

Using Fedora API's the tool will need to allow for the editing of private annotations by the users that created them.

## 10) Generate a Structure Map (SMAP) for a resource annotation object.

The tool will need to create a Structure Map (SMAP) datastream of the framing points provided for the annotation.

## 11) Purge existing private resource annotations and related relationships by the user that created them.

The tool will have to purge objects and relationships of private annotations when the purge has been initiated by the creator of the annotation.

## 12) Provide Flash (FLV) file playback for annotating video.

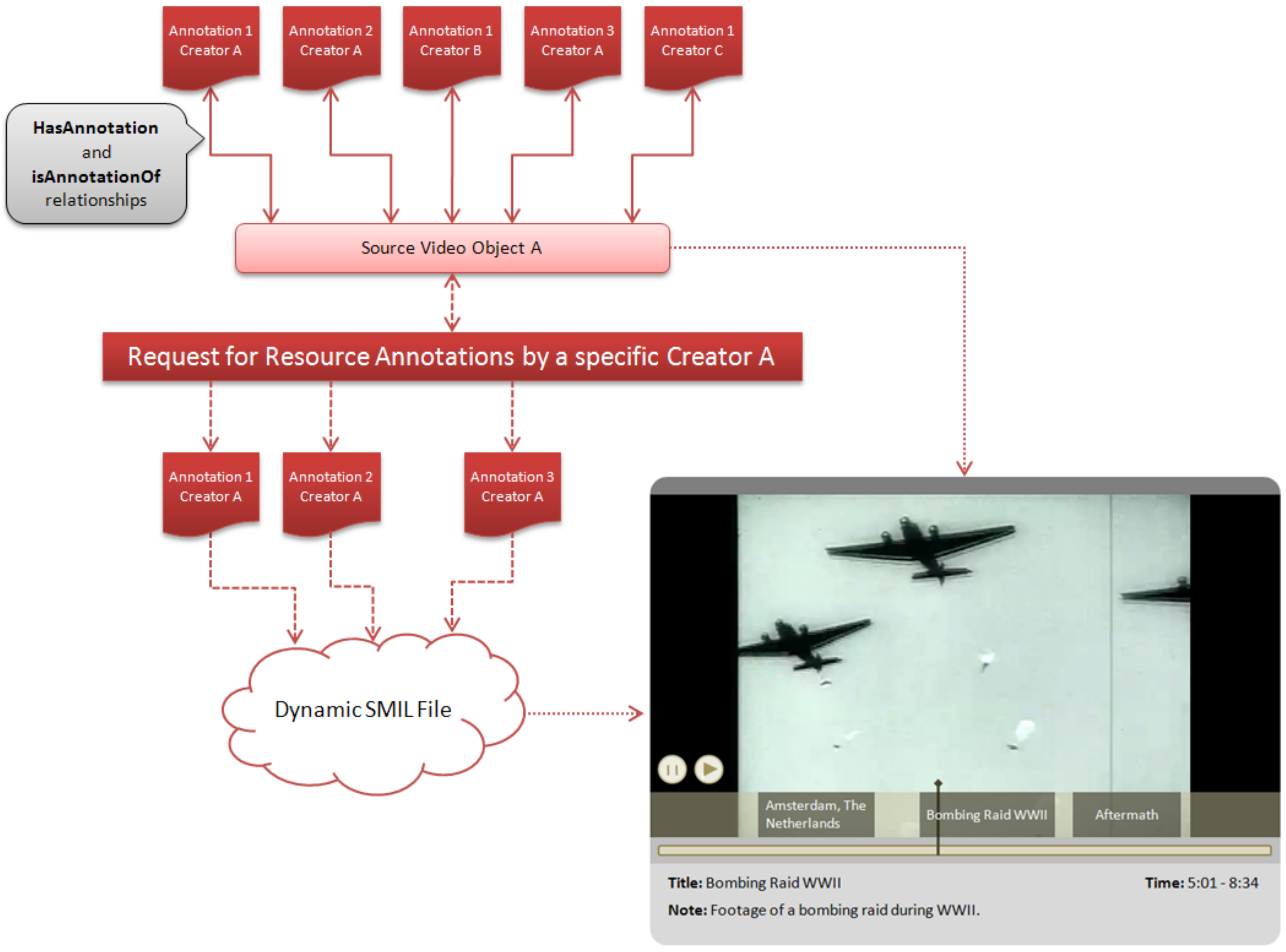
A new flash video player component will be written for playback of FLV datastreams in the Annotation Tool. Scrubbers to generate the framing points for the annotation will be built in. There will be investigation into using file seek techniques in PHP to allow for "jumping" around larger FLV files during playback. Moving forward the annotation tool will need to support MP4 video formats as well.

## 13) Provide Flash (FLV) file playback of video with annotations in the public search interface.

A new flash player video component will need to be written to implement annotation viewing in the public search interface. It could be very similar to the player mentioned in 12) without the addition of scrubbers for determining framing points. The player will need to allow for "jumping" to different annotations of a video by the same creator. The player will need to implement and aggregate all the annotations of a video by one creator for display as well.

Aggregation of annotations related to a particular source video being requested will be needed in order to create a Synchronized Multimedia Integration Language (SMIL) file. The SMIL file will be generated from this aggregation will be used during playback to the user. This aggregation will be performed on the fly by using metadata fields and SMAP information from all resource annotations that are relevant. This SMIL file generation needs to be available for both QuickTime and Flash playback. The aggregation should not be a component of the Flash video player being developed; instead it should be a shared method that can be used by Flash or QuickTime players, dissemination. Should we use the SMIL v2.1 specification when creating the SMIL files? Version 3.0 is still in draft at the W3C, http://www.w3.org/AudioVideo/.

See diagram below of a typical aggregation of multiple annotations by a specific creator of one video. Player used in diagram below is not a final rendition, it is just a mockup created for discussion purposes.



## 14) Architect tool for possible open source availability.

The tool will need to be written and designed in a way that is mindful to the fact that it might be made available to the Open Source community.

## Solution

- **Annotation Tool**

Considering the design issues with creation and association of creators annotations with collection objects, creating and applying XACML policies, ingesting resource annotation objects and editing ingested resource annotation objects developing the Annotation Tool as a module in the WMS would leverage a great deal of established and planned work. The Annotation Tool could be considered a “Faculty Deposit” type application that deals specifically with annotations only. Using the highly flexible design of the WMS the Annotation Tool could then use existing WMS modules to perform a number of the tasks mentioned.

- **Annotation Viewer**

A new flash video player will be written that can implement the decided SMIL version agreed upon. All flash video will benefit from this, not just video being viewed that has annotations.

- **Synchronized Multimedia Integration Language (SMIL) Generation**

SMIL file generation through aggregation of resource annotations needs to be developed and made available when viewing any video datastream, QuickTime and Flash. This needs to be a separate process that can be called upon when needed.