The Pennsylvania State University

Archival Storage Prototype Project

Final Report

Ben Grissinger, Christy Long, Mairéad Martin, Mark Saussure

October 5, 2009

Table of Contents

Introduction 3
Scope 4
Goals 4
Use Cases 4
Description of the Prototype 5
Reference Architecture 5
System Architecture 6
Metadata Development 6
Testing & Evaluation 7
Policy Engine 7
Metadata Engine 7
REST API 8
Data Validation 8
User Interface 8
Project Outcomes 9
Next Steps 10
Appendix A: Reference Architecture 11
Appendix B: System Architecture 12
Appendix C: Hardware, Software and Development Environment Specifications 13
Hardware Specifications 13
Server Software Specifications 13
Development Environment Software Specifications 13
Appendix D: Excerpt of XML PBCore File with System-Level XAM Assigned Metadata 14

Introduction

Since 2007, Information Technology Services (ITS) at Penn State University has been researching ways of addressing the challenges associated with storage support for the tremendous growth of information and data within Penn State University Libraries and the university as a whole. Simple processes used in the past, such as copying data sets, finding a specific subset of data, or replacing storage arrays, have become extremely complex operations that are too large and time-consuming to accomplish within acceptable levels of service. Software solutions built to manage large repositories of data also require application-specific knowledge, long-term migration, and entail integration challenges that often result in large and unwieldy silos of data.

To begin to address these challenges the Storage Networking Industry Association (SNIA) is endorsing the use of storage tiers and classes, data lifecycle strategies and new standards for managing structured data, such as the eXtensible Access Method (XAM). As Penn State began researching the XAM standard, implementation hurdles became apparent: hurdles such as integration with existing applications, working across heterogeneous data silos, and hitherto lack of application of metadata standards in the storage domain. In order to assess XAM's potential, Penn State initiated an Archival Storage Prototype project. The prototype entailed the development of a Storage Services Gateway which resides between applications and storage and consists of a policy engine and a metadata engine. The Storage Services Gateway is enabled by standards-based protocols including AJAX, SOAP, REST, HTTP, and FTP.

The project team included metadata experts who led the definition of technical and administrative metadata in order to perform functions such as searching across federated content, creating retention policies, and routing objects to tiered storage. A fundamental goal of the prototype was to demonstrate how the use of metadata is critical in the lifecycle management of stored data.

Another key component of the Archival Storage Prototype project was the testing and demonstration of XAM capabilities for enhanced storage management. Fundamental to the Storage Services Gateway was the development of a REST API that is capable of performing XAM translations when utilizing the storage cloud. REST is a well-established software architecture for many Web developers and is common in software repositories, therefore enhancing adoption of the XAM standard with little change to existing code or skill sets in addition to promoting adoption of this new standard.

This report describes the goals, architecture, methodology, and outcomes of the Archival Storage Prototype project as well as next steps.

About the Project

Scope

The project scope included testing two storage targets: XAM-aware disks and local (non-XAM) disks. XAM tools were utilized to mask local disks and provide capabilities that are inherent to XAM including assigning metadata such as unique ids, ingest dates and time. These two storage targets in our tier two service catalog offering provide vastly different Qualities of Service, with regard to retention, versioning and full-text searching, and provide a realistic environment for testing and demonstrating policy-based routing capabilities. Penn State also contracted with StorageSwitch, LLC., a storage integrator and technology provider for the fixed content storage market for training on XAM concepts. Ultimately, StorageSwitch generously contributed their base code set to the project.

Goals

The goals of the project are to:

1. Test capability of discovering objects originating from diverse repositories and residing on one or more XAM-aware storage disks

2. Test policy-driven storage management capabilities, such as automatic assignment of digital objects to tiered storage devices

3. Test enhanced storage system functionalities, such as availability, resistance to data loss, data de-duplication, and fault tolerance

4. Develop storage management metadata attributes

5. Promote integration of the XAM standard with various vendor-neutral storage devices

6. Support integration with existing repository software using a storage services REST API, or by means of new application development using XAM libraries

Use Cases

The Storage Services Gateway was designed to support the following use cases:

1. Management of electronic records (institutional data)

2. Digital video management

3. Storage system-level management of diverse digital objects (format, security, compliance and the development of storage management metadata attributes)

Description of the Prototype

Reference Architecture

The Reference Architecture in Appendix A represents the Ingest feature of the Storage Services Gateway. We are conceptualizing ingest functional components as residing Inside the Cloud or Outside the Cloud.

Inside the Cloud components:

1. Policy Execution Engine – Policy-based data management using metadata from XML and XSDs (Example functions are *move object to faster array to accommodate I/O spike*, or execute *retention schedules for digital shredding*)
2. Metadata Execution Engine – Data-driven decisions for storing objects and associated metadata on appropriate storage arrays within the storage cloud (Example functions are *commit large video towards lower cost arrays*, and *OCR, compression, encryption on save*)
3. Management Interface – Administrative tools and reports for collection and object maintenance (Example functions are audit logging, version history, ownership, and permission changes)
4. Metadata DB – Performance improvements for searches, federation and data sharing opportunities, and metadata redundancy
5. REST API – Get, Post, Put, and Delete calls using associated metadata with digital objects

Outside the Cloud components:

1. Authentication – Authentication is performed using bolted-on Shibboleth or Cosign
2. Authorization - Some level of authorization metadata (departmental-level access verification, for example) may need to be incorporated as part of the administrative XML to control access to private or confidential documents (Note that this was not in scope for the prototype.)
3. Digital Objects and XML – Created by submitter and validated by using beans (JSP) that reside on the Storage Services Gateway to parse the XML at the time objects are submitted to disk. Associated XSD (referenced in object XML), which validates context and rules of the object and XML, resides as a virtual view from a database located on the Metadata DB and/or via a disk query to the disks
4. UI Screens – Screens are rendered using XSLT and CSS along with XML and XSDs (This example is one approach for developing XAM-aware applications without using existing software repository packages. Another method is to

bolt-on existing repository software packages using RESTful calls to retrieve, submit, alter and purge objects, metadata, and XSDs via the Storage Services Gateway. In the prototype we have separate UIs for submission of objects with XML and submission/update/creation of XSDs)

System Architecture

The System Architecture graphic in Appendix B shows the components of the Storage Services Gateway. The hardware, software and development environment specifications, including software versions that were utilized for this prototype are detailed in Appendix C.

1. User Interface – HTML, XML, or JavaScript provide the user interface to the Storage Services Gateway for the storage, retrieval, and maintenance of digital objects. Other protocols such as AJAX, SOAP, and FTP may also be used, but were not in scope for this prototype.
2. Storage Services Gateway - The Storage Services Gateway is comprised of two servers for replication and high availability in a primary/secondary configuration. Components of the Storage Services Gateway include a source code management tool (SVN), an application server (Tomcat), a REST interface, Storage Policy Engine, and Metadata Engine. The Metadata DB resides on a separate Oracle server.
3. Storage Arrays - Disk arrays, local and remote, can be presented to the Storage Services Gateway. Two types of local disks were used in this prototype, XAM-aware disks, and local, non-XAM server disks. To test connectivity to remote storage, several publicly available XAM-aware disks outside of Penn State University were utilized.
4. Development Environment - The local development environment consists of a common suite of tools (Eclipse, Maven, Subclipse, SVN client adapter, SmartGWT, Java SDK, XAM SDK, local Tomcat or Jetty server). Project work is checked in to the SVN server and code updates are applied to the Storage Services Gateway server.

Metadata Development

Standard metadata schemas such as METS, MODS, and PBCore were extended to include additional attributes describing and managing digital content being stored through the Storage Services Gateway. Three categories of metadata were used: descriptive, administrative, and technical. Values were created for attributes such as access and ownership, number of copies, retention periods, backup schedules, and version control. System-level metadata was also created to describe file attributes such as file type and size, creation and ingest dates, and storage target information such as specific storage arrays. The latter were developed to support the third use case – “Storage system-level management of diverse digital objects (format, security, compliance & the development of storage management metadata attributes)."

Appendix D provides an example of PBCore metadata appended to system and administrative attributes during ingest of a digital object by the Storage Services Gateway. The system attributes are identified by the field names beginning with "dlt-." These system attributes along with the standard PBCore attributes can be used for policy enforcement during and after ingest.

Some examples of policies executed during this prototype were as follows:

1. Use system-level metadata values such as file type and file size to execute a routing policy to save digital objects to the least cost storage array in the storage cloud based on those attributes

2. Demonstrate retention capabilities using administrative metadata defined retention times of zero days, five minutes and 60 days

3. OCR, encrypt and compress on ingest, based on Boolean assigned values in the administrative metadata

Testing & Evaluation

Testing demonstrated the capabilities of the Storage Services Gateway:

Policy Engine

1. Direct object save to an array by file size:
    a. If > 10 MB, send object to local, low-cost XAM-aware disks
2. Direct object save to an array by file type:
    a. If the object file type is QuickTime (.mov), send object to the lowest cost default storage type
3. Move object to another array by changing metadata from private to public
4. Digital shred capabilities:
    a. Retention time set to five minutes
5. Test OCR, encrypt and compress on ingest

After executing these policies, we were able to examine the newly created system-level storage metadata and verify the object existence and change of disk location.

Metadata Engine

1. Parse XML using example XSD and XML parsing rules
    a. Verify that XSD shows the repository name (Video) and that Video namespace is in the repository
        i. If the folder (Video) exists, add (PUT) the object

        ii. If the folder (Video) does not exist, create (POST) the folder, add (PUT) the object

2. Write XSD object(s) and XML to storage
    a. Submit object destined for Video namespace, verify object in namespace
3. Validate XML parsing
    a. Compare XML file data to XML folder data

Metadata extract functionality was tested by reading in a namespace from an XML file or URL, and verifying that the namespace was created within the Storage Services Gateway.

Metadata and content-level searching tests were also performed across the entire disk pool. Metadata searches worked universally on both the XAM-aware and local disk (simulating XAM-aware disks).

REST API

1. GET
2. PUT
3. POST
4. DELETE

The REST API testing process involved using a scripted CURL command line to test the basic HTTP verbs of REST (GET, PUT, POST, and DELETE) to ingest and retrieve data.

Data Validation

1. Validate XML against XSD process
    a. Submit XML and object via UI
    b. Submit XSD object via UI
    c. Individual and Batch submission via UI

Virtual folder creation was verified in all tests and the associated XML or URL properties were determined to match the batch and individual submission properties.

User Interface

1. Bolt-on SmartGWT user interface (http://code.google.com/p/smartgwt/)
2. Return search information in JSON, text, and XML

A SmartGWT user interface was bolted-on to the prototype to demonstrate the capability and flexibility of utilizing user interfaces that have already been developed. Testing verified that data could be returned in various formats.

Project Outcomes

- Demonstrated the capability to store objects and metadata to XAM and non-XAM storage arrays, and to parse metadata into and out of the storage cloud for UI dissemination and routing of objects
- Implemented a REST API framework to simplify and enhance the adoption of XAM as a standard for saving fixed content with metadata to a storage cloud
- Demonstrated policy-based retention controls using metadata
- Demonstrated basic search capabilities
- Developed and implemented a working code set running on the Storage Services Gateway
- Implemented a unique ID for each object and demonstrated binding and non-binding effects (version control) for saved content
- Worked successfully with storage industry representatives to promote the adoption of XAM across all types of storage devices

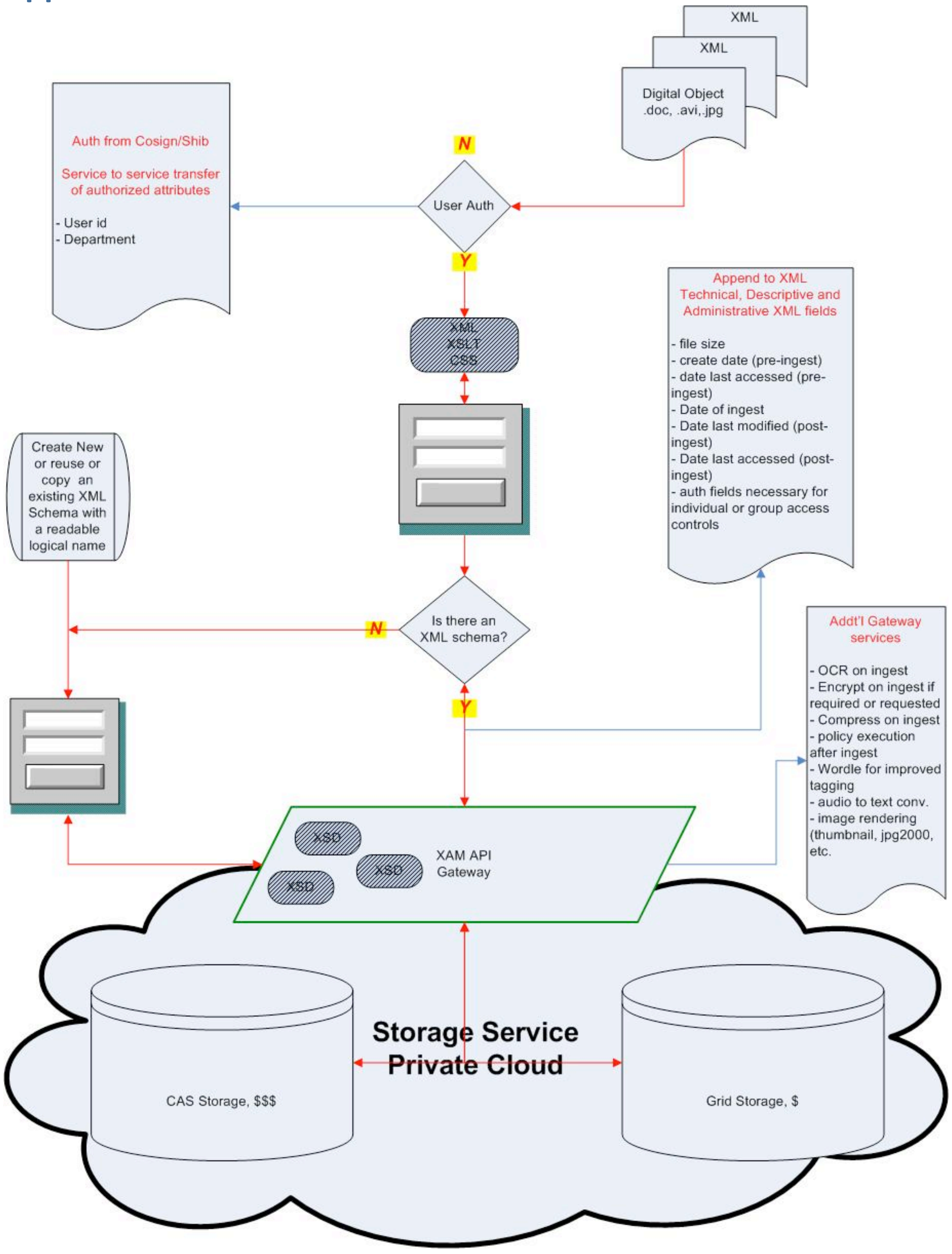
Next Steps

Penn State is currently working to integrate the Storage Services Gateway with an existing repository software application; Fedora was selected because of its native support for the http protocol and ability to reference external disks. This integration will demonstrate the flexibility of the architecture to incorporate existing capabilities and solutions in a diverse repository and storage environment.

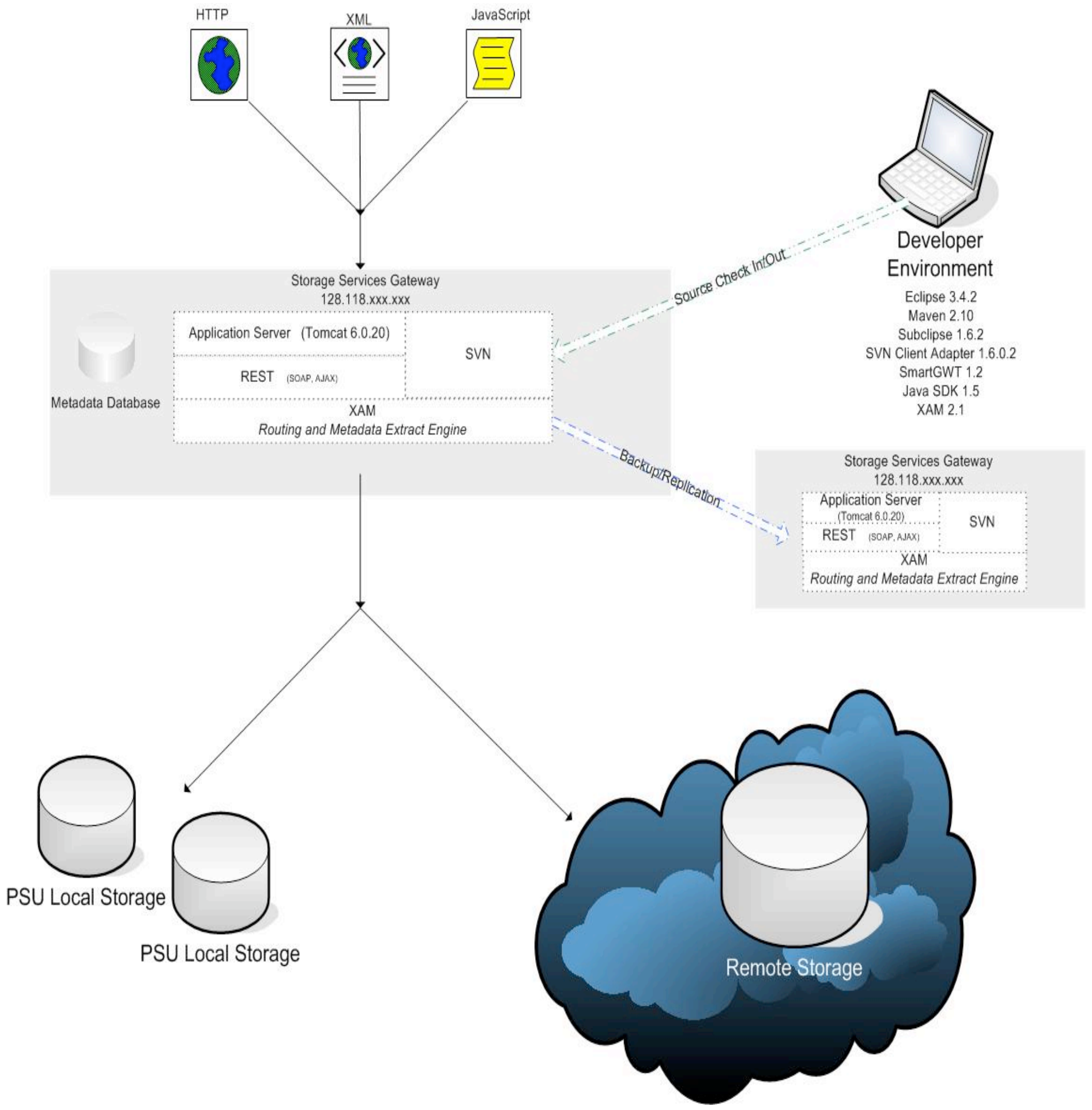
User Interfaces (UI) are also being developed to demonstrate the capabilities and functionality of the efforts to-date from a user perspective. The UIs will focus on ingest, retrieval, and advanced searching of objects. An additional screen will show the capability to route objects to specific storage arrays, revealing multiple storage targets. This project will demonstrate the functional and robust code set of the Storage Services Gateway via a UI utilizing REST and the http protocol.

An immediate goal is to evaluate the outcomes of this prototype project and its potential for meeting the preservation and archival needs of Penn State University Libraries as well as implications for overall storage strategies. In addition, Penn State and StorageSwitch are collaborating to make the Storage Services Gateway code base open source in Fall 2009. In Fall 2009, the results of the prototype will be disseminated with presentations at the Sun Preservation & Archiving (PASIG) SIG, the Storage Network Industry Association annual conference, and the EDUCAUSE annual conference.

Appendix A: Reference Architecture


XML
XML
Digital Object
.doc, .avi,.jpg
N
User Auth
Y
Auth from Cosign/Shib
Service to service transfer of authorized attributes
- User id
- Department
XML
XSLT
CSS
Append to XML Technical, Descriptive and Administrative XML fields
- file size
- create date (pre-ingest)
- date last accessed (pre-ingest)
- Date of ingest
- Date last modified (post-ingest)
- Date last accessed (post-ingest)
- auth fields necessary for individual or group access controls
Create New or reuse or copy an existing XML Schema with a readable logical name
N
Is there an XML schema?
Y
Addt'l Gateway services
- OCR on ingest
- Encrypt on ingest if required or requested
- Compress on ingest
- policy execution after ingest
- Wordle for improved tagging
- audio to text conv.
- image rendering (thumbnail, jpg2000, etc.
XSD
XSD
XSD
XAM API Gateway
Storage Service Private Cloud
CAS Storage, $$$
Grid Storage, $

Appendix B: System Architecture



Appendix C: Hardware, Software and Development Environment Specifications

Hardware Specifications

- 1 processor, 3.20 GHz Intel Xeon dual core, 800 MHz FSB, 4 MB cache, 4GB RAM, 150 GB disk space, Ubuntu 9.02
- 2 processor, 2.83 GHz Intel Xeon dual core, 1066 MHz FSB, 4 MB cache, 4GB RAM, 150 GB disk space, Ubuntu 9.02
- XAM-aware storage, 10 TB disk space

Server Software Specifications

- Tomcat 6.0.20
- SVN 1.4.6
- Java SDK 1.5
- XAM SDK 2.1

Development Environment Software Specifications

- Eclipse 3.4.2
- Maven 2.10
- Subclipse 1.6.2
- SVN Client Adapter 1.6.0.2
- SmartGWT 1.2
- Java SDK 1.5
- XAM SDK 2.1

Appendix D: Excerpt of XML PBCore File with System-Level XAM Assigned Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<PBCoreDescriptionDocument xmlns="http://www.pbcore.org/PBCore/PBCoreNamespace.html"
xmlns:fmp="http://www.filemaker.com/fmpxmlresult" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xalan="http://xml.apache.org/xalan">
<pbcoreIdentifier>
<identifier>Strugg11132007122925</identifier>
<identifierSource version="version1.1">WPSU</identifierSource>
</pbcoreIdentifier>
<pbcoreTitle>
<title>Voices of '69: The Struggle for PSU</title>
<titleType version="version1.1">Air Version</titleType>
</pbcoreTitle>
<pbcoreTitle>
<title>Struggle for PSU</title>
<titleType version="version1.1">VOD</titleType>
</pbcoreTitle>
<pbcoreSubject>
<subject>Discrimination</subject>
<subjectAuthorityUsed version="version1.1">Public Broadcasting Service
PODS Program Offer Data Service Metadata Dictionary
</subjectAuthorityUsed>
</pbcoreSubject>
<pbcoreSubject>
<subject>African Americans</subject>
<subjectAuthorityUsed version="version1.1">Public Broadcasting Service
PODS Program Offer Data Service Metadata Dictionary
</subjectAuthorityUsed>
</pbcoreSubject>
<pbcoreDescription>
<description>Demands from African American students and a ban on a student newspaper spiked student activism at Penn State in
1969. The resulting activities, including a dramatic Old Main sit-in and a visit from Jerry Rubin, are captured in this historic
documentary.</description>
<descriptionType version="version1.1">Program</descriptionType>
</pbcoreDescription>
<pbcoreGenre>
<genre version="version1.1">Documentary</genre>
<genreAuthorityUsed version="version1.1">Public Broadcasting Service
PODS Program Offer Data Service Metadata Dictionary
</genreAuthorityUsed>
</pbcoreGenre>
<pbcoreCoverage>
<coverage>Penn State</coverage>
<coverageType version="version1.1">Spatial</coverageType>
</pbcoreCoverage>
<pbcoreAudienceLevel>
<audienceLevel version="version1.1">General</audienceLevel>
</pbcoreAudienceLevel>
<pbcoreAudienceRating>
<audienceRating version="version1.1">TV-G</audienceRating>
</pbcoreAudienceRating>
<pbcoreCreator>
<creator>WPSU</creator>
<creatorRole version="version1.1">Production Unit</creatorRole>
</pbcoreCreator>
<pbcoreContributor>
<contributor>O'Connell, P. J.</contributor>
<contributorRole version="version1.1">Producer</contributorRole>
</pbcoreContributor>
<pbcorePublisher>
<publisher>The Pennsylvania State University</publisher>
<publisherRole version="version1.1">Copyright Holder</publisherRole>
</pbcorePublisher>
<pbcoreRightsSummary>
```

<rightsSummary>The WPSU web stream contains copyrighted material and may not be redistributed or reproduced. WPSU-FM is licensed for public performance of the BMI repertoire. Copyright ©2004-2009 Penn State Public Broadcasting, an Outreach service of Penn State. Privacy and Legal Statements - http://www.psu.edu/ur/legal.html</rightsSummary>
</pbcoreRightsSummary>
<pbcoreInstantiation>
<dateCreated>1969</dateCreated>
<dateIssued>11/13/2007</dateIssued>
<formatPhysical version="version1.1"/>
<formatDigital version="version1.1">video/quicktime</formatDigital>
<formatLocation>http://podcasts.wpsu.org/Strugg11132007122925.mov</formatLocation>
<formatMediaType version="version1.1">Moving Image</formatMediaType>
<formatGenerations version="version1.1">Moving image/Viewing copy</formatGenerations>
<formatStandard version="version1.1"/>
<formatEncoding/>
<formatFileSize/>
<formatTimeStart/>
<formatDuration>02:21:28</formatDuration>
<formatDataRate/>
<formatBitDepth version="version1.1"/>
<formatSamplingRate version="version1.1"/>
<formatFrameSize version="version1.1"/>
<formatAspectRatio version="version1.1"/>
<formatFrameRate version="version1.1"/>
<formatColors version="version1.1">B&amp;W</formatColors>
<formatTracks/>
<formatChannelConfiguration/>
<language version="version1.1">eng</language>
<alternativeModes>eng</alternativeModes>
<pbcoreFormatID>
<formatIdentifier>Strugg11132007122925</formatIdentifier>
<formatIdentifierSource version="version1.1">WPSU</formatIdentifierSource>
</pbcoreFormatID>
</pbcoreInstantiation>
</PBCoreDescriptionDocument>
<dlt-.xset.management.policy>retention.policy.RepositoryDefault</dlt-.xset.management.policy>
<dlt-.xset.time.xuid>2009-08-24T21:39:08.856Z</dlt-.xset.time.xuid>
<dlt-.xset.retention.base.enabled>true</dlt-.xset.retention.base.enabled>
<dlt-.xset.retention.base.starttime>2009-08-24T21:39:08.856Z</dlt-.xset.retention.base.starttime>
<dlt-.xset.time.residency>2009-08-24T21:39:08.788Z</dlt-.xset.time.residency>
<dlt-.xset.xuid>AAAACwAbnFMAAHhzZXQxMjUxMTQ5OTQ4Nzg4</dlt-.xset.xuid>
<dlt-file.directory>wpsu</dlt-file.directory>
<dlt-.xset.hold>false</dlt-.xset.hold>
<dlt-file.size>2728</dlt-file.size>
<dlt-file.name> Strugg11132007122925.mov </dlt-file.name>
<dlt-.xset.time.creation>2009-08-24T21:39:08.788Z</dlt-.xset.time.creation>
<dlt-.xset.retention.list.base>base</dlt-.xset.retention.list.base>
<dlt-.xset.time.commit>2009-08-24T21:39:08.856Z</dlt-.xset.time.commit>
<dlt-.vnd.com.a1.archive.xset.user>wpsu</dlt-.vnd.com.a1.archive.xset.user>
<dlt-.vnd.com.a1.archive.xset.version>1.0</dlt-.vnd.com.a1.archive.xset.version>
<dlt-metadata.file.namespace>/dlt-rest/file/demo</dlt-metadata.file.namespace>