Abstract: This document provides the Preservation DataStores prototype enhancements.
Delivery Type: Report
Author(s): IBM
Approval: David Giaretta
Summary:
Keyword List:
Availability: PUBLIC

Document Status Sheet

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Comment</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>1_0</td>
<td>31 May 2009</td>
<td>Final version</td>
<td>Simona Cohen, Aner Hamama, Ealan Henis, Petra Reshef, Shahar Ronen</td>
</tr>
</tbody>
</table>
Project information

Project acronym: CASPAR
Project full title: Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval
Proposal/Contract no.: IST-2006-033572

Project Officer: Carlos Oliveira

Address: INFSO-E3
Information Society and Media Directorate General
Content - Learning and Cultural Heritage
Postal mail: Bâtiment Jean Monnet (EUFO 1167)
Rue Alcide De Gasperi / L-2920 Luxembourg
Office address: EUROFORUM Building - EUFO 1167
10, rue Robert Stumper / L-2557 Gasperich / Luxembourg
Phone: +352 4301 33052
Fax: +352 4301 33190
Mobile:
E-mail: Carlos.Oliveira@ec.europa.eu

Project Co-ordinator: David Giaretta

Address: STFC (formerly CCLRC), Rutherford Appleton Laboratory
Chilton, Didcot, Oxon OX11 0QX, UK
Phone: +44 1235 446235
Fax: +44 1235 446362
Mobile: +44 (0) 7770326304
E-mail: d.l.giaretta@rl.ac.uk
CONTENT

1 INTRODUCTION ................................................................................................................................. 5
  1.1 PURPOSE OF THIS DOCUMENT ..................................................................................................... 5
  1.2 HOW TO READ THIS DOCUMENT .................................................................................................. 5
  1.3 GLOSSARY ....................................................................................................................................... 5

2 PDS BACKGROUND ............................................................................................................................ 7

3 PDS PROTOTYPE ................................................................................................................................... 9
  3.1 MAIN FEATURES ................................................................................................................................ 9
  3.2 PDS INTERFACES .............................................................................................................................. 10
  3.3 SCREEN CAPTURES .......................................................................................................................... 11

4 REFERENCES ........................................................................................................................................ 14

FIGURES

Figure 1: Preservation DataStore architecture ......................................................................................... 8
Figure 2: IngestAip demonstration ........................................................................................................ 11
Figure 3: accessContentDataAsLink demonstration .............................................................................. 12
Figure 4: accessRepInfo demonstration ................................................................................................ 12
Figure 5: accessProvenance demonstration ............................................................................................ 13
Figure 6: accessFixity demonstration ..................................................................................................... 13

TABLES

Table 1: Glossary ..................................................................................................................................... 5
Table 2: PDS Performance and Scalability Measurements ..................................................................... 9
Table 3: PDS manager interfaces summary .......................................................................................... 10
1 INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

The purpose of this document is to describe the Preservation DataStores (PDS) prototype enhancements. This prototype is installed in CASPAR SVN as well as deployed in ESA and ASemantics, and serves as the storage component of the CASPAR project.

This document provides an overview of the features and usability of the PDS component, as well as links to actually deploy, run and use the PDS prototype. The PDS prototype may be further enhanced when we publish it in AlphaWorks (www.alphaworks.ibm.com) in the end of the project. AlphaWorks is a publicly open site for software downloads.

1.2 HOW TO READ THIS DOCUMENT

Section 2 gives a short background on PDS functionality and architecture. Section 3, which is the focus of this document, describes the PDS prototype and provides links to how to deploy the prototype and use it.

1.3 GLOSSARY

Table 1: Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit preservation</td>
<td>The processes used to ensure that the bits comprising a preserved object are not lost or become corrupted over time. These processes include refreshing, backups, and error correcting code modules.</td>
</tr>
<tr>
<td>CASPAR</td>
<td>Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval</td>
</tr>
<tr>
<td>Content data object</td>
<td>The 'raw' data that makes up the content to be preserved</td>
</tr>
<tr>
<td>Content information object</td>
<td>The raw data and the metadata needed to interpret it; namely the content data plus its representation information</td>
</tr>
<tr>
<td>Context</td>
<td>The relationship of the content information to its environment, initially as perceived by the content data object provider. Later on, more context information may be implied or derived from the preservation process</td>
</tr>
<tr>
<td>Designated community</td>
<td>The primary OAIS user group that needs to access and understand the information preserved in the OAIS system. This means that the OAIS must have an appreciation of the community's knowledge base.</td>
</tr>
<tr>
<td>Digital preservation</td>
<td>The series of managed activities necessary to ensure continued access to digital materials for as long as necessary.</td>
</tr>
<tr>
<td>Fixity</td>
<td>The information that documents the authentication mechanisms and provides authentication keys to ensure that the content information object has not been altered in an undocumented manner.</td>
</tr>
<tr>
<td>Logical preservation</td>
<td>The processes used to ensure the understandability and usability of the data, in spite of the unknown changes in technologies and users in the future.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Metadata</td>
<td>Information about the content data object that is needed for its preservation.</td>
</tr>
<tr>
<td>Migration</td>
<td>The act of moving data from one system to another because of a change.</td>
</tr>
<tr>
<td>OAIS</td>
<td>Open Archival Information System. An ISO standard (ISO 14721:2003 OAIS) that specifies a reference model for an archive, consisting of an organization of people and systems that have accepted the responsibility to preserve information for a designated community.</td>
</tr>
<tr>
<td>Preservation system</td>
<td>An archiving system in which the lifetime of the data it needs to store exceeds the lifetime of the program/format with which the data is interpreted and the lifetime of the media that stores the bits.</td>
</tr>
<tr>
<td>Preservation aware storage</td>
<td>The storage component of a digital preservation system that has built-in support for preservation.</td>
</tr>
<tr>
<td>Preservation DataStores (PDS)</td>
<td>A new OAIS-based preservation aware storage. It will also serve as the storage component of CASPAR infrastructure.</td>
</tr>
<tr>
<td>Provenance</td>
<td>The information that documents the history of the content information. This information describes the origin or source of the content information, any changes that may have taken place since it was originated, and who has had custody of it since it was originated.</td>
</tr>
<tr>
<td>Representation information</td>
<td>The information that is required to interpret the content data object (raw data) into more meaningful concepts (ultimately more meaningful for humans).</td>
</tr>
</tbody>
</table>
2 PDS BACKGROUND

Preservation DataStores (PDS) are OAIS-based preservation aware storage that focus on supporting logical preservation [1, 2, 3]. They are aware of the structure of an archival information package (AIP) and offload OAIS derived generic functions such as representation information inclusion, provenance tracking, fixity computation, and migration support to the storage layer. They provide strong encapsulation of large quantities of metadata with the data at the storage level and enable easy migration of the preserved data across storage devices.

The following list includes the major requirements and desired features of an OAIS-based preservation aware storage such as PDS:

- In the storage, encapsulate and physically co-locate the raw data and its complex interrelated metadata objects, such as representation information, provenance, and fixity. This ensures that the metadata needed for interpretation is not separated from the raw data and thus never lost (if the raw data survives).
- Include the representation information of metadata (e.g., representation information of fixity and provenance) so the metadata can be interpreted when accessed in the future.
- Utilize the locality property and execute data intensive functions such as fixity (i.e., data integrity) computations within the storage component. This will lessen the network bandwidth and reduce the risks of data loss.
- Handle some of the provenance events internally. The applications on top of the preservation aware storage should be free of managing events that can be handled internally in the storage. Moreover, this enables richer types of provenance events and the inclusion of events related to the migration between physical medium and the transformation of representations.
- Support the loading and execution of external transformations during the migration process and facilitate on demand triggering of these transformations.
- Support media migration, as opposed to system migration. In media migration, performing migration from one system to another can be done by physically detaching the media from one system and attaching it to the new system.
- Maintain referential integrity including updating all the links during the migration process so they remain valid in the new system. This requires an awareness of certain metadata fields that represent links, both internally to the system and externally.
- Ensure readability of the data by a different system in the future. This is done by developing and supporting global self-describing media independent formats.
- Support a graceful loss of data. Some portions of the data are likely to be lost or become corrupted over time. If some data is lost, a good preservation system must minimize the economic effect of this data loss and prevent cases where data that is still intact in the system cannot be read or interpreted.

The PDS architecture includes a stack of three layers based on OAIS, XAM and OSD, respectively [4]. Figure 1 depicts the general architecture of PDS. The top layer is an OAIS-based preservation engine, which provides preservation functionality for heterogeneous data and applications. It includes efficient generation and placement of AIPs along with support for migration and data transformations performed within the storage. The second layer includes an eXtensible Access Method (XAM) library. XAM is an emerging storage standard intended for reference information that provides a logical abstraction for data containers. The bottom layer includes either a file system or an Object-based Storage Device (OSD), which has an object-based interface and built-in access control mechanism.

In OAIS, AIPs are the information objects that are passed to and from the storage component. Thus, in PDS, AIPs are the main objects in the interface. While PDS has a generic interface, our current
implementation supports both a direct API as well as a web services API. We chose a standards-based web services API since web services are platform independent and support clients built within different environments; this is an important feature in preservation environments.

Our proposed PDS is constructed of two processes. The higher level process includes the upper two layers and is responsible for implementing OAIS functions, providing the interface to the outside world, and incorporating XAM functionality. The lower level process includes the bottom layer and provides the storage in the form of the object store or a file system.

The higher level process is composed of a stack of the following components:

- Preservation Engine – provides the external interface and the OAIS specific preservation semantics.
- XAM Library – generates logical container objects of data and metadata under a common globally unique name.
- XAM to OSD or XAM to FS – a bridge that maps the logical XAM objects to physical objects either in a file system or in an OSD.

The second process in the PDS box serves as the object layer and includes an OSD component or a file system. The OSD process requires periodical communication with the security admin, an external component used to obtain security credentials that enforce access control. In the proposed architecture, the OSD calls a third-party security admin via an API or web services.

In the lack of a XAM over a filesystem implementation, we may prefer to omit the XAM layer when using a filesystem.
3 PDS PROTOTYPE

3.1 MAIN FEATURES

The PDS prototype includes the following main features:

- **Basic preservation functionality** – different ways to ingest and access of AIPs packaged in XFDU or SAFE formats. The ingest operation consists of unpackaging the AIP, assigning an AIP identifier, validating and computing its fixity, updating its provenance and reference, and storing each section separately for future access and manipulation. Access includes fetching and validating the data and metadata of the AIP. Each section of the AIP (content data, RepInfo, fixity, provenance, etc.) may be accessed separately.

- **Preservation of data objects** – includes supporting data transformations to maintain the usability and understandability of the content data objects in the future.

- **Authenticity management** – includes preserving and maintaining the PDI sections of the AIPs. PDS performs some of this work automatically while allowing external authenticity management by providing APIs to manipulate the PDI. You can find more about this in [5].

- **Integrating with existing archives** – includes various architectures, models and API to integrate existing archives with PDS. The data in existing archives is generally not packaged as AIPs so PDS added an AIP Generator component that automatically collects disparate metadata from the archive and generates AIPs. You can find more about this in [6, 7].

- **RepInfo management** – allows sharing, search and categorization of RepInfos. All RepInfos are clustered into categories which provide a second layer of hierarchy to RepInfo. When accessing the RepInfo of a content data object or a PDI record, PDS will return the category RepInfo AIPID. To find out which content RepInfos are contained in this category, the user should call the “getRepInfosInCategory” API. You can find more about this in [8].

- **Preservation policies** – AIP preservation policies may be added on ingest or manipulated later on. Currently PDS supports only the following policies: (1) choose the fixity algorithm to be used on the AIP, and (2) choose whether to preserve the original AIP, packaged and untouched, as it was ingested, along with the regular AIP preservation path that unpackages the AIP and handles the preservation of its different sections.

In addition, we have conducted performance and scalability tests. We tested ingest and access of AIPs of different sizes and measured the duration of each such operation. Here are the results:

<table>
<thead>
<tr>
<th>AIP size</th>
<th>24K</th>
<th>1M</th>
<th>8M</th>
<th>40M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of performed ingests</td>
<td>1600</td>
<td>1600</td>
<td>1500</td>
<td>800</td>
</tr>
<tr>
<td>Average ingest duration</td>
<td>854 ms</td>
<td>917 ms</td>
<td>1563 ms</td>
<td>4693 ms</td>
</tr>
<tr>
<td>Number of performed accesses</td>
<td>1600</td>
<td>1600</td>
<td>1500</td>
<td>800</td>
</tr>
<tr>
<td>Average access duration</td>
<td>173 ms</td>
<td>441 ms</td>
<td>731 ms</td>
<td>2662 ms</td>
</tr>
</tbody>
</table>

PDS can be deployed locally in the testbed location. To do that, just download pds_deploy.tgz deployment package from SVN (under sandbox\IBM) and follow the instructions listed in readme file contained in the package.

PDS can also be used remotely on a server deployed in one of the following locations:

2. ESA - http://pcgrid07.esrin.esa.int:8080/
The PDS example client provides a GUI interface to communicate with PDS server. In the next section, you can see some screen captures of this client. To use it:

1. Deploy PDS client and configure it according to the readme file.
2. From a web browser, go to http://<deployment_server_ip>:<web_service_port>/PdsWebClient/PDSDemo/Index.jsp

### 3.2 PDS INTERFACES

The PDS interfaces expose the PDS entry points and also a set of interfaces to support these entry points. The entry points may be called directly or via web services. The PDS interfaces aim to be abstract, technology independent and to survive implementation replacements. It provides a generic API to support the preservation aware storage functionality.

Full Java docs of PDS API are available at the CASPAR SVN (software/java/interfaces/datastore/doc) and also at the CASPAR Wiki [10, 11]. Note that in the web service version all API arguments and return values are passed as XML files, which comply with the PDS interface specification. The PDS deployment package contains the XML schema’s used and sample XML files.

The user can base his code to call the PDS web services on the PDS client code available on CASPAR SVN (under software/java/implementation/datastore/client) and CASPAR Hudson: http://developers.casparpreserves.eu:8080/hudson/job/CASPAR-PDS/. Additional examples of components that embed web services calls to PDS are the PACK component available at the CASPAR SVN, and ACS test component. The PACK component can be used to generate the packaged AIPs.

PACK and PDS are well integrated.

For self-containments, we describe here the interfaces and entry points in high level. The entry points are defined in a set of PDS manager interfaces including PDSManager, PDSIntegratedManager, PDSMigrationManager, PDSPdiManager, PDSRepInfoManager and PDSPackagingManager. Other interfaces are used by the entry points as arguments and/or return values. The other interfaces are basically data structures used by the PDS entry points. Their methods are not exposed as web services and are called locally to build the arguments required for the different entry points and to retrieve the values returned by API calls. When using the PDS entry points via web services, these structures are streamed into XMLs.

The API may return different exceptions that are also PDS interfaces.

Some of the structures used by the PDS entry points, such as the PDI sub-section records’ inner-structure (e.g. the inner structure of a Provenance record) may have different variants, depend on the source that generated the record. PDS aims to treat a set of records that may differ in their inner-structure in a uniform way (e.g. the set of Provenance records may contain records with different inner-structure; still PDS should treat them as a set of equivalent records). To enable that, the inner-structure of each record is defined in an XML schema (i.e. structural RepInfo for each record) kept along with the record’s content. When a record is generated by PDS we use the default schema for the record. These default schemas will also be exposed to the user if wishes to use them.

The following table is a summary of PDS interfaces.

<table>
<thead>
<tr>
<th>Interface Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PDSIntegratedManager</strong></td>
</tr>
<tr>
<td>PDSIntegratedManager interface defines the entry points that should be implemented in an environment in which PDS is integrated into an existing system.</td>
</tr>
</tbody>
</table>

---

Copyright 2007 IBM Corp. All rights Reserved
3.3 SCREEN CAPTURES

The following images show some screen captures taken from the PDS prototype demo. One can access the demo which is openly available at http://ibm.mooo.com:8080/PdsWebClient/PDSDemo/Index.jsp

A recorded demo also exists at [11].

![Preservation DataStores Demo](image)

**Figure 2: IngestAip demonstration**
Figure 3: accessContentDataAsLink demonstration

Figure 4: accessRepInfo demonstration
Figure 5: accessProvenance demonstration

Figure 6: accessFixity demonstration
4 REFERENCES


