The Use of the Producer-Archive Workflow Network (PAWN) in Support of Customized Archival Practice

Mike Smorul, Mike McGann, Joseph JaJa
Institute for Advanced Computer Studies
University of Maryland, College Park

Abstract
It is recognized that the long-term preservation of data is a process that must begin before the data is ingested into an archival system and must of course be continued throughout the lifetime of the data. In many cases, the workflow that data must follow will depend on the particular relationship between the producers and the archive, the type of data and the expectation of the users (consumers of the data); furthermore data may require different destinations within an archival system, or even ingestion into multiple archival systems. In this paper, we describe the ability of the Producer-Archive Workflow Network (PAWN) to support multiple archival workflows while providing a common infrastructure for ingesting data targeted for long term preservation.

1. Introduction

The PAWN environment provides a distributed, scalable platform for creating packages that are published into an archival system. PAWN builds on the previous version [1] by offering a considerably more flexible environment. We adopt the framework developed by the Open Archival Information Systems (OAIS) [2] focusing on the Producer – Archive interactions [3], in which producers prepare and transfer the information to be preserved to an archive, which is responsible for managing the digital information and for providing an interface to the consumers (data users). The PAWN platform consists of several components that provide cross-domain security package storage and integrity checking and a set of API's that allow PAWN to be extended as new ingestion techniques and archival systems become available. To support this extensibility, PAWN provides:

- A flexible environment of roles, and role assignment to support users ranging from producers, record managers, through archive administrators.
- An infrastructure for designing interfaces to allow for custom package creation and processing.
- Definable gateways that publish items into one or more long-term destinations.

These roles and API's have been used in expanding PAWN to handle different types of collections. Ingestion interfaces have been designed to ingest a collection of over 15,000 CD-ROMs, an online collection of children's books, and government records. An archive gateway to the Storage Resource Broker[4] (SRB) has been developed, and development is currently underway on a plug-in to publish packages into Fedora[5].

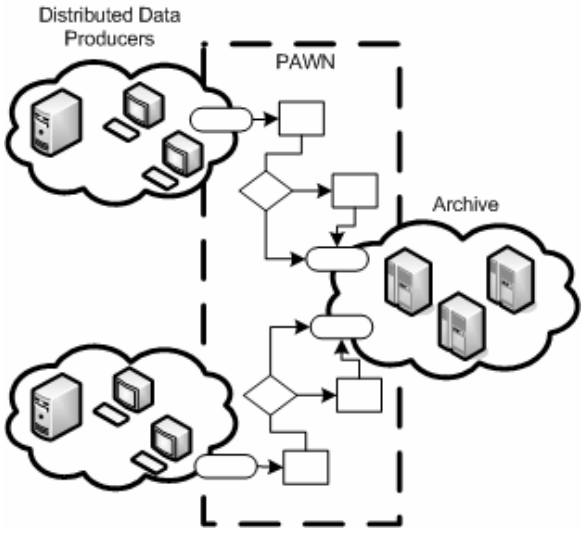

Figure 1: Pawn Architecture

The security framework in PAWN is based on the Security Assertion Markup Language (SAML)[6]. The SAML framework allows PAWN to issue tokens on a per-package basis which specify what actions a client may perform. These tokens allow a client to communicate with many components in PAWN without these components being tightly coupled with a centralized authorization facility. Since these tokens are issued on a per package basis, they not only include actions a user is allowed to perform, but also restrict actions based on the current state of a package.

Various combinations of the operations in PAWN can be grouped together to form a custom role. Operations that can be included in a role range from control over the producer-archive agreement to operations on items in a package. Every account in PAWN is assigned to a role. These roles have been used to model the relationship between the National Archives at College Park and Stanford Linear Accelerator to show how a government agency transfers records to the National Archives.

In addition to roles, multiple producer-archive agreements and different views of that agreement can be stored in PAWN. These agreements allow different users with varying degrees of

authority to have a part in the package workflow. Using roles, each agreement in PAWN specifies a workflow for accepting and publishing data into an archival system.

In the next section, we describe the three major components of PAWN which were considerably extended. In section 3, we discuss two case studies which show how PAWN has been adapted to handle different types of producer-archive interactions.

2. PAWN Technologies

PAWN is a set of distributed services that manage interactions between distributed producers and the archive. Figure 2 shows the interactions between an archive and one management server. As shown in Figure 1, a PAWN installation can include multiple management servers to handle many different producers. Extensible components were designed to allow easy customization of how data enters and exits PAWN while providing a platform for transfer, authorization and authentication.

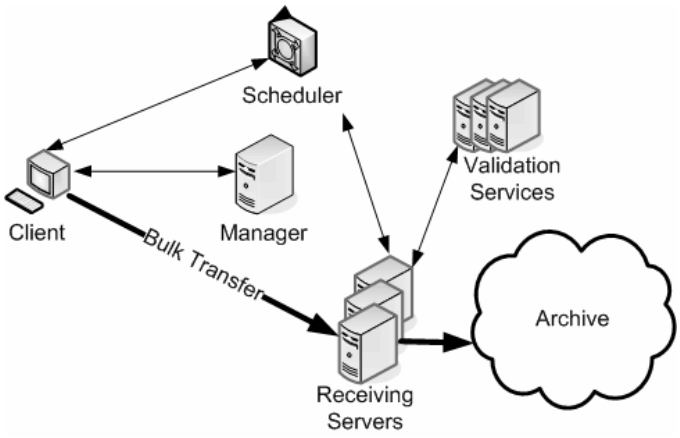

Figure 2: PAWN Components

Customization occurs in several parts of the PAWN network. Role customization is stored on the management server and used by clients when they login to determine their privilege level. The client runs any custom package building software. The receiving servers run any archive gateways what have been defined.

Role Customization

Given that the interactions between the producer and the archive can vary significantly depending on the communities and organizations involved, PAWN provides a flexible environment to enable the customization of the ingestion process to capture a wide variety of possible interactions between the producer and the archive. This is done through the introduction of *roles*. Each account in PAWN is assigned a role. A role is defined by a group of actions allowed, which can range from package management to creating new accounts and defining roles in PAWN. Roles are configurable and can be created as needed depending on the relationship between the producer and archive.

By default, there are four preconfigured roles in PAWN. These are a *global administrator* (GA), *records manager* (RM), *archive manager* (AM), and *producer* (P). The global administrator is able to perform all actions including the creation, modification, and deletion of domains, and setting up manager accounts. The records manager is expected to sit within the data producers administrative structure and is able to create record sets, end-user accounts, and assist in creating and editing packages. The archive manager can assist in managing record organization, edit submitted packages, move items from packages into long term storage, and remove items from PAWN after processing. The last role is an end-user (data producer) who creates and submits packages to PAWN for preservation.

The combinations of core roles allowed for each of the GA, RM, AM, and P are listed in the next table. Elements from the leftmost column can be combined in any way to create other roles as needed.

Action	**P**	**RM**	**AM**	**GA**
Domain creation, modification, deletion				X
Modification of the organizational structure of a domain		X	X	X
Account creation and modification		X		X
Role creation modification				X
Record set creation and modification		X	X	X
Setting permissions on record sets		X		X
Record Schedule creation and modification		X	X	X
Add or delete whole packages	X*	X	X**	X
Modify items in a package	X*	X	X**	X
Limiting an account to working with it's own packages, all packages, or all in a domain.				X
Approving, rejecting, and archiving items in a package		X***	X	X
Lock or unlock entire packages to prevent modification	X****	X	X	X
Configure publishing resources			X	X

* - limited to own packages
** - delete/modify only
*** - approve/reject only
**** - lock only

- P – Producer
- RM – Records Manager
- AM – Archive Manager
- GA – Global Administrator

Figure 3: PAWN Actions and Default Roles

Package Interfaces

The PAWN client provides a workbench for accessing most functionality in PAWN. It is used to configure domains, accounts, record organizational structures and manage packages.

The client provides a mechanism for developing custom package builders. PAWN handles the data transfer and management of higher level record set/schedule functionality. It supplies package builders with a simple package data model that can be used to create new packages or retrieve data from existing packages. In turn, the package builder supplies an interface for package creation and later modification.

The package data model is a set of nested typed manifests. In most instances, a manifest will represent a logical grouping of items, similar to a directory on a file system. Manifests contain an ordered list of data and an ordered list of metadata that is attached to the manifest. Each data item may have a list of attached metadata describing it. There are no format requirements for metadata, which can be treated as an attached file.

Each package builder provides its own namespace, similar to an XML namespace, which identifies which package builder created a given manifest. Within this namespace, the builder is free to create its own set of descriptive types for data, metadata, and manifests. These types can be used to specify uses for various components of a package. For example, a package builder that specializes in organizing collections of books may specify types of manifests to represent pages, chapters, and books, with Dublin core metadata attached at the book level.

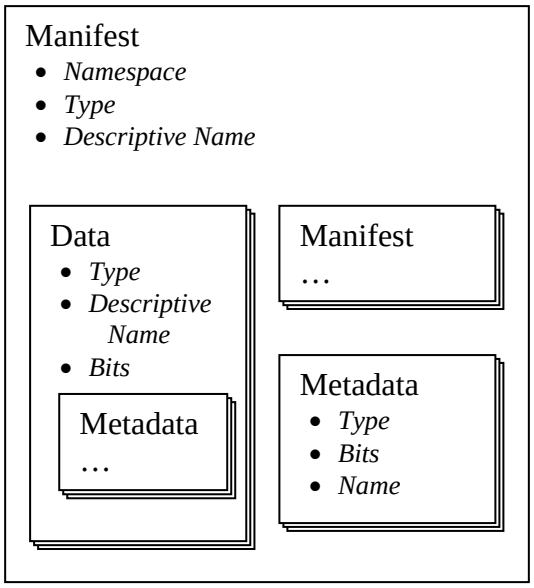

Figure 4: Package builder data model

PAWN ships with a simple package builder that functions similar to a file browser. This builder has a one-to-one mapping between directories and manifests. Local files and directories from a client's computer can be loaded and external metadata files (xml, or other descriptive documents) can be attached to any file or folder.

Archive Interfaces

PAWN allows for any number of gateway services to publish data from PAWN into an archive. An archive interface may take on any number of forms. Several gateways have been developed to ensure the API is flexible enough. The SRB provided a sample distributed storage environment, XFDU[7] was tested to show publishing into alternate package formats, and development for a Fedora gateway is underway.

When data is published from PAWN into an archive, PAWN does not assume responsibility for the published copy. PAWN however, does require each gateway to provide an identifier to the published data. This handle is for logging purposes and any changes to this handle in the archive are not reflected in PAWN.

Archival services are registered on a per-domain basis and are configured using the PAWN scheduler. A domain may have multiple archival gateways. For example, a domain may have gateways that allow data to both flow into an access layer such as a web portal and have a backup copy sent to a tape library or other preservation system.

PAWN stores any necessary driver files and configuration parameters for a resource on the scheduler. Each resource can have multiple configurations that are associated with different domains. Resources may also have custom user interfaces that configure global settings and are used on a per-submission basis.

When an archival resource is used, the following information from PAWN is supplied.

- Global resource configuration specified on a domain by domain basis
- Client parameters specific to the resource (final destination, account information, etc)
- Set of items from an individual package to be archived.

Once a resource has published all selected items from a package, the results are logged in PAWN. The final destination of items in the resource is logged, along with any errors that may have occurred during transport. After publication, items remain in PAWN until someone decides to remove them from PAWN's custody. The remaining archive event logs can be used to track where items may have been published.

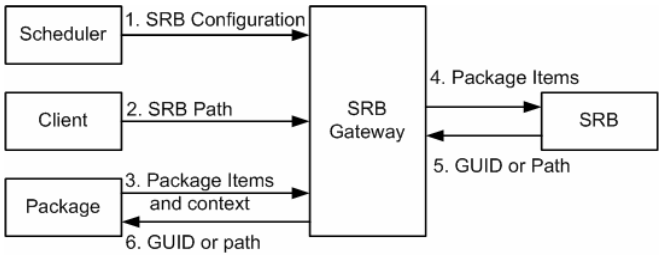

Figure 5: SRB Gateway Interaction

A gateway resource has been developed which allows PAWN to publish into the Storage Resource Broker (SRB)[13] from SDSC. The resource pushes selected data from a package into the SRB and provides the path or guid of an item's final destination. The gateway resource allows default SRB information to be provided through an administrative interface. Examples of this configuration information may be SRB MCAT, storage location, and base directory. The resource also allows the client to choose a final destination of files in the SRB.

3. Case Studies

We will now review few cases where the flexibility of the PAWN approach has been shown. The first consists of a case where a large amount of data needed to be loaded by an untrained operator and the second shows how the roles functionality can model different interactions.

Satellite Imagery

The University of Maryland Libraries has over 15,000 CDROMs each containing one landsat scene. The physical media was hastily rescued and transferred to the UM Libraries, unfortunately leaving the collection largely unorganized. A project was planned to ingest the data into online storage on the other side of campus. A solution was needed to allow an unassisted operator at the library ingest data, while allowing for remote oversight.

The solution was a custom package builder for PAWN. The custom package builder controls up to four CD-ROM drives in a computer. The interface loads and ejects the CD-ROMs, extracts label information and creates a PAWN package. The PAWN infrastructure handles transferring data from the ingestion workstation to a receiving server managed across campus. Additional accounts in PAWN were setup to allow for oversight of the ingestion process.

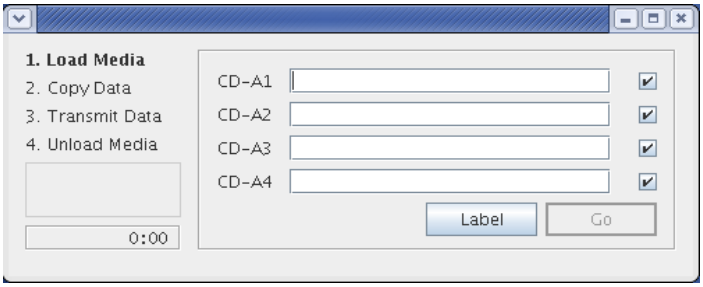

Figure 6: Custom PAWN Client

Stanford Linear Accelerator

Working with the National Archives and Records Administration (NARA) and Stanford Linear Accelerator (SLAC), PAWN was used to model a relationship between a government agency and NARA. The three goals of this test follow:

- Define the roles that various parties require when producers submit data to NARA.
- Test the modeling of NARA record schedules in PAWN.
- Show end to end ingestion of packages from a producer's desktop into the NARA archive testbed.

In looking at the relationship between SLAC and NARA, three different parties were involved in package creation and oversight. These parties were represented as the following roles in PAWN.

- Records Creator – Can create new packages and modify own submissions if they are not locked.
- Records Liaison Officer – Can view and modify any packages in their domain. In addition they can create additional users in their domain and create or modify record sets.
- Records Manager – Can send packages on for more permanent storage, and has the ability to work across domains and modify record schedules. This account used the SRB driver to publish the SLAC data.

In addition to role definition, domain, record scheduler, and record set configuration needed to be performed. A domain for SLAC was created. Within this domain, the' Department of Energy Records Schedule for Research and Development Records' was represented. A record set called 'SLD Experiment Case File' was created that mapped several categories to the disposition authority 'Level I R&D Project Case Files' from the record schedule.

4. Conclusion

We have presented an overview of the extensibility of PAWN. We have described two case studies that demonstrate how PAWN provides an extremely flexible environment to capture a wide variety of possible interactions between distributed producers and an archive. Moreover, the design of PAWN paid from the beginning a particular attention to security, reliability, and scalability using open standards and web technologies.

5. Acknowledgements

This work was supported in part by the National Archives and Records Administration, ERA Program through the National Science Foundation.

6. References

1. PAWN: Producer – Archive Workflow Network in Support of Digital Preservation, M. Smorul, J. JaJa, Y. Wang, and F. McCall, UMIACS Technical Report, UMIACS-TR-2004-49, University of Maryland, College Park, 2004.

2. Reference Model for an Open Archival Information System (OAIS), CCSDS 650.0-B-1, Blue Book, Issue 1, January 2002 [Equivalent to ISO 14721:2002].
3. Producer – Archive Interface Methodology: Abstract Standard, Consultative Committee for Space Data Systems, CCSDS-651.0-R-1, Red Book, December 2002.
4. MySRB & SRB – Components of a Data Grid, A. Rajasekar, M. Wan, and R. Moore, 11th International Symposium on High Performance Distributed Computing, Edinburgh, Scotland, July 24-26, 2002.
5. The Fedora Project: An Open- source Digital Object Repository System, Staples, Thornton, Ross Wayland and Sandra Payette, D-LIb Magazine, April 2003. http://www.dlib.org/dlib/april03/staples/04staples.html
6. Web Services Security (WS-Security) and Security Assertion Markup Language (SAML): http://www.oasis-open.org/specs/
7. XML Formatted Data Unit – XFDU: http://sindbad.gsfc.nasa.gov/xfdu/