Software Requirements Specification

for

DAITSS Version 1.0

draft 18 May 2006

Prepared by P. Caplan

FCLA

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

DAITSS (Dark Archive In The Sunshine State) is a digital preservation repository application. In addition to the repository functions of ingest, data management and dissemination, DAITSS supports the preservation functions of format normalization, mass format migration, and migration on request.

DAITSS is a "dark archive" intended to be used as a back-end to other systems, such as digital library applications or institutional repository software. It has no capability to provide real-time online access to stored content.

1.2 Document Conventions

Names of software routines, such as *ffident*, are italicized throughout. Terms that are defined in the glossary (Appendix A) are italicized the first time they are used.

1.3 Intended Audience and Reading Suggestions

This document is not a technical specification or end-user documentation. It can be used by DAITSS developers and/or DAITSS implementers to help in understanding DAITSS functionality, design and intent.

1.4 Project Scope

DAITSS is a preservation repository application intended for use by committed institutions for the long-term preservation of image, text, audio and video content. It is particularly suited to materials produced in higher education (electronic theses and dissertations), in academic libraries (master files for digitized collections), and in government agencies (documents and series). Because the preservation strategies supported in DAITSS are based on format migration, the application is less well suited to materials requiring emulation such as interactive multimedia and executables. It is also not designed for dynamic content such as databases or harvested websites.

1.5 References

Add references for:
[CRL]
[OAIS]
[METS]
[PREMIS]
[RLG/NARA]

2. Overall Description

2.1 Product Perspective

Digital preservation requires long-term institutional commitment. Application software may carry out certain repository and preservation related tasks, but these must take place within an organizational context that offers stability, fiscal viability, financial sustainability, and procedural accountability. There must be a framework of policies, documentation and transparency of operation sufficient to encourage trust in the repository. Efforts such as the RLG/NARA Digital Repository Certification Task Force [RLG/NARA] and the CRL Certification of Digital Archives Project [CRL] are attempting to define requirements for trusted digital repositories within this larger context.

DAITSS is designed to implement the repository functions specified in the Open Archival Information Systems Reference Model [OAIS]. In OAIS terms, DAITSS supports Ingest, Data Management, Archival Storage, and Access. It also supports the preservation functions of *normalization* and forward *migration*.

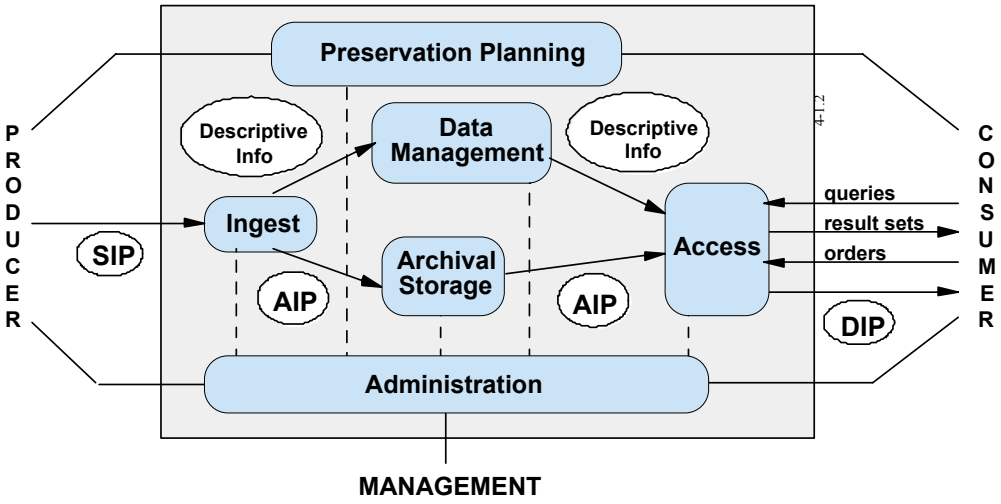

Figure 1: OAIS Functional Entities, from [OAIS] Figure 4-1.

DAITSS assumes that organizations that use the repository (*accounts*) take responsibility for formatting *Submission Information Packages* for Ingest outside of the DAITSS environment. It does not have an interface to support "self archiving" by end users or the creation of SIPs, although there is a Preparation module that can perform certain transformations on batches of submitted SIPs.

DAITSS also assumes that accounts take responsibility for displaying, presenting, or otherwise using content disseminated from the repository. It has no interface for real-time access to or online display of archived content.

2.2 Product Features

The major function of DAITSS is to turn submitted SIPs into stored *Archival Information Packages* (AIPs). The SIP may require some context-specfic pre-processing which is done in the Preparation (Prep) subsystem. The output of Prep is treated as the official SIP.

A SIP must include a SIP Descriptor and one or more files to be archived. The SIP Descriptor is an XML file conforming to the METS schema that references every file to be archived and may contain descriptive, technical, administrative and structural metadata.

DAITSS follows these principles:

- Original source files (that is, files submitted by the account in the official SIP) are maintained without change in perpetuity.
- No files can be modified inside the repository.
- If a change to a file is necessary (for example, a forward migration) a new version of the file is created and managed within the repository.

Much of SIP processing is format-related. DAITSS tries to identify the format of each file in the SIP and to extract or otherwise supply relevant technical and preservation metadata. Depending on the format of the file, DAITSS will create up to three derivative versions during ingest: localized, normalized and migrated. Localized versions are made when a file contains certain types of external references. The referenced files are downloaded and added to the SIP, and the references to them are replaced by relative paths. Normalized versions are created when a file format is considered sub-optimal for preservation and it can be transformed into a preferred format. Migrated versions are created if the file format is obsolete or in danger of becoming obsolete. DAITSS is careful to record relationship information associating files from the same SIP, and relating various versions of the same source file. DAITSS also records significant events related to the files or to the SIP as a whole.

To create the AIP, the files included in the AIP are written to storage along with an AIP Descriptor. All metadata, including the documentation of relationships and events, is recorded in the management database (DADI) and also represented in XML for inclusion in the AIP Descriptor, a METS file describing the AIP. DAITSS implements the concept of "representation" as defined in the PREMIS metadata specification. A representation is the set of all files and metadata needed to render a usable version of an intellectual entity. The original SIP is assumed to contain a single complete representation of an intellectual entity, called the zero representation or R0. The stored AIP will always contain the source files submitted in the original SIP. If any files were migrated on Ingest, the AIP will also contain the latest migrated version, which constitutes a second, current representation (Rc). The AIP Descriptor will contain one structMap for each representation.

DAITSS will disseminate an AIP on request by creating a Dissemination Information Package (DIP). Dissemination has no effect on the AIP (it does not delete or modify it) but the act of dissemination is recorded as an Event. The DIP always contains the original content and will also contain the most current representation if that is different. The DIP also includes a DIP Descriptor, which is actually a copy of the AIP Descriptor.

Because migration is always performed on Ingest, it is possible that a stored AIP may not contain the most current version of any particular file, as the routine to migrate that file format might not have been available at the time the file was ingested. For example, a file of format A may have been ingested in year x. In year x+10 a routine is written to migrate A to B. In year x+15 routines are written to migrate A to C and B to C. The Dissemination subsystem always checks to see if the files in the AIP are the most current version known to the repository, and if a routine exists to migrate any of the files in the DIP, the DIP is automatically routed to Ingest, where it functions as a

Deleted: appropriate

Deleted: I

Deleted: any files in the SIP require forward migration because

Deleted: , a migrated version of the file(s) will be created (assuming a migration routine exists for the format).

Deleted: .¶
¶

Deleted: n

Deleted: contains

Deleted: a METS file with one structmap for each representation.

SIP. Files in the SIP are migrated to their most current versions, and the AIP is re-disseminated with both R0 and Rc.

Files can be deleted from the repository in only two ways. First, entire AIPs can be deleted by the Withdrawal function, which may be initiated by customer request or by repository management. Second, individual files can be deleted by DAITSS in the course of Ingest processing if the file is "disposable". A disposable file must meet three criteria: a) the file was created by DAITSS, b) the file can be regenerated by program, and c) the file has a successor version. For example, a migrated version of an original source file is disposable, and can be deleted from the repository if a more current migrated version is created.

2.3 User Classes and Characteristics

There are only two classes of DAITSS users: repository management, and depositing organizations. Repository management and each depositing organization are represented in DAITSS by *accounts*. (That is, there is one DAITSS account for the repository and one for each depositing organization.) Each account is given an identifying code and descriptive literal string, for example "UNF" = "University of North Florida Library". Every SIP Descriptor must include exactly one account code; that account is considered the owner of the information package.

Each account must have one or more authorized users, or *contacts*. (Note that accounts must be organizations and contacts must be individual persons.) Each contact is represented by a table entry containing contact information (such as phone number and email address) and a set of permissions for actions the contact is allowed to do. Allowable actions are: Deposit materials, Request reports, Request dissemination, Request withdrawal. Contacts associated with depositing organizations can only perform actions on packages owned by the account. Contacts associated with repository management can execute actions on any package in the repository, so long as they are permitted to perform the action at all. That is, if the contact has permission to request withdrawal, then he can request withdrawal on any package in the repository regardless of owner.

Deleted: There is no distinction between the two types of account in practice.¶

Deleted: Sub-accounts may optionally be established under each account by assigning each of these a subaccount code and descriptive literal string.

Deleted: may also have any number or

Deleted: ¶
¶

Deleted:

2.4 Operating Environment

DAITSS is developed under Red Hat Linux running on an IBM Intel processor. Technical installation requirements are given in a separate document, the *DAITSS Installation Guide*.

DAITSS requires a relational database system for the management database (DADI) and is designed to work with MySQL.

Deleted: xxx

Formatted: Font: Italic

Third-party software may be used within DAITSS, particularly for format-specific processing like parsing, validation, normalization and migration. Open source applications are preferred; proprietary commercial products are used only if acceptable functionality is not available in open source software.

2.5 Design and implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

DAITSS SIP Specification
DAITSS Installation Guide

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. System Features: Data Model

4. System Features: General

4.1 OID Server

The OID (Object IDentifier) server assigns unique identifiers to Intellectual Entities, Data Files and Events, when requested by any module. The OID server is designed to handle requests from multiple processes simultaneously.

4.2 Subsystem logs

Subsystems have their own system logs. The log file is created upon initialization of the subsystem, and is named [how?] It will contain [what?] It is the responsibility of repository management to back-up, consolodate, delete and/or otherwise manage these system logs outside of the DAITSS application.

Deleted: Configuration options

Formatted: Bullets and Numbering

Formatted: Normal

Formatted: Bullets and Numbering

4.3 Preservation levels

DAITSS supports three preservation levels: full, bit, and none. Preservation levels are determined at the file level at Ingest based on the account, project code and file format. All XML must be accorded full preservation, and any file created by DAITSS must be accorded full preservation.

Full preservation means that all applicable and available preservation treatments, including forward migration, normalization, and localization, will be applied to the file.

Bit preservation means that the file will be ingested and stored and subject to refreshing and integrity checks, but that derivatives will not be created through migration, normalization or localization.

None means that the file will not be ingested and stored, even if it is part of a SIP being ingested.

4.4 Accounts, Subaccounts and Projects

The SIP Descriptor of every incoming package must contain an account code and a project code, and optionally may contain a subaccount code.

The account code is described in more detail in section 2.3 User Classes and Characteristics, above. It identifies the "owner" of the package and thereby associates the package with the set of contacts who can perform actions such as dissemination or withdrawal. It is also used in reporting and billing.

One or more projects must be established under every account by assigning a project code and descriptive literal string. Projects provide a way for accounts to specify different preservation levels for the same type of file. For example, a depositing organization could specify that for its account, all TIFF files associated with project code "Perm" receive full preservation and all TIFF files with project code "Temp" receive bit-level preservation only. Project codes can also be used in reporting and billing.

One or more subaccounts may optionally be established under any account by assigning a subaccount code and descriptive literal string. Once a subaccount code is established it can optionally be associated with a package by including it in the SIP Descriptor. The subaccount is used in reporting and billing only.

5. System Features: Preparation Subsystem (Prep)

The Prep subsystem preprocesses information packages in order to create good SIPs. The major function of Prep is to validate SIP Descriptors. For each package in the input directory, Prep will check that a SIP descriptor is present. If so, files that are not referenced in the SIP are deleted from the package. If configured to do so, Prep can supply account, subaccount and/or project information to the descriptor.

If there is no existing descriptor, and if configured with account and project codes to use, Prep will create a minimal descriptor for the package, referencing the files that are included.

The existing or created SIP descriptor is validated against the DAITSS SIP Profile and if the descriptor is conformant the package is written to the input directory for the Ingest subsystem. If

Formatted: Bullets and Numbering

Formatted: Normal

Formatted: Bullets and Numbering

Formatted: Normal

Deleted: Configuration options are set in a properties file (daitss.properties). Properties that can be configured include:¶
¶
REPORT_MAIL_BCC .When error reports, ingest reports and other reports are sent to the email account specified for the relevant account, a copy will also be sent to the email account given here. This can be used to provide copies to repository management.¶
¶
¶

Formatted: Bullets and Numbering

Formatted: Normal

the descriptor fails validation the SIP is rejected (that is, it is not copied to Ingest, and an error message is written to the subsystem log.

If special processing is needed for a set of packages, a version of Prep can be written to do this.

6. System Features: Ingest Subsystem

The Ingest Subsystem creates AIPs from SIPs. For each SIP it receives, Ingest performs quality assurance, creates derivative files when needed, generates an AIP Descriptor, and calls upon Storage Management to write the AIP to storage. Ingest also creates or obtains the metadata necessary for preservation and includes this in both the AIP Descriptor and in DADI, the repository management database.

Ingest processes both original SIPs and archive SIPs. Original SIPs are submitted by an account and have never been in the repository before – they contain no DAITSS-assigned identifiers. Archive SIPs contain content that has been disseminated from the repository in order to be re-ingested. The most common reason to re-ingest content is to cause a format migration of one or more files.

Deleted: holder

6.1 Ingest

6.1.1 Process each SIP submitted for Ingest.

Make a list of all SIPs in the input directory. Each SIP must be contained within a single subdirectory directly under the Ingest directory, and the name of the SIP descriptor must be the same as that of the directory. That is, if the SIP is in directory FOO, the SIP descriptor must be named FOO.xml.

For each SIP, find the SIP descriptor and validate it against each of the referenced schema. If invalid, reject the SIP and write an Error Report to the submitting account. "Reject the SIP" means, copy the SIP to the rejects directory, discontinue processing of that SIP, start processing the next SIP (if any).

Deleted: METS

If the SIP descriptor contains an Intellectual Entity Identifier (IEID) from the same installation of DAITSS, then the SIP is an archive SIP. Otherwise it is an original SIP.

Deleted: a re-ingest

If the SIP is original, assign an IEID to it, and create the metadata that pertains to the Intellectual Entity.

Process each file in the SIP as described below. Note that the original SIP Descriptor is treated as a file and undergoes all processing described below, but that archive SIP Descriptors are not treated as files and do not undergo this processing.

Deleted: noted

6.1.2 Process each file submitted in the SIP.

Check the file for viruses. If found, reject the SIP and write an Error Report to the submitting account. Otherwise, record the successful check as an Event.

Calculate SHA-1 and MD5 message digests for the file and compare it to the message digest(s) provided in the SIP descriptor (if any). If different, reject the SIP and write an Error Report to the submitting account. Otherwise,

Deleted: the

a) record the successful check as an Event;

b) look up to see if the message digest [either or both?] is already in the database; if so, the file is a re-ingest, otherwise it is an original ingest.

If the file is original, assign a Data File Identifier (DFID) to it.

Determine the file format by comparing key file characteristics against the characteristics of supported file formats, one after the other, until a match is found. Try the most likely file format(s) first based on a) the MIME type given for the file in the SIP descriptor (if any), b) the format identified by the file type extension (if any), c) the *ffident* function. Within (b), use a priority list of formats from the most specific to the most general, so that, for example, XML is tested for before text. If no match is found among supported formats, treat the format returned by *ffident* as a tentative format. If the format is not identified by *ffident*, treat the file as an unknown format.

Determine the preservation level desired by the account based on the file format and the project code submitted in the metadata. Unknown file formats get bit-level preservation and are assigned the MIME type "application/octet-stream".

If the file format is known and is supported in DAITSS, validate the file against file format specifications. Note any differences as anomalies. If certain severe anomalies are found (those flagged as such in the SEVERE_ELEMENT table in DADI) and the preservation level is full, downgrade the preservation level to bit and write an Event record to document the downgrade. If the file has too many anomalies (according to a threshhold set in the configuration file), do the same.

Deleted: not un

Deleted: listed

Deleted: d

Deleted: are found

Ascertain the file size. Parse the file to extract other appropriate technical metadata for the file format at the file and bitstream levels. Some elements of technical metadata may have been supplied in the SIP descriptor: MIME type, file size, create date, creator program, image compression name, image depth, sampling frequency on images, image dimension(X,Y), image orientation and image color space. Compare supplied values with determined values and note any conflicts (except in create date) as warnings in the Ingest Report. Where supplied and determined values differ, use determined values. Where values are supplied but can not be determined through parsing, use the supplied values. Exception: if the file format is unknown, use "application/octet-stream" as the MIME type even if a different MIME type value is supplied. Add metadata to DADI.

Deleted: there a confict between

Record the relationship between the file and the IE and add to DADI.

6.1.3 Normalize files that require normalization.

Formatted: Indent: Left: 36 pt

Deleted: ¶

Review each file in the SIP to determine if it requires normalization, based upon whether the file is designated for full preservation and whether there is a normalization routine for files of that format. If so, and the file is original, create a normalized version. If the file is a re-ingest, check to see if the file has already been normalized by the current normalization routine for that format and if not, create a normalized version. In either case do not save the normalized version, but record the successful normalization as an Event. [yes?] If the normalization was unsuccessful, write an Error Report to the submitting account and reject the SIP.

Formatted: Indent: Left: 72 pt

6.1.4 Migrate files that require migration.

Formatted: Indent: Left: 36 pt

Review each file in the SIP to determine if it requires format migration, based upon whether the file is designated for full preservation and whether there is a forward migration routine for files of that format. If so, and the file is original, create a migrated version. If the file is a re-ingest, check to see if the file has already been migrated by the current migration routine for that format and if not, create a migrated version. Treat the migrated version as a new file to be added to the AIP: create

appropriate technical metadata for the migrated version and add to DADI. Record the preservation level as full. Record the relationships between the migrated file(s) and the source file and add to DADI.

6.1.5 Localize files that require localization.

Review each file in the SIP, including any file(s) created via migration, to determine if it requires localization. This is done if the file is designated for full preservation and if it contains one or more *critical links* (a link to a file necessary for validation). If so, and the linked-to file is not included in the SIP, attempt to download the linked-to file. If the download is unsuccessful [what?] Otherwise determine if the downloaded file is a *global file*: calculate [a SHA-1?] checksum on the file and look it up in a list of checksums of global files. If it is a global file, use the global file and all of its descendents from cache in place of the newly downloaded file. Replace the link with a reference to the relative file path of the newly downloaded or global file. Treat the localized version and any downloaded or global files as new files to be added to the AIP: create appropriate technical metadata at the file and bitstream levels and add to DADI. Record the relationship between the localized file and the source file and add to DADI.

6.1.6 Create an AIP descriptor

The AIP must include one and only one AIP Descriptor, which is created by Ingest. (For a re-ingest, the existing AIP Descriptor is discarded in favor of a newly created AIP Descriptor.) The AIP Descriptor describes the original representation and current representation of the Intellectual Entity.

The AIP Descriptor contains one <filegrp> that references each file included in the AIP (see Create an AIP below). It also contains an XML rendering of all the metadata added to DADI for the Intellectual Entity and all included files and bitstreams, including Events and Relationships.

The AIP Descriptor also includes all pertinent information supplied in the SIP Descriptor. That means that the AIP Descriptor includes:
a) the Account, Subaccount (if any) and Project values from the SIP Descriptor,
b) all descriptive, administrative or other metadata referenced via an ID or IDREF mechanism in the SIP Descriptor,
c) the first structure map from the SIP Descriptor. If the SIP is original, add a reference to the original SIP Descriptor itself, which is now included in the AIP as a Data File.

If any files were migrated or localized on Ingest, include a second structure map in the AIP Descriptor to describe the current representation. This structure map is identical to the first structure map with the exception that the preferred versions of the files are referenced instead of the files from which they were derived. Localized migrated versions are preferred, followed by localized and migrated versions. If the preferred version is a distributed file, the root of the distributed file is referenced in place of the original file and the child files are referenced from s nested under the root.

6.1.4 Create the AIP

The AIP consists of the AIP Descriptor, the original SIP Descriptor, any supplied global files, and all files included in the SIP or created by Ingest that meet these conditions:

a) the file preservation level is "full" or "bit" (that is, not "none"),

Formatted: Indent: Left: 36 pt

Formatted: Font: Italic

Formatted: Font: Italic

Deleted: If the file is XML and [if it is designated for full preservation?] and if it references a file that is not included in the SIP, check whether the linked-to-file is a "global" file (a commonly used file known to the repository) and if so, obtain a copy of the global file.¶
If the file is designated for full preservation, determine if files of that format require localization. If yes, and if the file is original and if it contains links to other files, create create a "localized" version of the file. [define]

Deleted: a

Deleted: If the file is designated for full preservation, determine if files of that format require normalization. If yes, and if the file is original, create a normalized version. If yes and the file is a re-ingest, check to see if the file has already been normalized by the current normalization routine for that format and if not, create a normalized version. In either case do not save the normalized version, but write a message to the repository if normalization is unsuccessful or if any problems are encountered.¶
If the file is designated for full preservation, determine if files of that format require migration. If yes, if the file is original, create a migrated version. If yes and the file is a re-ingest, check to see if the file has already been migrated by the current migration routine for that format and if not, create a migrated version. Treat the migrated version as a new file to be added to the AIP: create appropriate technical metadata for the migrated version and add to DADI. Record the preservation level as full. Record the relationships between t ... [1]

Deleted: 3

Deleted: se

Deleted: c

Deleted: ef

Deleted: Descriptor

Deleted: the structure map is modified to include

Deleted: content

Deleted: d

Deleted: f

Deleted: i

Deleted: is included i

Deleted: migrated

Deleted: result of a migration

b) the file is not disposable.

Deleted: already in the repository (that is, the checksum of the file is not found in DADI)

The AIP is written to the output directory, and Storage Management is called to write n copies to repository storage, where "n" is a configuration option. If the writing is successful,
a) commit updates to DADI for this AIP,
b) delete the SIP from the input directory,
c) record the successful creation of the AIP as an Ingest Event,
d) create an XML Ingest Report and email it to the owning account.

Deleted: .

If the writing is unsuccessful, log the error in the log file and reject the SIP.

7. System Features: Access Subsystem

7.1 Access Control

Authentication of DAITSS users by userid and password is a system function performed by the operating system. Access control

???? will we use Unix or DAITSS to authenticate contacts?
???? will Access Control handle digital signatures, returning true if the request or package was sent by the purported sender and was unaltered in transmission?
??? Access control will provide the level of authentication that says "this user can issue this type of request (e.g. a dissemination request)" Will Access Control also provide the level "this user can request dissemination for package Y" ???

7.2 Web Interface

The DAITSS Web Interface allows authorized contacts to create requests and view report results via a Web form. Types of requests are:

- type 1: dissemination or withdrawal
- type 2: web report
- type 3: email report

[Assume reporting is a Data Management function, so requests for reports go to Data Management.]

7.2.1 Allow online creation of type 1 requests

Summary: Allow an authorized user to create a dissemination or withdrawal request through a web form. Validate and accept the request or return error messages to the user.

The Access Control function is used to authenticate the user and verify that the user is authorized to create the request.

The dissemination request form requires the requesting individual (provided by the system as the identity is known to Access Control) and the IEID of at least one AIP in the repository to be specified. An arbitrary number of IEIDs can be specified in a single request.

For each IEID specified, verify that:
a) the IEID identifies an AIP in the repository;
b) the requestor is authorized to request dissemination or withdrawal for the institution owning the AIP.

If (a) and/or (b) fail for any IEID, return a message in real time on the online request form. If (a) and (b) pass for all IEIDs, create one request for each IEID specified, and return a message in real time on the online request form that the request has been accepted.

Requests are put in the appropriate request queue [or whatever]. Each request must contain:
a) the requesting individual
b) the IEID identifying the package to be disseminated or withdrawn
c) the date and time of the request.

7.2.2 Allow online creation of type 2 requests and display results

Summary: Allow an authorized user to create a request for certain reports through a web form. Validate the request and display the report, or return error messages to the user.

The Access Control function is used to authenticate the user and verify that the user is authorized to create the request.

[needs designing]

7.2.3 Allow online creation of type 3 requests

Summary: Allow an authorized user to create a request for certain reports through a web form. Validate and accept the request, or return error messages to the user.

The Access Control function is used to authenticate the user and verify that the user is authorized to create the request.

[needs designing]

7.3 Email Interface

The DAITSS Email Interface allows authorized contacts to create requests via email. Types of requests are:

- type 1: dissemination or withdrawal

- type 3: email report

7.2.1 Allow email creation of type 1 requests

Summary: Allow an authorized user to create a dissemination or withdrawal request via email. Validate and accept the request or return an Error Report to the user.

The email must have "dissemination request" or "withdrawal request" as the SUBJECT line. The FROM address must be an email address associated with a DAITSS contact. The body of the message must contain one or more IEIDs separated by CR, and no other content. An arbitrary number of IEIDs can be specified in a single request.

The email request must be accompanied by a digital signature signed by the private key of the sender. The Access Control function is used to authenticate the sender and verify that the email was not changed since the time of its creation. It also verifies that the sender is authorized to create the request.

For each IEID specified, verify that:
a) the IEID identifies an AIP in the repository;
b) the requestor is authorized to request dissemination or withdrawal for the institution owning the AIP.

If (a) and/or (b) fail for any IEID, create an Error Report and email it to [the sender? or the report address?]. If (a) and (b) pass for all IEIDs, create one request for each IEID specified, and return an email message that the request has been accepted. [this would be a new type of email report – an acceptance report?]

Requests are put in the appropriate request queue [or whatever]. Each request must contain:
a) the requesting individual
b) the IEID identifying the package to be disseminated or withdrawn
c) the date and time of the request.

7.4 Command Line Interface

7.5 Dissemination

Dissemination is a DAITSS service in the Access Subsystem. Its main function is to create a Dissemination Information Package (DIP) from the objects stored in an AIP (Archival Information Package). Dissemination retrieves the appropriate metadata and data files from the AIP and creates a DIP consisting of

a) the originally submitted representation (R0) consisting of data files and the SIP descriptor,
b) the current ("last best") representation (Rc) if different from R0, and
c) a DIP descriptor, which is identical to the current AIP descriptor.

The DIP may be routed to the content's owner, or may be re-Ingested into the repository in order to effect a format migration.

7.5.1 Accept dissemination requests

Summary: Accept a dissemination request from the request queue. The requests may have been created by the Web Interface or the Email Interface.

Requests have already been authenticated and validated, and contain
a) the requesting individual
b) the IEID identifying the package to be disseminated
c) the date and time of the request.

7.4.2 Disseminate the requested material

Summary: Assemble a DIP for the requested IEID consisting of data files and a DIP descriptor. Make the DIP available for the requesting institution to pick up via FTP, and notify the institution to do so.

Obtain copies of the data files to be disseminated from the Storage Manager. The data files are those constituting a) the originally submitted representation (R0) consisting of data files and the SIP descriptor, b) the current ("last best") representation (Rc), if different from R0, and c) a DIP descriptor, which is identical to the current AIP descriptor.The Dissemination function can assume that the only files stored in the AIP belong to either R0 or Rc or both, and that the stored AIP descriptor accurately describes each representation. Therefore Dissemination can simply copy each of the data files associated with the IEID, including the stored AIP descriptor.

Determine whether the DIP needs to be re-ingested in order to perform format migration. If there are any unknown files, do an ffident on the unknown files; if any are identified, the DIP will be re-ingested. Otherwise check if any of the files in the DIP require migration (if there is a migration routine for the format, and the preservation level of the file is "full"); if yes, the DIP will be re-ingested.

Zip up the files and create a message digest for the zipped file.

Record the act of dissemination as an Event. Include the requestor and the message digest of the zipped DIP in the outcome detail.

Place the zipped file in the dissemination directory associated with the requestor. (FDA note: For customer requests, the directory will be their FTP directory; for repository management the directory will be the reservoir.)

Send an email to the appropriate contact address informing the requesting institution that the DIP is available for pick-up within the next seven business days. The email must include the requestor and the message digest of the zipped DIP.

8. System Features: Data Management

8.1 Withdrawal

Withdrawal removes an AIP from the repository. All copies of all stored files in the AIP are deleted, and all metadata pertaining to the files are deleted. Metadata for the Intellectual Entity remains, linked to metadata for the Withdrawal event.

8.1.1 Allow online creation and validation of withdrawal requests

Summary: Allow an authorized user to create a withdrawal request through a web form. Validate and accept the request or return error messages to the user.

The Access Control function is used to authenticate the user and verify that the user is authorized to create a withdrawal request.

Note: The withdrawal request form should be integrated with reporting, so that the user could get a report of (for example) all PDFs deposited in 2006 and transfer that information directly to the withdrawal request form.

The withdrawal request form requires the requesting individual (provided by the system as the identity is known to Access Control) and the IEID (Intellectual Entitity Identifier) of at least one AIP in the repository to be specified. An arbitrary number of IEIDs can be specified in a single request.

For each IEID specified, verify that:
a) the IEID identifies an AIP in the repository;
b) the requestor is authorized to request withdrawal for the institution owning the AIP.

If (a) and/or (b) fail for any IEID, return a message in real time on the online request form. If (a) and (b) pass for all IEIDs, create one request for each IEID specified, and return a message in real time on the online request form that the request has been accepted.

Each output request must contain
a) the requesting individual
b) the IEID identifying the package to be withdrawn
c) the date and time of the request.

8.1.2 Remove metadata and content for the AIP from the repository.

For each Data File included in the AIP, save the information pertaining to how to access the data file in storage, then delete all metadata from the management database (DADI) pertaining to the Data File. (Do not remove metadata pertaining to the Intellectual Entity.)

Record the withdrawal of the package as an Event associated with the Intellectual Entity. Include the requesting individual and the date and time of the Withdrawal in the Event record. Once this Event is recorded, the AIP is considered successfully withdrawn, even if subsequent processing fails.

Using the saved access information, remove all copies of each Data File from storage. If deletion of any Data File fails, write a log message and continue with subsequent Data Files. Any residual content can be removed manually by repository management.

9. External Interface Requirements

9.1

9.2 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

9.3 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

9.4 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

9.5 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

10. Other Nonfunctional Requirements

10.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

10.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

10.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

10.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

11. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

Account: A coded value in DAITSS representing repository management or an external entity having an agreement with repository management to use the repository. Each account is assigned an account code which must accompany all SIPs submitted by the account in order to associate content with the responsible organization.

AIP: Archival Information Package.

AIP Descriptor: A Data File conforming to the METS schema which contains all of the metadata describing an AIP.

*Archival Information Package: "*An Information Package, consisting of the Conent Information and the associated Preservation Description Information (PDI), which is preserved with an OAIS." [OAIS]. In DAITSS, the AIP consists of stored Data Files and the stored AIP Descriptor.

Archive SIP: A DIP disseminated from DAITSS used as a SIP for re-ingestion; a SIP created by DAITSS.

Bitstream: Contiguous or non-contiguous data within a file that has meaningful common properties for preservation purposes. [PREMIS]

Contact: An individual associated with a DAITSS account who is authorized to do certain things such as request dissemination.

DADI: Digital Archive Database Instance, the relational database used by the repository system for data management purposes.

Formatted: Indent: Left: 0 pt, Hanging: 72 pt

Formatted: Font: Not Italic

Data File: A file archived by DAITSS.

Descriptor: A file conforming to the METS schema which contains all of the metadata describing a SIP, AIP or DIP.

Disposable file: A Data File that can be deleted from the repository. A disposable file must meet three criteria: a) the file was created by DAITSS, b) the file can be regenerated by program, and c) the file has a successor version. For example, a migrated version of an original source file is disposable, and can be deleted from the repository if a more current migrated version is created.

Dissemination Information Package: "The Information Package, derived from one or more AIPs, received by the Consumer in response to a request to the OAIS." [OAIS] In DAITSS, the DIP is always derived from a single AIP, and contains both the originally submitted representation and the most recently migrated representation (if different) of a single Intellectual Entity.

Distributed file: A file containing links to other files.

DIP: Dissemination Information Package.

Event: Action that involves at least one Digital Object and/or Agent known to the Preservation Repository. [PREMIS] In DAITSS, important events are recorded in DADI and in the AIP Descriptor.

File: Named and ordered sequence of Bytes that is known by an operating system. [PREMIS]

Global file: A file which is has been pre-loaded into cache along with all of the files it references. A file is designated as a global file by repository management if it is frequently externally referenced in SIPs. The METS schema, for example, ought to be treated as a global file.).

Formatted: Indent: Left: 0 pt, Hanging: 72 pt

Formatted: Font: Not Italic

Information package: The Content Information and associated Preservation Description information which is needed to aid in the preservation of the Content Information. [OAIS] A SIP, AIP or DIP.

Intellectual Entity: A coherent set of content that is described and used as a unit, for example, a book, a map, a photograph, an issue of a serial.

Localization: A preservation strategy applicable to files that contain external references (links to external files); the referenced files are downloaded to the local file system if they are not already contained in the SIP, and the external references are replaced with relative file paths.

Localized version: In DAITSS, a new Data File created by localization of an existing Data File.

Migrated version: In DAITSS, a new Data File created by migration of an existing Data File.

Migration: A preservation strategy in which a Transformation creates a version of a Digital Object a different Format, where the new Format is compatible with contemporary software and harware environments. Also called "format migration" and "forward migration." [PREMIS]

Normalization: Form of Migration in which a version of a Digital Object is created in a new Format with properties more conducive to preservation treatment. [PREMIS]

Normalized version: In DAITSS, a new file created by normalization of a file contained in a SIP. The normalized version may or may not be stored as a Data File in the repository.

Package: See *Information Package*.

Representation: The set of stored Files and structural metadata needed to provide a complete and reasonable version of an Intellectual Entity. [PREMIS]

Root file:

SIP: Submission Information Package.

Submission Information Package: An Information Package that is delivered by the Producer to the OAIS for use in the construction of one or more AIPs. [OAIS] In DAITSS, one SIP always equates to one AIP.

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>

If the file is designated for full preservation, determine if files of that format require normalization. If yes, and if the file is original, create a normalized version. If yes and the file is a re-ingest, check to see if the file has already been normalized by the current normalization routine for that format and if not, create a normalized version. In either case do not save the normalized version, but write a message to the repository if normalization is unsuccessful or if any problems are encountered.

If the file is designated for full preservation, determine if files of that format require migration. If yes, if the file is original, create a migrated version. If yes and the file is a re-ingest, check to see if the file has already been migrated by the current migration routine for that format and if not, create a migrated version. Treat the migrated version as a new file to be added to the AIP: create appropriate technical metadata for the migrated version and add to DADI. Record the preservation level as full. Record the relationships between the migrated file(s) and the source file and add to DADI.

Record the relationship between the file and the IE and add to DADI.