Chapter 1: DAITSS Overview

Topics covered in this chapter:

DAITSS 2

DAITSS is a digital preservation software application developed by the Florida Center for Library Automation (FCLA) with some support from the IMLS. DAITSS is used by the Florida Digital Archive (FDA), a long-term preservation repository service provided by FCLA for the use of the libraries of the eleven publicly-funded universities in Florida. The FDA has used DAITSS in production since 2006; a second, completely rewritten version (DAITSS 2) went into production in December, 2010.

The FDA is a "dark archive", which here means that it offers no online or public access. DAITSS allows depositors to get their own materials back on request, but not to access the materials of other depositors. Non-depositors have no access at all. If an institution wanted to provide a "light archive", it would have to couple DAITSS with another application designed to provide end user access.

Similarly, DAITSS provides no support for assembling information packages suitable for submission to the archive. DAITSS' functionality begins at the point where a potential depositor submits the package. To support author self-archiving or package creation as part of various institutional workflows, other applications would have to be provided.

DAITSS Functions

DAITSS provides automated support for Submission, Ingest, Archival Storage, Access, Withdrawal, and Repository Management.

Submission

In Submission, a depositor provides content to DAITSS, which DAITSS either accepts for Ingest or rejects. Content is provided in the form of a Submission Information Package (SIP) that must contain one or more files to be archived and a METS document (that is, an XML file created according to the METS specification) that provides certain information about the package. DAITSS will check that the submitter is authorized to deposit content in the archive and that the content is correctly identified as belonging to the submitter. It will also validate the contents of the package, checking that the METS document is properly formatted and contains the right information, and that the right files have been included for archiving. If the SIP passes validation, the depositor gets a message that includes the system identifier assigned to the package. If it fails validation, the submitter gets a rejection message explaining the problem.

Ingest

Ingest is defined in OAIS as the process of transforming a Submission Information Package into an Archival Information Package (AIP). In other words, Ingest takes what a depositor submits and transforms it into something suitable for long-term preservation. Ingest is a rather involved process with some steps that act upon the package as a whole, and other steps that focus on individual files.

First, every file in the SIP is checked for viruses. (If a virus is found, the entire package is copied into a holding area where it will remain until a repository operator rejects it or releases it for further processing.) Each file is then analyzed by DAITSS in order to determine its file format, validated against file format specifications, and described in technical terms. Depending on the file format and how preservation-worthy it is considered to be, one or more new versions of the file can be made on the spot. The file as originally submitted remains unaltered, and is related to the new versions by metadata. XML files undergo additional special handling to ensure that all of the schema required to validate them are present; missing schema are downloaded if possible, combined into a tarball and added to the package.

When every file has been completely processed, Ingest creates a new METS document called an "AIP Descriptor" to describe the new package. This includes technical descriptions of all the files, "event" information that describes all processing steps and transformations that occurred, and general, structural information about the package as a whole. With the addition of the completed AIP Descriptor, the AIP is complete and ready to be bundled up as a TAR file and handed off to Archival Storage. Before it goes, however, information useful for

future processing, reporting, selection, and other operational functions is copied from the AIP Descriptor and added to a fast-access relational database.

Archival Storage

What storage media to use, how many copies of each package to store, and where stored copies are kept are decisions entirely up to the institution running DAITSS. The institution must set up one or more storage silos, which are chunks of storage available for use, and then assign each silo to a storage pool, which is a designated set of similar silos. For example, the FDA has defined two storage pools in different cities, each pool containing dozens of silos. The silos in one pool are defined as 2TB chunks of disk and the silos in the other pool are 2TB chunks of tape. (Note that non-disk silos present certain challenges the institution should be aware of.)

Ingest delivers the AIP package as a single TAR file to the Storage function. DAITSS will write a copy of the package to each defined storage pool. Within each pool, DAITSS will identify all the available silos that have enough room to store the package, and select from these the silo with the least free space. This ensures that silos are filled as full as possible with the least waste of space. Each time a copy of the package is stored its checksum is calculated and compared to a checksum previously calculated by Ingest, to ensure no errors affecting the bitstream occurred. Information about the location of the copies of the package and the timestamp of the latest checksum calculation are stored in databases to be used in later fixity checking.

Fixity checking is DAITSS' primary means of ensuring the integrity of stored AIPs. As a general technique, a fixity check involves calculating a checksum over a file and comparing it with a previously calculated checksum; if the two strings are identical, the file has not been altered in the period of time between the two calculations. DAITSS continuously goes through its storage silos calculating and storing checksums on the stored packages. For a fixity check on an AIP to be considered successful, the checksums of all stored copies must be found to match correctly. Information about the latest successful fixity check on each AIP is recorded. DAITSS operators are alerted to an unsuccessful fixity check on any package.

Withdrawal

Withdrawal of an archived package is initiated by submitting a request through a Web interface. If the requester is authorized to withdraw that particular AIP, the request is forwarded to a repository operator to confirm. All copies of the AIP stored as TAR files are deleted, and most of the information about the AIP is removed from the system. However, the information that the package once existed, basic facts about the package, and facts about the withdrawal request (who made the request and when) is permanently maintained by the system.

Access

As noted above, DAITSS does not support online public access. The only way to access a stored AIP is by issuing a dissemination request. Dissemination does not withdraw the package from the archive, it just provides a modified copy of the package in a location that the requester can access.

DAITSS will check that the requester is authorized to disseminate the AIP, and if so, DAITSS will produce a Dissemination Information Package (DIP). Since DAITSS is continually upgraded with new and improved format recognition and processing routines, DAITSS "refreshes" the AIP – that is, it repeats the Ingest processing steps on each file -- before generating the DIP. That ensures that the DIP contains the best possible versions of all files, no matter when the SIP was originally ingested.

The DIP will contain a METS document describing the package (the "DIP Descriptor"), all of the content files in the original submission, and files added by DAITSS itself, including the tarball of schema downloaded by XML Resolution. The DIP Descriptor contains information showing the relationship of different versions of files to each other. It also identifies the set of all files in the original submission, and if different, the set of files constituting the latest, best version. For complex objects, this can be less than straightforward. For example, imagine submitting a package of 5 files comprising a single Web page: an HTML file, a stylesheet, a couple of photographs in JPEG format, and an icon in GIF format. Now imagine that years later GIF becomes obsolete, and DAITSS creates a version of the icon in still-usable PNG format. The DIP will contain 6 content files and the DIP Descriptor will identify two sets of files: first, the set of the original 5 files, and second, the set of the currently usable files, which would be the HTML file, the stylesheet, the 2 JPEGs and the new PNG. If the requestor wants to examine his originally submitted files he can look at the first set, and if he wants to recreate the Web page, he can use the second set.

A second form of access, tentatively called "Peek", is planned for future DAITSS development. Peek will let an authorized user take a look at an AIP as it exists in Storage, without being refreshed or transformed into a DIP. Peek will be especially useful to auditors who need to verify that a repository holds what it purports to hold in exactly the way it purports to hold it.

DAITSS operators have access to information about all packages stored or in process in the system. Authorized depositors to the repository can access information about their own packages via a Web interface. Users can see the status of any of their own submissions, dissemination requests, and/or withdrawal requests that are still in process, and can query summary information about stored packages. They also have access to both ad hoc and pre-configured reports.

Repository management

Repository staff have special privileges and functions in DAITSS, such as setting up accounts, confirming withdrawals, and managing packages that fail some step of processing. DAITSS operators must decide whether to reject the error package, release it to the next processing

step, or take some remedial action. For example, if the virus checker found a problem in a file, an operator could determine it was a false positive and send the package on its way. If a package raised some program error, the operator may have to reject the package so that it can be corrected and resubmitted.

DAITSS provides a "dashboard" so operators can see an overview of operations at any given time, and some GUI interfaces for managing system tables and user accounts. As time goes on these will be improved and enhanced based on operator suggestions.

DAITSS Preservation strategies

DAITSS implements active preservation strategies based on format transformation: normalization and forward migration. *Normalized* versions are intended to be more preservation-worthy, even though they may lose some of the original content; for example, a proprietary spreadsheet might be normalized into an open, XML-based format. *Migrated* versions are more up-to-date versions of the format (e.g. Excel 2007 to Excel 2010) or versions in successor formats (e.g. MP3 to MusicDNA).

Transformation-based preservation is obviously sensitive to file formats. Currently DAITSS can identify over 600 file formats and fully supports (that is, can analyze, describe, and transform as needed) about a dozen commonly used formats. To added support for a format, a format expert must study the format, decide how to treat it, and find or code programs to carry out that treatment. Intellectually this can be difficult and require significant format expertise, but once the treatment is decided, the programming required to implement the action plan in DAITSS is relatively simple.

Transformation-based strategies are well-suited to text, image, audio and video formats. Some other formats, such as computer executables, interactive multimedia, games, and databases are better candidates for emulation-based strategies. DAITSS can provide a secure environment for bit preservation of these formats, but cannot ensure that they remain usable.

The DAITSS approach is to keep up to three versions of any given file. The version as originally submitted is kept unaltered indefinitely. Depending on the format, one normalized version and/or one migrated version may also be stored. This means that if a particular file format is has many successor versions only the latest migration will be retained.

Derivative versions are created by the DAITSS Ingest function for new AIPs, and by the Refresh function for existing AIPs. Refresh is performed as a matter of course as part of Dissemination, ensuring that all disseminated packages contain the best possible content. There has been some discussion in the preservation community over the relative merits of mass migration (doing one-time forward migration on all files of a particular format held in a repository) versus migration on request (performing forward migration on individual files at the point they are used). This is essentially a debate about just-in-case v. just-in-time. DAITSS supports both approaches. File migration will always be performed just-in-time, since Refresh is part of the Dissemination process. If a repository wants to do a mass migration, it

needs only use the reporting function to identify the affected packages and then initiate a request for all of them to be refreshed.

Standards Conformance

DAITSS adheres to the major standards in use in the international preservation community:

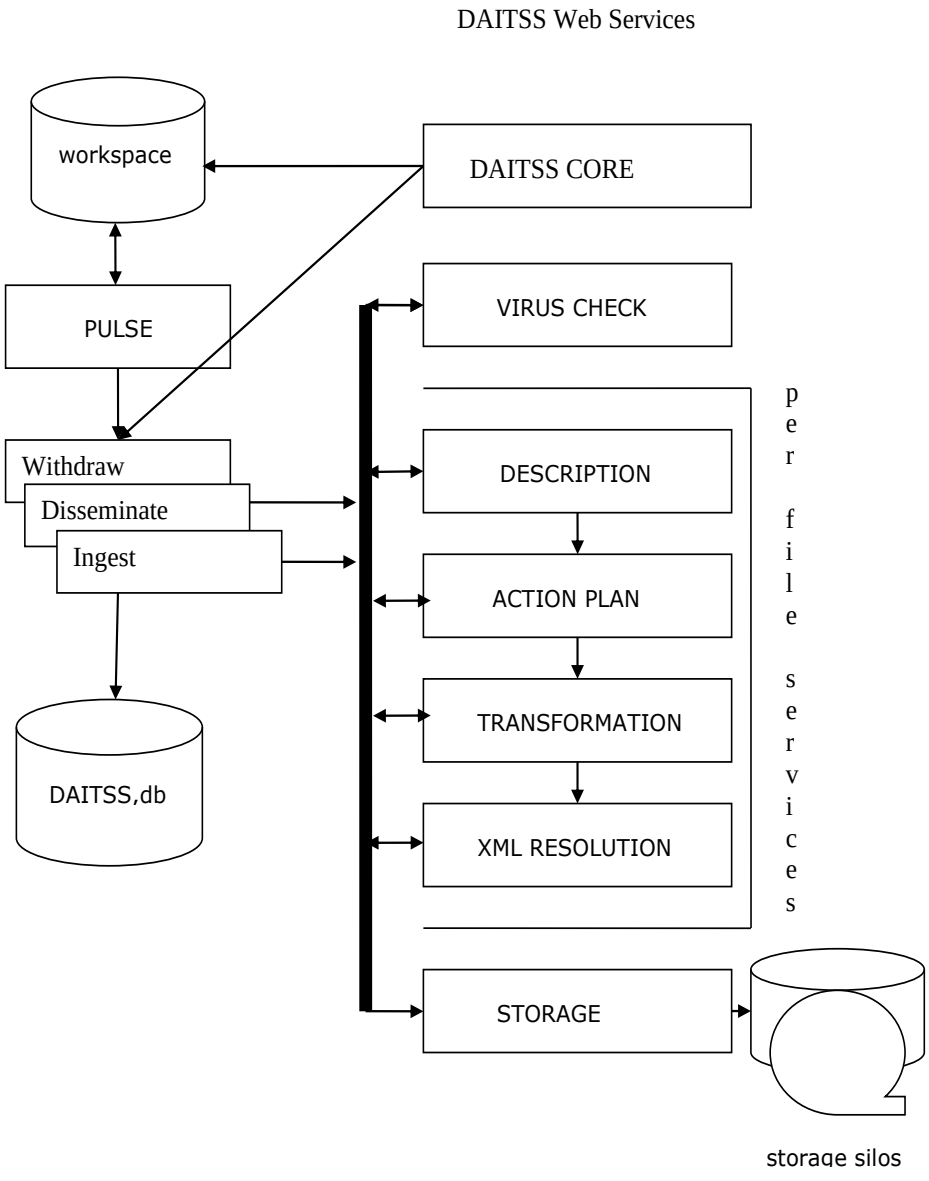
Reference Model for an Open Archival Information System (OAIS) is a standard of the Consultative Committee for Space Data Systems (CCSDS 650.0-B-1) and the International Organization for Standardization (ISO 14721:2003). It is a fundamental standard for digital preservation repositories, defining the functional model and information model that must be supported by the repository. DAITSS is closely OAIS-conformant, implementing the functional model directly and the information model via PREMIS.

The PREMIS Data Dictionary for Preservation Metadata defines core metadata that repositories need to know in order to implement long-term preservation. DAITSS supports the PREMIS data model, all mandatory data elements and most optional elements. DAITSS also implements format-specific technical metadata standards used in the community, including MIX for digital still images, AES for audio, textMD for text, and docMD for documents.

The Metadata Encoding and Transmission Standard (METS) is an XML document format for encoding packaging and structural information. DAITSS uses METS as the format for its SIP descriptor, AIP descriptor, and DIP descriptor. The Repository Exchange Package (RXP) is a XML document format that includes elements from METS and PREMIS specifications to create packages for transferring information from one repository to another. DAITSS includes code to transform its standard DIP into RXP format, and can ingest a SIP sent from another repository in RXP format.

DAITSS Architecture

DAITSS consists of DAITSS Core (a set of scripts and programs that execute various functions) and a series of RESTful Web services. Each component is described briefly below.



fly below.

DAITSS Core is a web service that contains the functions GUI, Submit and Request.

Submit is a function within DAITSS Core that accepts a SIP or batch of SIPs from a DAITSS operator or authorized user, and validates each SIP. If the SIP passes validation, it is copied into the DAITSS Workspace for Ingest processing. If errors are found, the SIP is not copied and an error message is returned to the submitter.

Request is a function within DAITSS Core that accepts a request for dissemination, withdrawal, or refresh from a DAITSS operator or authorized users, validates it, and copies it into the Workspace.

The **Workspace** is an area of storage (directory) defined to DAITSS in which the SIP (now called a WIP or Workspace Information Package) is stored as it is transformed into an AIP. The Workspace also holds dissemination and withdraw WIPs, which are generated to by **Pulse** to handle the processing of dissemination and withdraw requests.

Pulse is a program that automatically monitors the Workspace for new SIPs and new requests and starts the appropriate task for each one. It monitors progress through the steps of the task and keeps them moving along to completion. Pulse also scans the request table in the DAITSS database (**daitss.db**) and creates WIPs for requests that are ready for processing.

Withdraw, Disseminate and Ingest are processes (preservation tasks) that can be initiated by **Pulse** or the **GUI**.

The **GUI** is a function within DAITSS Core that provides a graphical interface to the DAITSS database and DAITSS management services. Using the **GUI**, an operator can do everything **Pulse** does and much more.

The **Virus Check Service** checks all files in a new WIP for viruses.

The **Per File Services** are iterated once for each file in the WIP. Together, the per file services – **Description, Action Plan, Transformation** and **XML Resolution** -- execute the DAITSS Preservation Function.

The **Description Service** has three components: file type identification, file format validation, and file description. Standard schema for format-specific metadata are used for description when possible.

The **Action Plan Service** uses the format of the file and an XML Action Plan document instance to determine whether normalized and/or migrated versions of a file should be created, and passes the file to the **Transformation Service** if so.

The **Transformation Service** uses instructions in the XML Action Plan to create a derivative version of a file.

The **XML Resolution Service** is invoked for each XML file in the WIP. It parses the XML file for references to external schema and downloads the referenced schema for later inclusion in the AIP.

The **Storage Master Service** mediates all access to archival storage via the **Silo Pool Service** (not shown here).

daitss.db is a fast-access database that contains selected information from the completed AIP, as well as a copy of the entire AIP Descriptor. In addition, it contains information operations needs to manage the system (users, accounts, projects, etc.), an audit trail of operations activities, and requests for withdrawal, refresh and dissemination.