A Pragmatic Application Of PREMIS

Mapping the key concepts to a real-world system

Jack O'Sullivan
Preservica Ltd
United Kingdom
jack.osullivan@preservica.com
0000-0002-0306-761X

Richard Smith
Preservica Ltd
United Kingdom
richard.smith@preservica.com

Alan Gairey
Preservica Ltd
United Kingdom
alan.gairey@preservica,com

Kevin O'Farrelly
Preservica Ltd
United Kingdom
kevin.ofarrelly@preservica.com

Abstract – a data model is an expression of how a system is intended to be used, and a statement of how it should interact with other systems. As part of the development of the latest version of Preservica, the underlying data model was significantly altered, informed by what went before and by reference to the experiences and best practices distilled into the PREMIS Data Dictionary. This paper reports on some of the key decisions made in the application of the PREMIS concepts to an extant digital preservation system.

Keywords – PREMIS, Data Model, Digital Preservation Systems

Conference Topics – The Cutting Edge: Technical Infrastructure and Implementation; Designing and Delivering Sustainable Digital Preservation.

I. Introduction

This paper describes some of the decisions made in the course of making improvements to the underlying data model upon which the Preservica product is built. In section II we describe the role and importance of the data model in a system, section III describes why this work has been undertaken and sections IV and V describe some of the key features and how they correspond to the PREMIS Data Dictionary.

II. The Role Of The Data Model

The data model of a system codifies how it views and understands the wider ecosystem in which it operates. It is a means of describing external realities, and of describing and enabling the processes performed by the system. As part of the system interface, it can shape how the system interoperates with other systems. Since it is generally defined to enable the processes the system is designed to perform, it can act as limitation on what the system is able to do.

The data model is also the lens through which the system is viewed. As such, it shapes the way in which users and developers think about the system, its capabilities and its limitations. This means that even if the data model can support an operation, users may be reluctant to perform it, or even unaware that they can, and developers may seek to artificially restrict it. Conversely if something seems like it should be possible, users and developers can be encouraged to use the system in ways it was never intended, often with less than optimal results.

The data model also reflects an understanding of what the system is, or should be, at a particular

16th International Conference on Digital Preservation
iPRES 2019, Amsterdam, The Netherlands.

DOI: 10.1145/nnnnnnn.nnnnnnn

moment in time. As the world around the system changes, the definition of what the system can, should or should not do is likely to change.

To allow a system to be used flexibly, and to perform functions that could not have been anticipated at the time of the original design, a data model must itself be flexible and extensible. There are however pitfalls of an overly flexible model. If the model does not clearly describe the required functionality, or if it is overly permissive in interpretation, then actual implementations will tend to diverge to the point that managing system changes becomes problematic. A good data model is thus always based on a trade-off between allowing users the freedom to do what they need and constraining them to behave in a way that is consistent with the intended use.

III. The Preservica Data Model Challenge

Preservica has been built on an eXtensible Information Package-based data model (XIP) since the initial end-to-end preservation system was developed in 2006. Minor extensions to this model have been made with each subsequent release through to v5.10 (July 2018), but the fundamentals of the model have not been significantly altered in this time. This model was built with the intention of describing processes that would be generally applicable to anyone performing digital preservation, in the light of the original PREMIS definition published slightly earlier [1]; however, since the original users of Preservica were archives, this model was initially tested and validated specifically in the context of the requirements of an archival setting. It is a testament to this original design that as its user base has swelled to include libraries, museums, and business records repositories, across multiple sectors, each bringing different, sometimes competing requirements, the model itself has held firm, able to satisfy most of these requirements. The XIP model has proven itself to be flexible enough to describe several complicated use cases, some of which are referred to within this paper.

Increasingly however, it has become a constraint on the functionality Preservica can offer, or at best a complication, making the development of new features slower and more complex than they might seem at face value. It has also become increasingly clear that the way in which the model has been implemented by many users has diverged from the intention of the original design in some areas. This often leads to having to decide whether a new feature should interpret the model in its "pure" form, making the behavior unintuitive for users, or in the way it is more commonly used, gradually breaking the assumptions that can be safely made. As such, version 6.0 of Preservica is based on a new data model that seeks to address these limitations in the existing XIP.

Whilst a "new data model" sounds like a large green field opportunity, the development has been shaped from the start by several constraints. Firstly, there is the decade plus of data amassed in Preservica systems world-wide in the older XIP format, which dictate that clear mapping from XIP to the new model should be available; whilst it is desirable to be able to perform the reverse mapping, it was not a requirement to ensure a "round-trip" would return exactly the same data in exactly the same structure. Secondly, the longevity of XIP indicates that it has been a successful model in many ways, losing sight of what it does well would risk throwing the baby out with the bathwater. Thirdly, Preservica does not exist in a vacuum. It must interact with other systems, from standard operating and file systems, through widely used content management, catalogue, and discovery systems, through to bespoke access, workflow, and storage systems, all of which means mapping to other formats and standards must be possible. Perhaps most importantly, Preservica is intended to be a long term preservation system, and its users have responsibility for long term planning, so being able to describe the model in terms of a recognized and accepted industry standard is paramount, hence all stages of the design have informed by version 3.0 of PREMIS [2], and the process has been, in effect, an exercise in an implementation of PREMIS in the context of an active preservation system.

Finally, Preservica will need to interact with systems that use the operating system/file system model of folders/directories and files. This is also the default mental-model that end users tend to bring to any hierarchical presentation of information. Whilst being able to map to this model of the world may seem so obvious that it doesn't need to be stated, the concepts involved in digital preservation mean that this is not necessarily a straightforward process.

The high-level conception around how the existing data model maps to PREMIS entities and file system models is given in Figure 1.

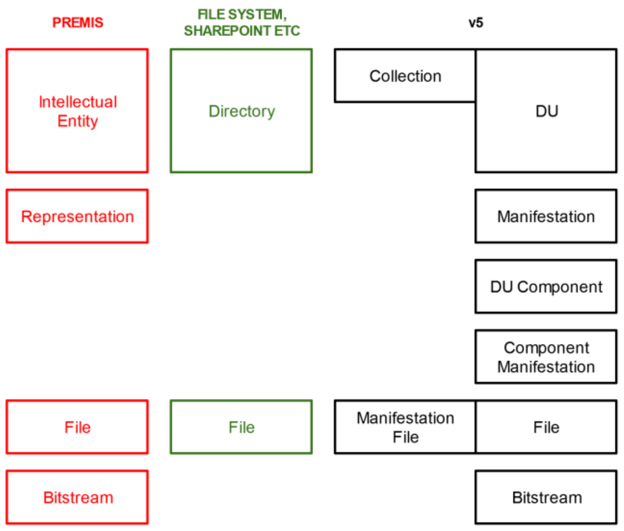


Figure 1 - A mapping comparison of high level entities in PREMIS, a standard File System, and Preservica, up to version 5 (XIP data model). Entities at the same vertical level have the same responsibilities.

The “DU” in this mapping stands for “Deliverable Unit”. These are described as “information objects that are the conceptual entities (records) that are delivered by archives and libraries”; they are hierarchical in that a DU may contain nested “child” DUs, but are also considered, confusingly in many cases, atomic units of information. A “top-level DU”, that is a DU with no parent DU, is taken as the implementation of an Archival Information Package as described in the OAIS reference model.

The “Collection” is described as “a hierarchical grouping of Deliverable Units”. They are hierarchical in that a Collection may contain nested “child” Collections, but they are effectively just metadata records for organizing DUs.

In mapping this existing model to PREMIS, both Collections and DUs have historically been regarded as Intellectual Entities.

The “Manifestation File” is a conceptual entity creating a link between a “File” (which follows the PREMIS definition of File) and the “Manifestation” (a close analogue of the PREMIS Representation). This allows a single digital file to simultaneously be part of multiple Manifestations (and indeed multiple separate DUs).

This abstraction between files as “physical” entities and the construction of higher level “conceptual” entities allows for some of the most complex use cases of the data model. For example, one user of the system stores television broadcast video. The “physical” files represent distinct time units of broadcast (for example the hour of video stream broadcast between 00:00:00 and 00:59:59 on a given day). The “conceptual” entities, the DUs, represent actual programmed content, (e.g. an episode of the daily news). The programmed content may be wholly contained within a single time unit, may approximately equal a single time unit, or may require content from two (or more) consecutive time units. The same “File” is thus potentially a component of multiple DUs. Secondary “Manifestations” can then be created which splice the relevant parts of the underlying streams into a single video file representing only the programmed content.

Similarly, when performing web-crawls to preserve a website, the content of the crawl is stored across multiple “physical” WARC files (so that no individual file is too large), but creates a single Manifestation of a single conceptual “DU”. For the most part, users interact with the DU, particularly for rendering, where the whole web crawl is played back rather than individual files.

The “DU Component” and “Component Manifestation” are elements attempting to record and reconcile how a single DU is composed of multiple elements. In most cases each component is effectively comprised of a single “File”, but conceptually, a component such as an email may require multiple files (the message itself plus any discrete attachments). This recognizes the idea that a single “record” may comprise multiple pieces of otherwise distinct content.

The rest of this paper will describe Preservica’s new data model in terms of the external realities we are modelling, how these are modelled in PREMIS, and the decisions, trade-offs and compromises made. References to “XIP” should be read as the data model of Preservica versions up to 5.10, whereas “XIPv6” refers to the new data model.

The exercise of performing an in-place update of a V5.X system to a V6 system has been developed in parallel to the development of the new model to ensure that existing customers will be able to upgrade and that information will be presented in the new model in the way that best represents what it actually is. This process itself merits further discussion but is beyond the scope of this paper.

IV. ENTITIES AND OBJECTS

The goal of digital preservation has long been described in terms of preserving digital *information* [3]. In a general sense, information is obtained when we use some software to interact in some way with some digital data. All digital data can be thought of as streams of bits, binary ones and zeroes. The data only become information when the software used to interact with it can interpret those ones and zeroes as higher-level abstractions. The lowest abstraction above the stream of bits, is a stream of bytes, each byte consisting of 8 bits. This is generally the lowest level aggregation of digital data that software systems address and process. Above this, bytes may be interpreted either individually or in aggregation as simple entities such as numbers, characters, words, or more complex entities such as arrays.

In order to consider the information we can abstract from these data to be "preserved", we need to be confident that we have some piece of software that is able to correctly interpret a particular stream of bytes as the intended series of more complex entities. In some cases, it is sufficient that the software can render the correct "human-readable" entities such as strings, numbers and images, in other cases the "machine-readable" entities manifest as some form of behavior that must be correctly replicated.

Whilst processing these data, the software builds an internal state map of how certain bytes or groups of bytes should be interpreted, potentially updating as user inputs modify the data. These data need to be persisted between sessions in such a way as allows the software to re-build this internal state map; to do this, the data is written out, or serialized, according to a set of rules, typically referred to as a format. It is this format that allows software to interpret data, and it is changes to the list of formats that a given piece of software is aware of that leaves digital data vulnerable to becoming unusable.

A. Aggregations of Data

Operating systems and storage systems group the formatted bytes that need to be persisted into units called files. Just as the byte is the lowest aggregation of data that software interacts with, the file is the lowest aggregation that the operating and storage systems will address, as evidenced in the PREMIS definition: "*A named and ordered sequence of bytes that is known to an operating system*" [2]. It is often assumed that the file is an atomic aggregation of data that can be regarded as an intellectually complete unit of information, whereas in fact it is an artifact of the implementation of data persistence, part of the data model for an operating system or file system.

However, files are not necessarily atomic. PREMIS specifically makes provision for files to be considered as containers or aggregations of other units of information, drawing a distinction between two different types of potentially embedded units, a Bitstream and a filestream.

Bitstreams are first class Object entities within the schema, defined as "*Contiguous or non-contiguous data within a file that has meaningful properties for preservation purposes*" [2].

It is further clarified that these are *"only those embedded bitstreams that cannot be transformed into standalone files without the addition of file structure (e.g. headers) or other reformatting to comply with some particular file format*" [2]. An example would be the video and audio streams within a video file; although encoded according to well defined schemes, these are not necessarily directly extractable as standalone files.

If an embedded stream can be transformed into a standalone file without the addition of extra information, for example a TIFF image embedded within a zip File, PREMIS considers it to be a filestream. The filestream is not a first-class Object entity within the schema, although all properties of files apply to filestreams as well.

In either case, a bitstream is only significant if it can a) be identified and characterized independently of any overall file, and b) carries a separate preservation risk. This describes both the video and audio streams within an AV file, and individual documents within a zip file.

The logical separation of files and filestreams makes sense in the context of relating to external systems, where a file is a physical reality, whereas a filestream is an abstract concept. Internally however, to perform any preservation action, Preservica will have to retrieve either a file or a filestream from a long-term storage location, creating a temporary local copy and as such the distinction between the two becomes rather academic. It is also clear that the definition "contiguous or non-contiguous data that has meaningful properties for preservation purposes" fits both files and filestreams, as well as describing why they are interesting to a preservation system. As such, both files and filestreams are represented internally in XIPv6 as “bitstreams”.

It should also be recognized that an individual file can be a “sub-atomic” unit as well as a “super-atomic” unit. In this context, we should understand that atomicity can mean different things and whilst there are several senses in which this may occur, these can generally each be considered as one of two categories.

1) Cases where the file is technically atomic: there exists software that is capable of interpreting the file in isolation, and where examining the file provides a self-consistent and interpretable set of data to a consumer, but where the true information content can only be understood in the context of other files. There are two important sub-cases here, firstly where the other files are unknowable from a technical interrogation of the files, and secondly where the other files are explicitly referenced. In the first case, the context will have to be defined and recorded explicitly by a user. In the second, this context can be inferred and recorded automatically in software.

This case requires higher-level entities to be introduced to the data model and will be addressed in later sections.

Examples of technically, but not informationally atomic files include:

- A series of word processing documents, each of which represent the minutes for a particular meeting, but where the meetings all relate to a single project and the complete set of minutes should be considered the atomic unit of information to be preserved.
- A Digital Video package where video, audio, and additional data such as subtitles are stored in separate files. Each file might be usable in its own right, but the true information context is revealed only by software that interprets all the files together. Such an example is given in [4].
- A web-page, where a single HTML file can be well structured and readable by a browser, but where images are referenced rather than embedded, or links to documents exist; in this case, all the “linked” files must be available and returned by following the links, before the full data is provided to user and before the information content can be understood.

2) *Cases where the file is not technically atomic*: there is, by design, no software capable of correctly interpreting this file in isolation. In this case, there may well be alternative means of storing the same information that would be a single, technically atomic file.

Examples of technically non-atomic files include:

- Certain types of disk images (e.g. BIN/CUE) or scientific data sets (e.g. ENVI [5]), where the raw binary data is contained in one file, but the header information that instructs the software in how to read that data is contained in a separate file. In this case, the “data” file without the headers is a meaningless binary dump, and the header file without the data is better understood as metadata than actual data. In some cases, these could be considered PREMIS Bitstreams, with the additional “file structure (e.g. headers)” coming from a different file, but they are already, by definition, a “stand alone file”.
- A case where a single large file has been partitioned into many smaller files for the purposes of by-passing storage or transport limits, and where only by recombining the files into a single large file can the data be understood by any software. Again, they are Files by definition, but in these cases, it is not clear that the Bitstream definition applies at all. They are not just missing “file structure” or in need of re-formatting, but rather are missing actual bytes of content, often with boundaries in the middle of a complex entity such as a word or array.

It is clear from this that the risks associated with format obsolescence should be properly thought of as being an “atomic content” issue rather than strictly a file issue. In this light, XIPv6 uses a “Content” entity to track content with a specific “format”. This entity can contain multiple bitstreams, although in practice, in most cases a “Content” entity has a direct mapping to a single bitstream.

B. Higher Level Entities

Having established that even in the case of technical atomicity, the file is not necessarily an atomic unit of information that should be preserved, we are left with the need for a higher-level entity to describe that unit. This is an Intellectual Entity in the PREMIS model.

Intellectual Entities can include other intellectual entities and may have one or more digital or nondigital instantiations. Digital instantiations are Representations in the PREMIS model.

There is an implication here that an Intellectual Entity has two responsibilities. The first, as a hierarchical aggregation of material, and the second, as an expression of an individual piece of information.

XIP's "Deliverable Unit" satisfied this definition, however combining these two responsibilities in a single entity type made the system complex and allowed different customers' usage of the same system to diverge.

For our earlier "Archives" customers, the DU was considered to be an immutable record, something that could be defined and laid out during ingest, and never altered. In this interpretation of the model, the hierarchical nature of the DU was fairly abstract and secondary to its "immutable" nature.

The fact that DUs were hierarchical became more of a primary concern for two main reasons. Firstly, a growth in the user base of "non-Archival" customers, for whom records are not necessarily fixed for all time, but may require restructuring. Secondly, a growth in the number of less technically expert users, for whom a hierarchical entity like a DU mapped in their mental model to a folder/directory. The second of these was probably not helped by a choice to represent DUs in the GUI with a "folder" icon, whilst Collections were represented as "filing cabinets".

Requests from both the non-Archival and less expert users to be able to "re-arrange their folders" increased in both volume and legitimacy of use case. The implementation of post-ingest restructuring was a good example of a change that appears "trivial" to users, but was complex to implement. It is also an example of a change that broke several assumptions implicit in the way the data model was envisioned, but not explicitly codified. For example, the XIP model had an "IngestedFileSet" entity, which broadly represented the set of a physical content ingested as part of the same transfer. A working assumption that files in the same FileSet were directly related by something other than coincidence of timing was broken by allowing Files to be spread at will through the logical hierarchy of the repository.

For XIPv6 therefore, it was determined that a separation of these responsibilities should be undertaken.

A top level intellectual entity should be created as a means of aggregating related content and providing a primary hierarchical structure.

This "Structural" entity can be part of another structural entity and can be a parent for any other intellectual entity, giving rise to hierarchical structure. It does not have any direct instantiations itself, and whilst the metadata describing it may need to be retained for the long term, it is not considered to have long term preservation risks beyond that of metadata storage. This entity is analogous to the "Collection" from the XIP model, however as the details of the metadata fields available has changed between the two models, and to ensure that the term "collection" does not have confused meaning in internal communication (and communication with users), a different name ("Structural Object") is used in XIPv6.

The second intellectual entity that XIPv6 defines exists as a means of defining the unit of information that should be considered atomic for the purposes of preservation. This is most closely related to the DU in XIP but differs in important respects. By definition, as an atomic unit, this "Information" entity cannot be part of another "information" entity (it will exist in the hierarchy within a "Structural" entity). This is probably the most important respect in which it differs from the DU.

It does allow for multiple instantiations, as discussed further below.

The assumption that files are atomic information units (as mentioned in IV.A above) manifests as requests from users to be able to perform operations such as movements and deletions on individual files. In most cases, a file is a complete set of data required for an atomic entity, and such requests are entirely reasonable, however a long-term preservation system must protect against those cases where that is not true, and changes to an individual file risk corrupting the entity.

Defining a higher-level Information entity such as the "Information" entity allows us to define this as the level at which actions such as deletion or movement are performed. As the "atomic" entity of information, it is also the natural level at which preservation actions such as normalization, migration and rendering should be performed, and is thus the "natural" level to define as an AIP. In most cases, this Information entity is the base unit that should appear to the user, thus in the user's mental model it will likely equate to "file", solving the issue of allowing "individual file deletion" and allowing the "Structural" entity to map more correctly to "folder/directory".

The conceptual structure of entities within the repository means that another PREMIS concept, the Relationship, is also required in the model. There are clearly inherent parent-child relationships within the structure, but an explicit Link between two or more entities can also be defined, allowing the model to record connections such as "derivedFrom", "inResponseTo", that may not be otherwise apparent.

C. Instantiations

The PREMIS definition of a Representation provides for it being "*The set of files, including structural metadata, needed for a complete rendition of an Intellectual Entity*". Within the XIPv6 model, this is applicable to "Information" entities, but will rely on the more generic "Content" rather than files.

The need for more than one instantiation of an entity can arise from multiple requirements. There are however, two obvious cases.

In the first, there is considered to be a long-term preservation risk due to format obsolescence, i.e. it is feared that there is no available software capable of correctly interpreting the content, or that access to such software is not feasible or possible for the repository, and as such long-term management should occur within the context of an institutionally supported format. In such cases, the transformation of content is at issue. A standard example of this concern is word processing documents in an outdated or proprietary format, such as WordPerfect or Microsoft Word. In this case, having an instantiation in OpenDocument Text (ODT) might be the desired outcome.

In the second, there is considered to be an issue with the presentability of the content as is. In such cases the holding institution may not consider the original content to be at risk but may consider that the designated community will not have the means to interact with the information in its original form, or simply that there is a more convenient alternative means of dissemination. Examples of this might include a desire to have lower resolution/lower bit rate images/videos for access to conserve bandwidth; or a digitised manuscript, book or journal where the digital information is in the form of a series of high resolution images, but a PDF or eBook is the more natural digital surrogate.

In the first case, the format of the content may need to be changed, but the fundamental form of information remains unaltered, WordPerfect and ODT are merely different encoding of the same fundamental "word processing document" type. In the second case, the form of the information may be altered, but the content still represents the same intellectual entity, a book is still a book whether presented as a series of TIFFs or a single PDF.

If the first type of migration can be considered as happening at the Content entity level, then the Representation of the Information entity level can always refer to the same Content entity. In our example, the Information entity might be a report, with a single Representation that is comprised of a single "word processing document". After migrating from WordPerfect to ODT, the Representation simply provides a later "Generation" of the document. It is in these cases that serious consideration of significant properties, and significant significant properties [6] must be made in validation of the migration.

The migration from a series of TIFFs to a PDF creates an entirely new Representation that leans more naturally to the idea that "this book can be represented as a collection of pages, or as a single complete document". In this migration, it is more problematic to rely on significant properties as a validation as the significant properties for an image are likely to look very different to those for a document; similarly, certain properties like image size might deliberately be changed in the creation of a presentation version.

It is in the second sense that XIPv6 uses the term "Representation", using "Generation" to reflect the set of files needed for a complete rendition of an atomically complete piece of content. This allows the use of "Normalization" [7] to describe the process of creating new generations of content, and the more generic "Migration" [7] to describe the process of creating new representations of information, allowing different policies to be specified for each and different levels of validation criteria to be applied to each.

In XIP, both forms of "instantiation" were represented by "Manifestations" of a DU. This made it difficult to assess whether pieces of content in different manifestations should be considered to be intrinsically linked for the purposes of validating transformations. By separating Generations from Representations it is clear that all Generations of a piece of content should ultimately derive from the same source and have some shared invariant “properties”, but content in different Representations may be entirely unconnected (except insofar as the combined content of one Representation should convey the same “information” as the combined content of another).

A simplified summary of the entities in the XIPv6 data model, alongside the levels at which they will relate to other logical data models in the system, is

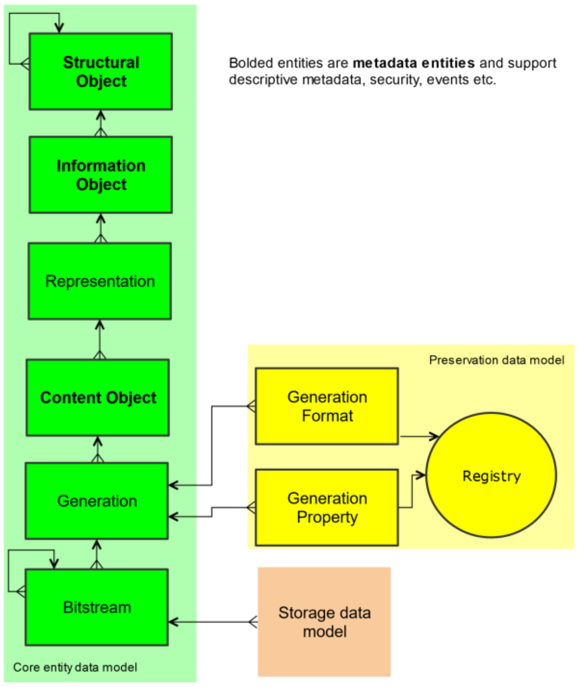
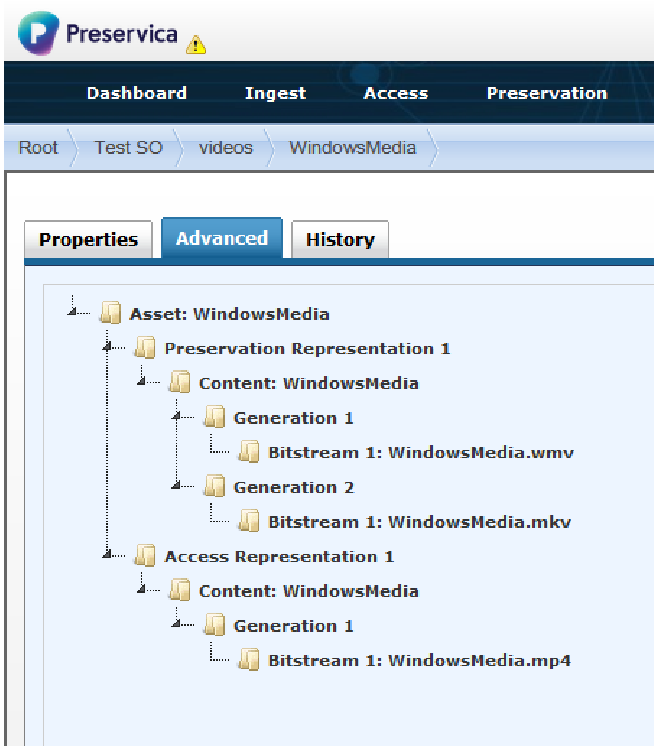


Figure 2 - Simplified summary of the XIPv6 Data Model

depicted in Figure 2.

Figure 3 shows a screenshot of the details of an Asset in a Preservica V6 system, this highlights several of the concepts. The “breadcrumbs” in the light blue bar towards the top show the hierarchical structure, where in this case “Test SO” and “videos” represent Structural Entities and “WindowsMedia” represents the Asset in Question.

The tree in the centre of the image shows the Preservation Representation, which consists of a single piece of content. The first Generation of this content was a Windows Media Video (wmv) file. At some point in its history, this content has undergone normalisation to create a second Generation; in this case to a Matroska (mkv) file. There is also an Access Representation of this asset, which also consists of a single piece of content. In this case, the content is a MP4 video, which is more usable for streaming video access than either the original wmv or the mkv.



Asset: WindowsMedia
Preservation Representation 1
Content: WindowsMedia
Generation 1
Bitstream 1: WindowsMedia.wmv
Generation 2
Bitstream 1: WindowsMedia.mkv
Access Representation 1
Content: WindowsMedia
Generation 1
Bitstream 1: WindowsMedia.mp4

Figure 3 - Screenshot of an Asset in a Preservica V6 system

V. Events and Agents

As a preservation system, Preservica undertakes preservation actions that will need to be recorded as part of a chain of custody for the materials stored within it. These actions may be performed directly by the system or by deferral to third party tools and applications. Again, PREMIS defines the relevant entities to enable this record to be made, with Events and Agents.

Events in PREMIS are defined as “*an action that involves or affects at least one Object or Agent associated with or known by the preservation repository*”. What it does not specify is the granularity of record required, or the actions that should be audited.

Reference [8] discusses what event metadata needs to be recorded, specifically with reference to idea of “metadata bloat”. That discussion is helpful in guiding a decision but does phrase the discussion in terms of “the organization”. As a platform used by a large number of organizations, who may come to different conclusions in this regard, Preservica needs to ensure that enough metadata is recorded to satisfy any of its users.

Metadata bloat is unlikely to be a major concern in terms of the sheer volume of storage required as it is likely that this will always be marginal compared to the size of the repository as a whole, but it can impact on the performance of the system, particularly in reference to any process that attempts to extract the audit trail of a particular object, and so in designing this model, care was taken to try to avoid "event explosion".

To take the example of the characterization of content during submission to the repository, this requires running a format identification tool on each incoming bitstream. As a result of that, validation of the format may be required, followed by the measurement of technical properties, potentially by multiple tools. However, at a high level, all that has happened is that the Information has been characterized.

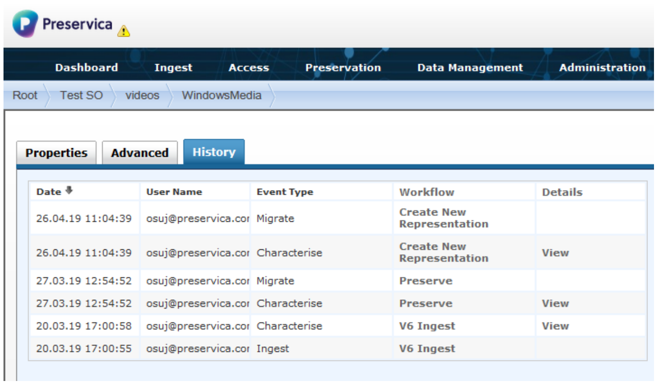
The fact that these actions happened should be surfaced to the user through the User Interface (UI), otherwise there is little point in recording it. From the earlier statement that the Information entity is the atomic unit of preservation, and the base unit presented to the user, it follows that this user presentation must happen at this entity level.

For an Information entity consisting of 10 bitstreams, the single process of "characterizing" an entity could result in at least 30 events being presented to the user. This is likely to be an overwhelming to a user, and risks burying pertinent information in a sea of detail.

This "event explosion" is avoided by asserting that Events in Preservica must be recorded against a high-level entity (generally the Information entity, but potentially also the Content itself), but that individual actions can be recorded against an event at a more granular level. This achieves the compromise of allowing the high-level events to be examined free from the minutiae of what happened, whilst ensuring those details are available if required.

In this model, the Events in XIPv6 do not store results directly. These are associated with each individual action.

The Event history of the Asset from Figure 3 is show in Figure 4. This shows "Characterise" events connected to each of the original "Ingest", "Preserve" and "Create New Representation" workflows, each of which also records an event. The details of the characterization events are recorded against lower level objects.



Date	User Name	Event Type	Workflow	Details
26.04.19 11:04:39	osuj@preservica.cor	Migrate	Create New Representation	
26.04.19 11:04:39	osuj@preservica.cor	Characterise	Create New Representation	View
27.03.19 12:54:52	osuj@preservica.cor	Migrate	Preserve	
27.03.19 12:54:52	osuj@preservica.cor	Characterise	Preserve	View
20.03.19 17:00:58	osuj@preservica.cor	Characterise	V6 Ingest	View
20.03.19 17:00:55	osuj@preservica.cor	Ingest	V6 Ingest	

Figure 4 - The Event History of an Asset in a Preservica V6 system

Agents have not been formally implemented as an entity in the model itself, but each event action is recorded as having been performed by a specific tool, or piece of software (generally also including details of the version of the software used). In cases where actions were triggered by a specific user, details of the user are also recorded. In such a way, there is a "soft" implementation of agents as meaning the user and/or the software performing an action.

This definition of Events formalizes the way in which events and actions were recorded in XIP, which allowed for the recording of some events as part of the metadata of the specific entities on which they acted, others to be recorded in specific audit tables in the database, and others to be use the process metadata of the workflow to record them.

VI. Discussion

Part of Preservica's aim is to provide a system that enables non-specialist, non-technical end users to ensure they can preserve their digital assets. To do this, the system must mask some of the complexity of the digital preservation process.

Having to balance our aim to conform to PREMIS with the need to continue operating the product with over a hundred live users, and millions of live objects amounting to hundreds of terabytes of live content, and to provide a system that masks the complexity of the digital preservation processes it provides has meant that the strict definitions of the Data Dictionary have not always quite met the exact need of this model. This is most readily apparent in two areas.

As such conformance to Level 3 [9] (through internal implementation) has not been possible.

However, we have attempted to define our entities to be as close as is possible to those in the Data Dictionary, with the intention that Level 2 (through export) should be possible, and even straightforward. In this way we can provide our users with a system-independent exchange format version of their data.

The basis of our entity model has been the Intellectual Entity, albeit we have specifically sub-classed this into three distinct types (meaning no XIPv6 entity explicitly implements the PREMIS Intellectual Entity). The first, the "structural entity", to provide means of aggregation and to allow multiple levels of description; the second, the "information entity", to define the unit of information to preserve; and the third, the "content" entity to describe the base level at which data is formatted in a way that carries some long-term preservation risk.

To attempt to break a reliance on assumptions of file-level atomicity, and because conceptually to the system it doesn't matter whether a particular stream of bytes is recognized on its own terms by the underlying storage and operating system, we have effectively consolidated the PREMIS file and filestream into a single "bitstream" entity.

We have modelled the record of specific preservation actions in the system around the PREMIS event, with a controlled vocabulary of event types, each with actions performed by specific named users or tools (i.e. agents).

By doing this, we believe that our mappings and exports should conform to level B of the PREMIS conformance statement.

We have not changed our existing security model, and so whilst our users may model their own access and permissions around the PREMIS Rights model, this is not done explicitly in the system itself.

From the inception of this project, conformance to PREMIS has been both a goal and a requirement, and one which we believe we have achieved whilst delivering a data model that will enable Preservica to continue to improve its digital preservation functionality.

REFERENCES

[1] PREMIS Data Dictionary v1, https://www.loc.gov/standards/premis/v1/index.html

[2] PREMIS Data Dictionary v3, https://www.loc.gov/standards/premis/v3/index.html

[3] D. Waters and J. Garrett, "Preserving Digital Information, Report of the Task Force on Archiving of Digital Information", Council on Library and Information Resources, 1996

[4] K. Van Malssen, "Digital Preservation Metadata Practice for Audio-Visual Materials", in Digital Preservation Metadata for Practitioners, Springer Press, 2016, pp 45-58

[5] ENVI Image Files, Harris Geospatial, https://www.harrisgeospatial.com/docs/enviimagefiles.html

[6] P. Lucker, C. Sijtsma, R. van Veenendaal, "Significant Significant Properties", *iPres 2018*

[7] Library of Congress Event Type controlled vocabulary, http://id.loc.gov/vocabulary/preservation/eventType.html

[8] E. Cochrane, "Implementing Event and Agent Metadata for Digital Preservation", in Digital Preservation Metadata for Practitioners, Springer Press, 2016, pp139-150

[9] P. McKinney, "Conformance With PREMIS", in Digital Preservation Metadata for Practitioners, Springer Press, 2016, pp247-257