FP7-ICT-2007-3-231161

Internal Deliverable ID3.3.1

Media formats, identification methods and implementations for multivalent preservation

Ann Gledson and Paul Watry – University of Liverpool

27/05/2010

Document administrative table

Document Identifier	PP_WP3_ID3.3.1_multivalent_R0	Release	0
Filename	PP_WP3_ID3.3.1_multivalent_R0_v1.02.odt		
Workpackage and Task(s)	WP3 Data management and processing for media preservation WP3T3 Multivalent approaches to long-term audiovisual preservation		
Authors (company)	Ann Gledson (UoL), Paul Watry (UoL)		
Contributors (company)			
Internal Reviewers (company)	Kurt Maçjen (JRS), Jean-Hugues Chenot (INA).		
Date	27/05/2010		
Status	Release		
Type	Part of Deliverable		
Deliverable Nature	R - Report		
Dissemination Level	Public		
Planned Deliv. Date	31/12/2010		
Actual Deliv. Date	23/02/2010		

Abstract

The objective of this report is to provide guidance on the range and nature of problems in audiovisual digital preservation, and the potential for Multivalent and iRODS technologies to address them. We illustrate how this is the most reliable method, when compared with other alternatives such as file conversions and creating wrappers/bindings to native libraries.

DOCUMENT HISTORY

Release	Date	Updates	Status	Distribution
0.01	22/12/09	First Draft	Draft	Confidential
0.02	02/05/10	Reviews Completed and related amendments made, and also: Added File format Identification ($4.4) Added Example Policies ($4.2) Added section on relationship with other PrestoPRIME tasks ($2)	Draft	Confidential
0.03	02/06/10	Updated glossary, Added OAIS Diagrams Improvements to $4.3 (Integrated Preservation Environment)	Draft	Confidential
0.04	16/02/10	2nd Reviews Completed	Final Draft	Confidential
0.05	19/02/10	Added Note to start of $2	Final Draft	Confidential
1.00	23/02/10	Finalised, delivered	Release	Restricted
1.02	27/05/10	Finalised for public dissemination	Release	Public

Table of Contents

Scope

The European Commission supported PrestoPRIME project (www.prestoprime.org) is researching and developing practical solutions for the long-term preservation of digital media objects, programmes and collections, and finding ways to increase access by integrating the media archives with European on-line digital libraries in a digital preservation framework. This result will be a range of tools and services, delivered through a networked Competence Centre.

The purpose of Work-Package 3 is 'to develop an architecture that may be used to implement a distributed and shared repository for long-term sustainable audiovisual collections; to demonstrate extensibility by federating with collections in other data management systems'. Consequently, this document describes the Multivalent and iRODS preservation architecture, focussing specifically on:

(a) The nature of file formats, their associated vendor-led, proprietary technologies and the problems caused in the preservation of digital data. These problems are exemplified in the Audiovisual digital preservation world where a large variety of encodings, formats and containers exist, each with different codecs and implementation issues.
(b) A brief list of the currently available approaches to solving the problems of file format variety, the dependence upon transitory technologies and the inherent dangers of format / technology obsolescence.
(c) A detailed description of the Multivalent architecture, in the context of audiovisual digital preservation, and its potential to overcome the above preservation challenges, by removing the dependence on format specific, vendor-led technologies and by maintaining platform independence.
(d) A list of the most important file formats currently in use in the broadcasting and digital cinema industries, and their inclusion / potential for inclusion in the Multivalent framework.

The objective of this report is to provide guidance on the range and nature of problems in audiovisual digital preservation, and the potential for Multivalent and iRODS technologies to address them.

Finally, the present work-package is set in the context of a proposed demonstration of the above technology, which aims to evidence its potential as a fully implemented and working solution.

1 Executive summary

This report is a justification for the selection of Multivalent and iRODS technologies in order to solve the problem of audiovisual data obsolescence. The following is a summary of its findings:

Problems with Audiovisual File Format Migration

Digital file format migration is problematic, as over the course of 50-100 years there are likely to be many new popular audiovisual formats and each time files are migrated, there is a considerable risk of data loss. Also, it may not be financially feasable to migrate collections that are petabytes in size. We therefore emphasise the need to store the original data along with the encoding method. We pursue an approach based on persistent objects: the original data, an infrastructure independent parser, and parsers for transcoding the data, (to include 'on-the-fly' parsing). The proposed Multivalent technology allows the development of policies that apply incrementally more powerful algorithms to decompress video data that has been parsed by the Multivalent media adaptors.

The Proposed Technologies

Multivalent is a document parsing / rendering technology which is completely independent of application features and is written in java, in order to maintain platform and architecture independence. Both the Java Virtual Machine (JVM) and Multivalent are based on open specifications and are available with full source code, under an open source licence. We can therefore read and convert the media engine and Multivalent source code to a new language as and when required. In addition, the JVM can itself be recreated or converted if it is no longer available.

We argue that Multivalent avoids architecture obsolescence because:

1. The JVM is such an important part of the programming industry that it will be ported to new environments for the foreseeable future, possibly indefinately, without any extra efforts by the media industry.
2. If for some reason, Java does become obselete, the fact that the Java code is stored means that it will be available for translation; and due to the investments made in Java, such translation will be a problem that will be shared by many, thus causing such translation tools to become widely available.
3. As Java itself is open source, the media industry could undertake to port to new architectures between the time when 1 (above) is still valid and if 2 (above) were to occur.

Future users will have access to a complete description of the rules for encoding and decoding a file format and also the complete code that allows the understanding of how to render and implement all the features of such a format. Direct interaction with data from the current (future) operating system and user interfaces would be possible.

Multivalent media engines are the sources of *representation information*, which allows the recreation and interpretation of the information object in the *OAIS model*. Multivalent requires adherence to the Universal Virtual Computer (UVC) principle, which means the C libraries cannot be used directly, and codecs / transcoders must be written in java. One alternative to this is library level emulation, which converts the C libraries to java bytecode.

The proposed underlying storage technology is *iRODS*, a policy-based approach to distributed data management. iRODS will store the video stream along with the Multivalent software. Both objects will be migrated to new storage technologies in the future, as and when they become available.

The iRODS and Multivalent architecture is *infrastructure independent.* The archivist controls the properties of the video stream, including the policies and procedures that govern record management. This independence is important as data is distributed across multiple types of storage systems and across a federation of multiple data grids.

Integrated Preservation Environment: The Multivalent technology (coded in java), the media adaptors for each data type and whatever is required to preserve the JVM in the future, are all archived. Emulation then consists of supporting the original operations for manipulating the digital entity. Migration consists of porting the JVM to new technologies as required. The digital entity itself remains unchanged, but new, future operations are possible.

The *audiovisual formats* to be included in Multivalent

Due to the complex nature of audiovisual files, the support of a wide variety of media types is expensive. Codecs can either be written directly for Java, the preferred option, or they can be written using native C library emulation, to expand more quickly and easily to a wider set of supported formats.

An extension to our current Multivalent browser has been implemented, in order that it can support already available Java codecs. For example, there is currently support for MPEG (1,2,4) and OGG files. The next priorities are to write java codecs for MXF wrapped JPEG2000 and MXF wrapped D10.

2 Problems with Audiovisual File Format Migration

As pointed out by Herr in The Digital Dilemma [1], the main problem in the design of digital preservation systems is that 'the period during which such assets need to be accessible is very long – much longer than the lifetime of individual storage media, hardware and software components, and the formats in which the information is encoded' … and therefore, 'a dynamic approach that anticipates failure and obsolescence will be essential'.

More specifically, if audiovisual files are encoded and simply archived as is, and the vendor of the encoding technology disappears, the data itself disappears as the method of decoding the data is no longer accessible. An example is given in the Digital Dilemma, of such file obsolescence:

> *"The early digital data from the NASA Viking probes launched in 1975 was transmitted from Mars back to the Jet Propulsion Lab in Pasadena, California, where it was recorded on magnetic data tape, analyzed by scientists... and then archived in a cool, dry data warehouse and left undisturbed until 1999 when USC neurobiologist Joseph Miller asked NASA to check some of the old Viking data. NASA found the tapes... but could not find any way to read them... the data was in a format that NASA had long since forgotten about. Or as Miller puts it, "The programmers who knew about it had all retired or died."*
>
> *(Herr, The Digital Dilemma p.31 [1])*

The problem becomes more acute as the digital revolution takes hold of the audiovisual world. One of the most important problems in the context of Multivalent, is that the industry is used to working with analogue formats that can be read by the human eye, so the creation of new hardware that can read these materials in the future is always fully conceivable. Digital data, on the other hand, can only be read by software which knows about how the data was encoded to that format. More specifically, a file format specifies the organisation of data in a file, so that conforming files can be interpreted and rendered by certain technologies. Such formats are created in the context of a certain type of usage and if they change, then so does the technology, which leads to problems of obsolescence.

Another alternative is the migration of files from one format to another, as such formats become popular and then later obsolete. An essential capability that is being explored within the PrestoPRIME project is the concept of on-demand transformative migration of audio-visual records. The major problems with this approach is that firstly, there is a high likelihood of losing data, during the migration process and secondly, for collections that are petabytes in size, it may not be feasible to migrate records to new data formats each time it is required.

In addition, in the scope of the OAIS reference model, file format specifications need to be preserved, in order to allow future interpretation of the information expressed in the data. Thus, in a future where there is need to interpret a specific type of data, lacking any application that is capable of interpreting it, someone would be able to create an interpreter for the file format.

A technical specification of a file format alone is often not a sufficient description in order to recreate the original rendering and features, and often lacks fundamental parts required for correct implementation. The source code of media engines can provide a standardised

view on parsing and rendering different file formats. This includes for example rendering of file formats that are not compliant to the file format specifications, but are normally interpreted correctly by current access software. The complete source code of a parser and renderer is thus needed to ensure future interpretation and the completeness of the representation information. More specifically in the context of audiovisual data, the video encoding method, as well as the encoded data itself, should be stored.

We pursue an approach based on persistent objects:

- The video format of the original record is preserved without any changes.
- An infrastructure independent method is provided to parse the video format using the Multivalent Browser technology.
- Parsers are written that migrate a record from an original format to the new desired display format, for example algorithmic colour transformations.
- When an object is accessed, the transformative migration to the new display format can be executed on-the-fly

Future algorithms are incrementally more powerful than existing ones, with the potential to further minimise the artefacts present in the decoding signal and to improve the image representation. The proposed Multivalent technology allows the development of policies that apply incrementally powerful algorithms to decompressed video data, parsed through Multivalent. This fits the use case of digital cinema, which requires the highest levels of fidelity to be maintained over a long time-span (greater than 100 years).

3 The Proposed Technologies

This section details the proposed technologies of Multivalent and iRODS, with the advantages of each being outlined in sections 4.1 and 4.2 respectively. Section 4.1 outlines the Multivalent principle, how it fits in with the OAIS (Open Archival Information System) reference model and compares it with conventional approaches to the problem of file format variety. Section 4.2 describes the benefits of the iRODS (integrated Rule-Oriented System), highlighting its infrastructure independence and listing example preservation policies that might be implemented in this system. Section 4.3 then describes how these technologies are integrated and finally, Section 4.4 describes how file format identification will be performed within this environment.

3.1 Multivalent

We are proposing the use of Multivalent Media engines: these are components of a well established framework for document parsing and rendering (the Multivalent framework) that allow the complete parsing and rendering of theoretically any file format.

This approach has several advantages. Media engines are independent of the application features that are normally implemented by the file format editors, as they are focused on interpretation and rendering. This way the source code can avoid all the routines irrelevant to the parsing/rendering, which would contribute to the overall complexity. Multivalent core features are contained in a small, compact core that allows a very flexible extension mechanism; Multivalent Media engines are written in Java, a system that abstracts from the characteristics of the Operating system in use, and that is compiled to a bytecode that is also architecture independent.

The JVM that can execute Multivalent is based on open specifications; multiple, independent implementations of such specifications already exist, and the reference implementation is now available with full source code, under an open source license. Apart from the Java language, the virtual machine supports programming in a variety of languages[1]. We argue that these features of the JVM give in fact the possibility of solving the problem of the architecture obsolescence in two different ways: by reading and converting the Media Engine and Multivalent source code to a new language, or by recreating or converting the JVM, in case such technology is no longer available for the future architectures.

We argue that Multivalent avoids architecture obsolescence because:

1. The JVM is such an important part of the programming industry that it will be ported to new environments for the foreseeable future, possibly indefinately, without any extra efforts by the media industry.
2. If for some reason, Java does become obselete, the fact that the Java code is stored means that it will be available for translation; and due to the investments made in Java, such translation will be a problem that will be shared by many, thus causing such translation tools to become widely available.
3. As Java itself is open source, the media industry could undertake to port to new architectures between the time when 1 (above) is still valid and if 2 (above) were to occur.

[1] http://www.is-research.de/info/vmlanguages/

At this point, future recipients will have a complete description of not only the rules for encoding and decoding a file format, but also the complete code that allows the understanding of how to render and implement all the features of such format, and the interactivity within them. (As outlined below: the source code of media engines provide a standardised view on parsing and rendering different file formats. This handles, for example, the common occurrence of file formats that are not compliant to the file format specifications, but are normally interpreted correctly by current access software.) Access to the essence is also provided, so that the data can be transcoded, and then manipulated to be used in other ways (eg re-editing). By executing Media engines within the JVM, they will be able to interact directly with the data, straight from their current operating system and user interfaces. This has several advantages over the most commonly proposed approaches of migrating file formats and of emulating the whole computer architecture [2].

Multivalent media engines in the OAIS perspective

The OAIS (Open Archival Information System) [3] is a widely known and used reference model for digital preservation, and as such it has a wide, generic description of all its components.

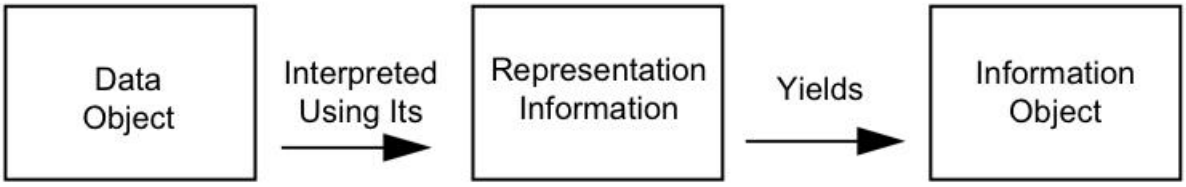

Figure 1: OAIS information model (Figure 2.2 from OAIS)

Central to the OAIS concepts is the object of the archival process: the information. The OAIS information model assumes that a data object alone is not sufficient to describe information; it needs to be interpreted through some representation information, that will in turn allow the recreation and interpretation of the information object, as illustrated in Figure 1 (from the OAIS standard).

While the OAIS definition for representation information is very open (as it includes all the information needed to interpret a document that is not explicitly expressed in the data object) and can bring to a very extensive discussion, in the scope of this report we are focusing on the subset of the definition most relevant to our developments.

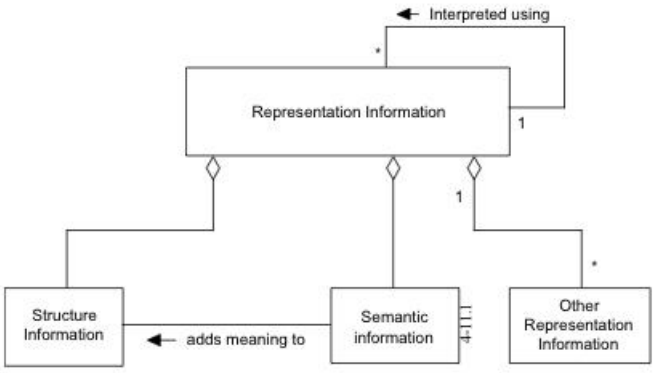

Figure 2: Representation Information Object (Figure 4.11 on OAIS)

Representation information can include for example the file formats specifications and encodings, and, on a higher level, the semantic information needed to have a correct interpretation of the information object. This is illustrated in Figure 2: (also from OAIS, paragraph 4.2.1.3).

In the scope of these definitions, Media engines have the role to capture and interpret the data structure information, allowing correct presentation of the document format to the end user and internally to the archive.

Overall in the model, media engines can have a role that is central to the interpretation and rendering of the information, and can be seen as an implementation of different functions and roles:

- **Preservable access software**, allowing users to access the information by parsing, rendering and presenting the data (including interactivity) in an obsolete format;
- **Representation rendering software**: allowing users of the system to parse and render the representation information (for example, format specifications) stored in a specific data format. In addition: it allows the software used for quality assurance during the ingest phase to be parsed, for example to implement file identification and validation.
- **Representation information**: a Media Engine's source code on its own can be regarded as part of the representation information for a supported format. A technical specification of a file format alone is often not a sufficient description in order to recreate the original rendering and features, and often lacks fundamental parts required for correct implementation. The source code of media engines can provide a standardised view on parsing and rendering different file formats. This handles, for example, rendering of files that are not compliant to the file format specifications, but are normally interpreted correctly by current access software.

Media engines present some advantages compared to other complex software solutions, tailored for present day use: the representation information expressed by the engine source code is cleanly separated from other components and features that are more involved with the operating system integration and editing functionality; this way the source code for decoding the information is focused and much shorter and simpler to comprehend. Such media engines also include a precise reference or implementation of the embedded and required format and software libraries. This way there is a lower risk of not preserving some of the background representation information, as there is a clear specification of the JVM, source code for all the supporting system libraries, across platforms, and thus can represent full, understandable (structural) representation information and executable access software in the longer term[2].

Thanks to the fact that Multivalent media engines will not require any format conversion for supported media types, this will allow direct access to the original bitstream, maintaining the fixity of the data and thus giving a higher level of guarantee of its authenticity and integrity. This will allow, for example, to verify digital signatures used to prove origin and authenticity, applied directly to the original object, without the need of the signature refresh when applying file format conversion.

Leaf media format support and library level emulation

Media support, as in general new media engines, can be quite challenging to write, and require substantial resources if developed independently from the start. For this reason, it is auspicious to be able to reuse existing Java libraries where any exist. Although there are

[2]See also the OAIS standard paragraphs 2.2.1, 4.2.1.3.1, 4.2.1.3.2, 4.2.1.4.1, 5.2

Java libraries for dealing with a great number of file formats, in some situations it is desirable to have direct access to a native library (C or other compiled language) as they can in such cases provide a better, or updated, support for such formats.

The issue once again is that this would define some external dependency (Java Native Interface (JNI) or some other method) that is machine architecture dependent and would break the Universal Virtual Computer (UVC) principle, and should be avoided in the scope of digital preservation. Furthermore, native libraries are not integrated with the rest of Multivalent architecture to the tree level. Still, they would be valuable in order to extend the support for media that is normally interpreted at the “leaf” level, that is, media that is no longer broken into subcomponents in the Multivalent architecture (as for example is the case with images).

The library emulation approach we are proposing would improve the sustainability of the Multivalent approach to extend to a variety of different formats quite quickly, without losing the fundamental feature of execution only code that runs inside of the JVM (avoiding the call to external, native methods). The different approaches to media format support, together with an existing approach we are proposing for the first time in the context of digital preservation, are illustrated in the following paragraphs, with their pros and cons.

Conventional approaches

Conversion

The most conventional approach to the problem of file format variety is probably that of data format migration through conversion. Although this can be an effective technique from the point of view of the reduced requirement of resources (conversion software is widespread), it introduces a series of problems, including loss of authenticity, possible loss of data, and additional requirement of storage space.
Pros: Fast to implement using external software, simplifies the problem and can offer a temporary fix, leverages on existing open source software, can be relatively trusted in the case of well formed media (video and audio).
Cons: Breaks authenticity and integrity, breaks the Multivalent paradigm of parsing the original file format, can introduce conversion errors and data loss, etc.

Wrapper / bindings to a C or native library

This task is simpler than that of porting of a library to Java, but by introducing native components in Multivalent, it breaks the principle of architecture independence. A software library would need to be ported to any current architecture, and future architecture will break the Media Engine using such library
Pros: Fast development, simpler to keep up to date with the evolution of a library, reuse of available code
Cons: Breaks the UVM principle, less portable and future proof, very loose integration with the framework, little added value

Porting existing Open Source code

Porting existing libraries or software to the Java language, and integrating them into Multivalent, may be a solution in some cases, but it is still time consuming, and manual. On one side, existing libraries are a good reference when trying to understand specifications; but they sometimes introduce unnecessary complexity.

Creating new media engines from scratch

The creation of a totally new Media Engine for a media format can be expensive. On the other hand, in situations where there can be a tight integration in the Multivalent architecture, as in the case of document formats, it has the advantage of implementing all the features of the framework.

Existing Multivalent node types and media types can be reused, thus reducing the amount of code that needs to be implemented, and giving the complete set of features provided to Multivalent documents.
Pros: Very tight integration in Multivalent architecture, full feature set, complete control over code quality.
Cons: Expensive development for new media.

Native library emulation

The library level emulation approach is based on the concepts developed, for different purposes, by two recent projects: the Nested Virtual Machine [4] and Cybil [5]. Since the approach taken by Nested Virtual Machine is closer to the set of problems we are trying to solve, we will focus on that project in our description.

The use of this method in the context of Digital Preservation is, to the best of our knowledge, novel, and we have reason to believe it may bring some real benefits to the creation of new media engines, for certain audiovisual formats required in PrestoPRIME scenarios.

The objective of the Nested Virtual machine was to enable the safe execution of native programming libraries through emulation in the sandbox of the JVM; for Cybil, a very similar approach was implemented in order to enable the use of different programming languages for creating applications for mobile devices (especially games for the Java Micro Edition virtual machine common on mobile phones).

The basic idea, common to both projects, is that by compiling a C library into a MIPS binary, using a specially patched version of the GCC compilers, one obtains a binary executable or library that can then be converted to Java bytecode or source code. Once the conversion is done, the Java class is executed directly in the JVM, producing the same outputs as the original C library.

Although the native library emulation approach is close to the machine emulation approach, in this case it's only the MIPS microprocessor instruction set that is emulated, and the software library can be included and used directly by the Java software, rather than being executed in a separate, isolated process.

There is even an implementation in Java of the Unix runtime, so that the library system calls can be run directly inside the JVM. This allows the integration of the library directly into existing Java code, as opposed to the emulation of the whole computer architecture. The Nested VM is an implementation of such an approach that has already been used to convert a set of popular libraries and executables that are written in different languages (including libfreetype, libjpeg, the SQLite database, and different scientific applications) to Java executable code.

Even if the approach was created for a different objective, we believe that it can be applied successfully to the Multivalent in the scope of digital preservation: this will enable the use of C libraries directly in Java, avoiding the problems that are present when using native wrappers.

A library, using the same build process and source code, can be directly translated to Java, so there is little human intervention needed; this way the library could be kept up to date even with an automatic process, as the source code improves and bugs are fixed. The result of the conversion is pure Java code that can be executed by any JVM across platforms and machine architectures. In the case of loose integration, this would be a good alternative to the wrapper/binding approach, as it would offer the same set of features without leaving the JVM, maintaining the Universal Virtual Machine paradigm. The library can then be embedded directly into any Media Engine to extend its support of a data type.

Even more interestingly, the compiled code is bug to bug compatible to the original library, a feature that would enable a great level of fidelity in emulating not only the correct behaviour of a certain software, but also its defects. This is important for fidelity, as sometimes documents and files are created keeping in mind the bugs that were present in the mainstream software used to decode them. (For example, in Excel file formats, a bug in date calculations (coming from Lotus 123) has been reintroduced in the new latest version of the file format (OOXML) in order to maintain consistency with the files created that far. In such cases, 'correct' parsing according to specifications would introduce errors.)

There is of course a negative side to this approach. Firstly, the number of environments that would need to be emulated is likely to be too large, to justify the effort required. Secondly, the emulation of the MIPS instruction set comes at the cost of performance loss (roughly estimated around three to ten times the speed of the native version [4] and see also the NestedVM website: http://nestedvm.ibex.org); but we think this could be a reasonable expense in situations where the cost of developing a new code would exceed the value of the faster execution and the full implementation (as for example, for rare media types). The performance problem also depends on the use case scenarios. In addition, the problem may reduce in the longer term as technologies improve (eg Moore's Law) and become less relevant.

Pros: Large number of libraries that would need emulating. Fast development, simpler to keep up to date with the evolution of a library, reuse of available code; automatic conversion to Java bytecode, ability to run as pure Java. Can compile to Java bytecode any language that compiles on the GCC compiler platform (C, Pascal, Fortran). Bug to bug compatible with the original. Minimal human intervention required in the translation to Java
Cons: Porting to the MIPS architecture, where not direct (ie when software needs consistent changes in order to work in the MIPS architecture and without a user interface; where changes to code or makefiles are required, and where inter-library dependencies are required) can be problematic. Technical limitations such as those described on the NestedVM website and in [4]. (These include: a limit to the size of a single method in Java, and all the work being based on an older version of the GCC compiler suite which may limit the capability of porting some libraries. There are workarounds for such cases but not all are implemented.) Bug-to-bug compatible with original. Not as integrated as a complete Java media engine. Depending on the exact conversion, this may sometimes be slower than the original binary.

3.2 IRODS

The underlying storage technology for our proposed solution, is based on the use of the integrated Rule-Oriented Data System (iRODS), which implements a policy-based approach to distributed data management. Within this context, the Multivalent preservation model is used as a presentation and manipulation tool. Using the Multivalent system will permit an infrastructure independent method of viewing the video data retained in the preservation environment. The video stream and the Multivalent software are archived as part of the iRODS collection and migrated on to new storage technologies as and when available. By differentiating between different phases of the data life cycle based upon the evolution of data management policies, the infrastructure can be tuned to support data publication, data sharing, data analysis, and data preservation of video. It is possible to build a generic data management infrastructure that can evolve to meet the management requirements for different use cases, spanning different communities (broadcast / digital cinema).

The transformation of preservation policies into computer actionable rules is the essential capability that is needed to manage data collections that will aggregate hundreds of petabytes of data, as required for video files. The proposed iRODS based architecture makes it possible to automate required administrative functions (such as distribution, retention, disposition, replication, synchronization), enforce management policies (such as formation of AIPs, generation of audit trails, creation of error reports), and validate assessment criteria (such as integrity, authenticity, chain of custody, and trustworthiness) for broadcast and video.

The policies that are used for preservation form a continuum with earlier stages of the data life cycle. The mechanisms to manipulate the data remain the same through the data life cycle, but the management policies evolve as the user community broadens, as is inevitable. The policies required by the PrestoPRIME prototype are broadened to meet data publication expectations by broadcast and digital cinema, and then broadened to meet preservation requirements for future use. Preservation policies can be interpreted as the most stringent data management requirements, because they need to ensure that the required context (representation information) is available for use by future users.

Through the integrated system, all phases of the data life cycle can be controlled. The end goal is a reference collection that can be used to support business requirements, or serve as a resource against which future business requirements can be compared, or serve as a knowledge base. For the reference collection to be trusted, the preservation environment needs to document the policies under which the reference collection was managed, the procedures that were applied, and the assessment criteria that were used to validate compliance. The iRODS data grid provides an explicit characterization of polices as computer-actionable rules, of procedures as computer-executable micro-services, and of assessment criteria as queries on state information and parsing of audit trails. It is now possible to build a data management system that manages all phases of the data life cycle.

Infrastructure Independence

The extraction of video records from the creation environment and their import into the preservation environment is an example of infrastructure independence. The archivist controls the properties of the video stream within the preservation environment, including the policies and procedures that govern the management of the records.

The concept of infrastructure independence is important when building a preservation environment that distributes video data across multiple types of storage systems and across the federation of multiple data grids, required for digital broadcasting and cinema. Within the iRODS data grid, the policies govern the management of video data residing in multiple administration domains (for example, across different broadcast enterprises or different digital cinema companies). The policies can be enforced on top of the local administration policies of the individual enterprises. Examples include replication of data across multiple storage systems, association of descriptive and provenance metadata with each record, and uniform creation of AIPs for all records independently of where they are stored.

Example Policies

The iRODS data grid framework will be instrumented automatically to invoke policies associated with specific data manipulation operations. Examples include the application of colour mapping algorithms to suite different ambient viewing environments. Multivalent policies to be invoked during the ingest and maintenance of the audiovisual archive will include:

- **Identify**: Given a bitstream, identify the file-type so that it can be interpreted by other services or tools
- **Validate**: Given a bitstream of a given file-type, is it correctly formed? We need to validate a collection of such data (video) objects (that is, the video essence) against errors during transmission or bit rot.
- **Metadata**: Given a video object, what are its metadata? Examples of metadata include image width and height, length of audio, compression method
- **Component**: given a video object, what other files/media/documents does it contain or embed? We want to access images, video, and other media in compound objects; and validate or display embedded media
- **Parse**: Given a document, parse the file format and build a runtime presentation
- **Display**: Given a video object, build a runtime representation that can be meaningfully displayed or played.
- **Transcoding**: Given a video object, obtain the digital essence to be transcoded to a new digital format, for example, for the purpose of re-editing.

More general digital preservation policies include information such as:

- What resource pool(s) (eg which nodes in the iRODS distributed data managment system) to use when storing an object in a collection? This might differ based on size, type, user and/or metadata .
- How many replicas are needed? How are they distributed? This is decided by (for example) location, resource-type and/or vendor.
- What metadata needs to be extracted? How should the extraction be done.
- Who needs to be notified, and how? Eg e-mail, twitter, Xmessage
- What pre-publication processing needs to be done? For example: anonymization, integrity checks and format conversion.
- What metadata schemata are needed for context-sensitive discovery? For example: Dublin Core, METS .
- Who can access data? What can they access? ie group/role-based access
- What pre-transfer conversions are needed?
- What additional material to show? For example: associated digital objects, metadata, URLs

- What collections, sub-collections are to be preserved?
- How long should they be preserved for?
- What solutions should be used to address technological obsolescence? For example: migrate hardware, migrate formats, migrate applications.
- Archival information Package (AIP) details : What is in it? Is the context preserved? How should and AIP be repurposed?
- Who can curate this collection? What are the hand-over policies?

3.3 Integrated Preservation Environment

The use of the Multivalent browser technology is based on four preservation perspectives: **extraction** of records from their creation environment (identify, validate, metadata); **migration** of the video data onto new storage infrastructures; application of **preservation policies** for particular business use cases (e.g. digital cinema); and **validation** of the preservation policies. A full set of preservation policies for digital cinema and broadcast industries is currently being developed as part of the project (see above for possible examples). For PrestoPRIME, the Multivalent browser technology (media engine) provides the means to parse legacy video data formats using portable parser technology written in Java.

This approach also enables the integration of digital library annotation services with preservation. Annotations can be applied to the records, but are kept as separate metadata associated with each video object (video essence as an OAIS data object, see above). The annotations are to be linked to the video object without modifying the video object. An implication is that the annotations will be associated with the display of the video object, and can be mapped to any display choice. In addition, other ancillary files, relating to each video object, such as scripts and artists sketches, can be stored, presented and manipulated within this environment.

The primary components are based on the application of the iRODS data grid, the associated metadata catalogue, and the Multivalent preservation technology to present or manipulate the objects through rules executed in the iRODS system. Archival services need to create archival state information that is mapped to the logical name space. In the case of the iRODS data grid, the name spaces include operations for micro-services; rules; and persistent state information. These name spaces enable the addition of new management policies, new preservation capabilities, and new persistent state information without destroying the ability to execute previous management policies.

Thus, using the iRODS rules and services to integrate collections, Multivalent is the tool to interpret digital entities for presentation and manipulation. The use of Multivalent is as a presentation tool that can interpret digital entities, independently of the storage or software infrastructure. As such a “presentation tool”, Multivalent can both render (view) and characterize the set of operations that can be performed on the archived digital objects. These sets of operations can be mapped to future applications (emulated). It is also possible to use the Multivalent system to extend the set of operations and include new capabilities not thought of in the original application.

The current deliverable specifically discusses the ability of the Multivalent preservation architecture to manipulate the encoding format of a digital entity, across audiovisual formats. For a given data type, a media adaptor is built for the Multivalent browser. The Multivalent technology (Java program) and the media adaptor are archived, along with whatever is needed to preserve the JVM into the future. Emulation then consists of supporting the original operations for manipulating the digital entity. Migration consists of porting the JVM to new technology as needed. The digital entity remains unchanged, while making it possible to apply new operations that become available in new versions of the Multivalent preservation architecture. It is anticipated that the porting of the JVM will be done for all major platforms by the software industry, because of the large amount of Java code that is currently relied upon by many other industries.

3.4 Format identification

Three automatic file format identification methods are proposed:

Multivalent: Multivalent media engines encompass a file identification process. A media engine can examine a file and determine if it is of a type it supports. This will be the first approach applied when identifying files. It will be the most precise mechanism, with the greatest understanding of those formats that are part of Multivalent.

DROID: The DROID Java package implements an algorithm for using the PRONOM database to identify files. It uses a description format automatically generated from PRONOM descriptions, and can check for updates to the central description files automatically, downloading newer versions. DROID provides a stand-alone program, as well as an API for Java programs to use[3]. This will be the second approach applied when identifying files; it will be used when the Multivalent file identification cannot identify the file type. This is an international standard that continues to be expanded and improved. When we require additional file types that can be integrated into the PRONOM database, we will submit them to the PRONOM project, improving and standardizing this extended knowledge.

The POSIX file(1) command: The POSIX (Unix) file(1) command uses the "/etc/magic" database. The source for the program is available along with the magic files themselves, at the URL noted above This source exists as a stand-alone program, written in C. As of yet there is no Java library, but translation of the algorithms into Java is possible.

This will be the third approach applied when identifying files; it will be used when the DROID file identification cannot identify the file type. In the case that we cannot add new file types to DROID (for technical reasons, or if they choose not to accept a format) we can add these to the file(1) database, which has a very open policy for properly-formed submissions.

[3]See:
http://www.nationalarchives.gov.uk/aboutapps/fileformat/pdf/automatic_format_identification.pdf

4 Audiovisual Formats to be included in the Multivalent Framework.

4.1 Existing Audiovisual processing technologies

Timed media, that is audio and video content, is usually encoded using a rather complicated structure that spans across classes of wrappers, container formats, media encodings, and codecs (encodings implementation) with their own particular implementation issues. This situation makes the support of a wide variety of media types expensive.

On the other hand, the fact that video can be conveniently handled as an end media type in Multivalent, opens the possibility of using existing C libraries under library emulation in order to expand to a wider set of formats.

To test the extension of Multivalent to the domain of moving images, and as a proof of concept, two simple media engines, one for the C OGG Theora and Vorbis and a second for MPEG (1, 2 and 4) (both using existing libraries), have been successfully implemented.

The following is a list of different open source software codecs that could be used for the future extension of video in Multivalent.

Pure Java codecs

- JFFMPEG (*implemented in current Multivalent demo*):
  http://jffmpeg.sourceforge.net:
  This is a real porting to pure Java port of the FFMPEG, but seems to have recently limited development: “Jffmpeg is a plugin that allows the playback of a number of common audio and video formats. It is based around a Java port of parts of the FFMPEG project (see 'Other native codecs and software' below), supporting a number of codecs in pure Java code. Where codecs have not yet been ported, a JNI wrapper allows calls directly into the full FFMPEG code.” According to the documentation, it still does offer reasonable codec support, see: http://jffmpeg.sourceforge.net/formats.html
  Formats Supported: MPEG(1,2,4); AVI, DIVX, MJPEG
- Cortado OGG Theora/Vorbis: http://www.flumotion.net/cortado/
  (*implemented in current Multivalent demo*)
  The cortado applet, in the fork now developed by the WikiMedia foundation.
  Formats Supported: OGG, VOB; MJPEG
- MediaFrame MPEG4: http://mediaframe.org/
  Open Source decoder of MPEG1 and MPEG4 in pure java. “Video: MPEG4 Simple Profile (I and P-type frames, video packets, up to 4 motion vectors per macroblock); Audio: AAC, MPEG 1/2 Layer 3 (MP3)
- Sun’s Open Video:
  http://blogs.sun.com/openmediacommons/entry/oms_video_a_project_of
  A project for an open video codec by Sun; not very promising as based on old technology
- JavaFX:
  http://javafx.com/
  The new Java FX language libraries, developed by Sun, and available, but in a closed source release, include support for decoding Flash Video.

Java wrappers to existing C libraries

- http://code.google.com/p/gstreamer-java/ : a wrapper to the Gstreamer framework
- http://fmj-sf.net/ : “FMJ is an open-source project with the goal of providing an alternative to Java Media Framework (JMF), while remaining API-compatible with JMF. It aims to produce a single API/Framework which can be used to capture, playback, process, and stream media across multiple platforms.” It uses the native ffmpeg library for doing the decoding.
- http://fobs.sourceforge.net/index.html this is again a wrapper to FFMPEG, and Java Media Framework
- http://wiki.videolan.org/Java_bindings Java bindings for VLC, the home page is currently unavailable.

Other native codecs and software

- The native C library FFMPEG is available at: http://sourceforge.net/projects/ffmpeg/ This is probably the most used library in other Open Source codecs; it includes the libav codec library. Includes support for a wide variety of video and audio encodings.
- http://www.tele.ucl.ac.be/PROJECTS/OPENJPEG/ An open implementation of the JPEG 2000 standard, includes support for the MJPEG2000 (Motion Jpeg 2000) standard for video.
- VLC: A video player that supports a wide variety of formats. It uses different libraries to decode the video, including the FFMPEG library and others: see http://www.videolan.org/project/
- http://www.mplayerhq.hu The MPlayer video player natively supports a wide variety of audio and video formats. It also uses FFMPEG for most of its format support.
- http://libmpeg2.sourceforge.net/ MPEG-2 video stream decoder
- http://liba52.sourceforge.net/ ATSC A/52 stream decoder
- http://www.xvid.org/ A popular open source codec implementation for MPEG 4 AVC
- Other references: http://en.wikipedia.org/wiki/Open_source_codecs_and_containers

Media engine implementation status

As part of the experimentation on introducing audiovisual formats to Multivalent, two video media engines have been developed, firstly for Theora and Vorbis formats and secondly for all formats included in the Java Media Framework (and JFFMPEG plug-in), most importantly the MPEG-2 compression format.

In both cases, the video is decoded and rendered as a Multivalent leaf that is rendered to the screen. This allows for most of Multivalent’s features to work, for example lenses and static (non timed) annotations are working. Further support for the timed characteristics could be improved in a general way to be applied to different timed media types. The HTML engine has also been modified to allow a simple embedding of the video to the HTML web pages as specified in HTML5.

The next steps in the implementation are to write a Java codecs for MXF-wrapped JPEG2000 and MXF-wrapped D10. Also, transcoding facilities must also be written in Java.

4.2 File Formats

MPEG-1

Common Names: MPEG-1
MIME types: `audio/mpeg, video/mpeg`
File extensions: `.mpg, .mpeg, .mp1, .m1v, .mpv`
Reference web sites: http://www.chiariglione.org/mpeg/
Developed by: ISO, IEC
Type of format: audio, video, container
Extended from: JPEG, H.261
Extended to: MPEG-2
Standard(s): ISO/IEC 11172
Bandwidth: Medium (up to 1.5Mbits/sec)
Other specifications (typical):

1.25Mbits/sec video 352 x 240 x 30Hz
250Kbits/sec audio (two channels)
Non-interlaced video
Optimized for CDrom

Patents: Due to its age, many of the patents on the technology have expired. However, a full MPEG-1 decoder and encoder can not be implemented royalty free since there are companies that require patent fees for implementations of MPEG-1 Layer 3 Audio however. (source: Wikipedia: MPEG-1 – Dec 09)

General Notes: 'MPEG-1 is the exclusive video and audio format used on Video CD (VCD), the first consumer digital video format, and still a very popular format around the world.'
'The widespread popularity of MPEG-2 with broadcasters means MPEG-1 is playable by most digital cable and satellite set-top boxes, and digital disc and tape players, due to backwards compatibility.'(Wikipedia: MPEG-1 – Dec 09)

Implementation in Multivalent: This format is playable in the current PrestoPRIME Multivalent demo, as it is one of the formats covered by the JFFMPEG plug-in to the Java Media Framework.

MPEG-2

Common Names: MPEG-2
(MPEG-2/Video is formally known as ISO/IEC 13818-2 and as ITU-T Rec. H.262)
MIME types: `audio/mpeg, video/mpeg`
File extensions: `.mpg, .mpeg, .mp2, .mpv`
Reference web sites: http://www.chiariglione.org/mpeg/
Developed by: ISO, IEC
Type of format: audio, video, container
Extended from: MPEG-1
Standard(s): (ISO/IEC 13818)
Bandwidth: Higher (up to 40Mbits/sec)
Other specifications:

Up to 5 audio channels (i.e. surround sound)
Wider range of frame sizes than MPEG-1 (including HDTV)
Can deal with interlaced video

Patents: According to the MPEG-LA Licensing Agreement MPEG-LA, **any** use of MPEG-2 technology is subject to royalties.

- Encoders are subject to a royalty of $2.50 per unit.
- Decoders are subject to a royalty of $2.50 per unit.
- Royalty-based sales of encoders and decoders are subject to different rules and $2.50 per unit.
- Also, any packaged medium (DVDs/Data Streams) is subject to licence fees according to length of recording/broadcast.

In the case of free software such as VLC media player (which uses the ffmpeg library) and in which the software is not sold, the end-user bears the royalty.

(source: Wikipedia: MPEG-2 – Dec 09)

General Notes: MPEG-2 was developed to deal with the problems in MPEG-1 and includes support for interlaced video, the format used by analog broadcast TV systems
(source: Wikipedia: MPEG-2 – Dec 09)

Implementation in Multivalent: This format is playable in the current PrestoPRIME Multivalent demo, as it is one of the formats covered by the JFFMPEG plug-in to the Java Media Framework. *(Although not all MPEG-2 files play – this is a problem that needs to be addressed)*

H.264/MPEG-4 AVC

Common Names: MPEG-4, H264
MIME types: `audio/mpeg, video/mpeg`
File extensions: `.mpg, .mpeg, .mp1, .mp2, .mp3, .m1v, .m1a, .m2a, .mpa, .mpv`
Reference web sites:
Developed by: ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG)
Type of format: audio, video, container
Extended from: MPEG-2, H.263, or MPEG-4 Part 2
Standard(s): The H.264 "family of standards"
Bandwidth: Aims at half or less the bit rate of MPEG-2, H.263, or MPEG-4 Part 2), without increasing the complexity of design so much that it would be impractical or excessively expensive to implement. MPEG-4 is efficient across a variety of bit-rates ranging from a few kilobits per second to tens of megabits per second.
Other specifications:
It allows the transmission of synthetic music and sound effects at very low bit rates (from 0.01 to 10 kbps)

Patents: 'Consensus in discussions is that the allowable use depends on the laws of local jurisdictions. If operating or shipping a product in a country or group of countries where none of the patents covering H.264 apply, then using, for example, an LGPL implementation of the codec is not a problem: There is no conflict between the software license and the (non-existent) patent license. Conversely, shipping (not necessarily implementing) a product in the U.S. which includes an LGPL H.264 decoder/encoder would be in violation of the software license of the codec implementation. In simple terms, the LGPL and GPL licenses require that any rights held in conjunction with distributing the code also apply to anyone receiving the code, and no further restrictions are put on distribution or use. If there is a requirement for a patent license to be sought, this is a clear

violation of both the GPL and LGPL terms. Thus, the right to distribute patent-encumbered code under those licenses as part of the product is revoked per the terms of the GPL and LGPL. It should be realized that the party who would enforce any such breach of copyright would be the people who hold copyright: its writers, whereby any suit on a breach of that clause would have to argue that there exist valid, applicable patents that apply to the capabilities GPL licenced code, a stance copyright holders have not taken.'
(source: Wikipedia: H264/MPEG-4 – Dec 09)

General Notes:
(Source Wikipedia: H264/MPEG-4 – Dec 09)
'H.264 is most popular for its use on Blu-ray Disc, HD DVD and videos from the iTunes Store.'
'The intent of the H.264/AVC project was to create a standard capable of providing good video quality at substantially lower bit rates than previous standards (e.g. half or less the bit rate of MPEG-2, H.263, or MPEG-4 Part 2), without increasing the complexity of design so much that it would be impractical or excessively expensive to implement. An additional goal was to provide enough flexibility to allow the standard to be applied to a wide variety of applications on a wide variety of networks and systems, including low and high bit rates, low and high resolution video, broadcast, DVD storage, RTP/IP packet networks, and ITU-T multimedia telephony systems.'

***Implementation in Multivalent**: Untested: The MPEG-4 format is playable in the current PrestoPRIME Multivalent demo, as it is one of the formats covered by the JFFMPEG plug-in to the Java Media Framework, but it has not been fully tested. It is likely that* JFFMPEG supports only the basic MPEG4 profile, and does not support the H264 profile.

MXF-Wrapped JPEG-2000 with Uncompressed Audio

Common Names: JPEG 2000 , MXF-JP2k, MXJP2, MXF_OP1a_JP2_LL (lossless), MXF_OP1a_JP2_LSY (lossy)
MIME types: image/jp2, image/jpx,
File extensions: .jp2, .jpx
Extended from: Motion JPEG
Standard(s): ISO/IEC 15444-1; Follows MXF Operational pattern OP1a
Reference web sites: http://www.jpeg.org/
See also: http://dcimovies.com/DCIDigitalCinemaSystemSpecv1_2.pdf
Developed by: Joint Photographic Experts Group
Specification License: Specifications available from ISO; Implementation of Part 1 is licensed for free; Part 2 requires licensing. Not Open.
Local specifications path: Specifications/ISP1/JPEG2000

General Notes:
JPEG2000 is a still image format from the JPEG group, defining a **lossy** or **lossless** compression format superior to the JPEG standard, based on wavelet compression. It is described in two parts: part 1, core, that can be licensed for free; and part 2, extended, that requires royalties.

'It specifies the use of the JPEG 2000 codec for timed sequences of images (motion sequences), possibly combined with audio, and composed into an overall presentation. It also defines a file format, based on ISO base media file format (ISO 15444-12).

Open source Java (or C) libraries for still image JPEG-2000: A pure Java implementation of the core system (part 1) is available as the reference implementation if JPEG2000, that can be freely downloaded at the JJ2000 website. Jasper is the reference C implementation of JPEG2000, also covering part 1 of the standard.

Implementation in Multivalent: High Priority; Not yet implemented.
(See also MXF below)
There is already a JPEG-2000 decoder written in Java, but tests would be required to find out if it can be used to code moving image formats. Some complexity would be due to the various profiles of JPEG2000 and the different colour spaces. Another problem is the high resolution, but the image could be decoded at a lower resolution for display purposes. There are also a few other open source C libraries which should be tested.

D10

(source: Wikipedia: D10 – Dec 09):

'SMPTE 356M Television specification for a professional video format, it is composed of MPEG-2 Video 4:2:2 I-frame only and 8 channel AES3 audio streams. These AES3 audio usually contain 24 bit PCM audio samples. It is possible to find video bit-rates of 30, 40 and 50 MBit/s.

This format is described in the document SMPTE 356M-2001, "Type D-10 Stream Specifications — MPEG-2 4:2:2P @ ML for 525/60 and 625/50".

D10 is also called IMX by Sony.'

Implementation in Multivalent: High Priority; Not yet implemented.
(See also MXF below)

4.3 Wrapper Formats

Material Exchange Format (MXF)

MXF is an open source container format for professional digital video and audio media defined by a set of SMPTE standards. It is the primary professional broadcast wrapper format, being used for example in British and US public broadcasting and in digital cinema.

This format “supports a number of different streams of coded "essence", encoded with any of a variety of codecs, together with a metadata wrapper which describes the material contained within the MXF file.” (Source: Wikipedia - “Material Exchange Format”)

Implementation in Multivalent:
(See also MXF-Wrapped-JPEG2000 above)
Wrapper technologies are not currently implemented in the Multivalent demo, but the ability to input this format, read the metadata and then process the 'essence', in the same way that other AV formats are already handled is an immediate priority. (The essence may already be in a format that is already handled by the Multivalent demo).

A number of MXF libraries exist, for example: libMXF written by the BBC to read and write to this file type. Unfortunately, no libraries are available in non-native code. A possible approach would be to convert the open source C code of libraries such as libMXF to Java.

Quicktime

(Source: Wikipedia - “QuickTime” Dec 09)

“The QuickTime (.mov) file format functions as a multimedia container file that contains one or more tracks, each of which stores a particular type of data: audio, video, effects, or text (e.g. for subtitles). Each track either contains a digitally-encoded media stream (using a specific codec) or a data reference to the media stream located in another file. Tracks are maintained in a hierarchical data structure consisting of objects called atoms. An atom can be a parent to other atoms or it can contain media or edit data, but it cannot do both.”

“The ability to contain abstract data references for the media data, and the separation of the media data from the media offsets and the track edit lists means that QuickTime is particularly suited for editing, as it is capable of importing and editing in place (without data copying).”

“Because both the MOV and MP4 containers can use the same MPEG-4 codecs, they are mostly interchangeable in a QuickTime-only environment. However, MP4, being an international standard, has more support.”

Implementation in Multivalent:
As highlighted above, wrapper technologies are not currently implemented in the Multivalent demo, in a way that the user can input this format, read the metadata and then process the 'essence', in the same way that other AV formats are already handled. In addition, the priority wrapper format to be included in the demo is MXF rather than Quicktime.

On the other hand, the ability to play Quicktime files is already included in the Java Media Framework (and JFFMPEG plug-in), but this is limited to only those that are encoded using the Cinepak codec.

AVI

Common Names: AVI
MIME types: video/avi, video/msvideo, video/x-msvideo
File extensions: .avi
Standard(s): ISO/IEC 15444-3 and in ITU-T T.802, ISO 15444-12
Reference web sites: http://www.jpeg.org/
Developed by: Microsoft
Patents: Patent encumbered
General Notes: (source: Wikipedia: AVI – Dec 09)
'AVI is a derivative of the Resource Interchange File Format (RIFF), which divides a file's data into blocks, or "chunks." Each "chunk" is identified by a FourCC tag. An AVI file takes the form of a single chunk in a RIFF formatted file, which is then subdivided into two mandatory "chunks" and one optional "chunk".
The first sub-chunk is identified by the "hdrl" tag. This sub-chunk is the file header and contains metadata about the video, such as its width, height and frame rate. The second sub-chunk is identified by the "movi" tag. This chunk contains the actual audiovisual data that make up the AVI movie. The third optional sub-chunk is identified by the "idx1" tag which indexes the offsets of the data chunks within the file.

By way of the RIFF format, the audiovisual data contained in the "movi" chunk can be encoded or decoded by software called a codec, which is an abbreviation for

(en)coder/decoder. Upon creation of the file, the codec translates between raw data and the (compressed) data format used inside the chunk. An AVI file may carry audiovisual data inside the chunks in virtually any compression scheme, including Full Frame (Uncompressed), Intel Real Time (Indeo), Cinepak, Motion JPEG, Editable MPEG, VDOWave, ClearVideo / RealVideo, QPEG, and MPEG-4 Video.'

Implementation in Multivalent: Currently being implemented.
This format is playable in the current PrestoPRIME Multivalent demo, as it is one of the formats covered by the JFFMPEG plug-in to the Java Media Framework. *(Not currently playing in Fab4, but does play in standalone JMF/JFFMPEG applications. Needs to de-bugged within Fab4 environment.)*

OGG

Common Names: Ogg video, Ogg Theora video, with Vorbis audio
MIME types: application/ogg, video/ogg, video/theora
File extensions: Ogg, ogv, oga, ogx
Registry Identifiers PRONOM: N.A.
GDFR: N.A.
Versions (Year): 1.0
Can contain encodings: Theora video, Vorbis audio, FLAC audio, SPEEX audio
Reference web sites: http://www.xiph.org/
Specification License: Patent free (OGG); or patent guaranteed on a free unrevocable license (Theora)
Implementation in Multivalent: This format is playable in the current PrestoPRIME Multivalent demo, as it is one of the formats covered by the Cortado plug-in.
Reason for inclusion: Extension of media engines to a new media type, as a proof of concept. Format of recent interest as an open solution for video in the upcoming HTML standard; availability of Java libraries for rapid prototyping. Possibility of converting to the format using open and royalty free tools. The Xiph.org foundation maintains these formats and an open source implementation. The formats are unrestricted by patents.

5 Glossary

Term	Definition
AIP	Archival Information Package: Related to OAIS. This is the concept for identifying a collection of information that can be preserved over the long term.
Bitstream	“Contiguous or non-contiguous data within a file that has meaningful common properties for preservation purposes. A bitstream cannot be transformed into a standalone file without the addition of file structure (headers, etc.) and/or reformatting the bitstream to comply with some particular file format”[6]
Bitstream encoding	An encoding that in normally used as part of a wrapper format, in a bitstream. For example the Theora video encoding, used in OGG files for the video bitstream.
Container Format (aka wrapper format)	See Wrapper Format
DROID	Digital Record Object Identification: A format identification toolDigital Record Object Identification - a software tool developed by the UK National Archives to perform automated batch identification of file formats. Designed to allow to identify the precise format of all stored digital objects, and to link that identification to a central registry of technical information about that format and its dependencies.
GCC	GNU Compiler Collection: The compiler for the GNU (free open-source operating system) operating system.
GPL	General Public Licence: A free, copyleft licence for software and other kinds of work.
iRODS	Rule-Oriented Data System: Data grid software system being developed by the San Diego Supercomputer Centre
JNI	Java Native Interface: A programming framework that allows Java code running in a JVM to call and be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages such as C, C++ and assembly.
JPEG2000	An architecture for lossless and visually lossless image compression that supports multi-resolution imaging and scalable image quality
JVM	Java Virtual Machine: A set of computer software programs and data structures that use a virtual machine model for the execution of other computer programs and scripts. The model used by a JVM accepts a form of computer

	intermediate language commonly referred to as Java bytecode.
MIME Type	Also called internet media type, is an extensible two part identifier for file formats defined originally in RFC 2046. Types and subtypes starting with 'x-' are non standard; 'vnd.' Is used for vendor specific, '.prs' is for personal. The IANA maintains the list of registered MIME types[4] and character sets.
MIPS	A reduced instruction set computer (RISC) instruction set architecture (ISA) developed by MIPS Technologies.
Multivalent	The Multivalent preservation architecture is a presentation tool that preserves the ability to manipulate the encoding format of a digital entity. For each new format type, a media adaptor is created. New standard operations can be added to the system. The system is implemented in Java, enabling the port onto new technology.
MXF	Material Exchange Format: a container format for professional digital video and audio media defined by a set of SMPTE standards.
OAIS	Reference Model for an Open Archival Information System (OAIS)
POSIX	Portable Operating System Interface [for Unix]: a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system.
Transcoding / Transformative Migration	The direct digital-to-digital conversion of one encoding to another.
UVC	Universal Virtual Computer: A virtual machine (VM) specially designed for preservation of digital objects such as held by libraries, archives and institutions alike
Wrapper format (aka Container format)	Format that encapsulates different bitstream encodings, together with metadata to describe its contents

[4] http://www.iana.org/assignments/media-types/

6 References

[1] Herr, Lauren, 2007; *The Digital Dilemma - strategic issues in archiving and accessing digital motion picture materials;* The Science and Technology Council of the Academy of Motion Picture Arts and Sciences (AMPAS).
[2] Phelps, T., Watry, P., 2005; "*A No-Compromises Architecture for Digital Document Preservation*", in Research and Advanced Technology for Digital Libraries 9th European Conference, In proceedings ECDL2005, pp. 266—277 .
[3] "*Reference Model for an Open Archival Information System (OAIS).*", 2002; CCSDS 650.0-B-1, Blue Book.
[4] Alliet, B., Megacz, A, 2004; "*Complete translation of unsafe native code to safe bytecode*", In proceedings of the 2004 workshop on Interpreters, virtual machines and emulators, Washington, D.C.
[5] Kågström, S., Grahn, H., Lundberg, L, 2007; "*Cibyl: an environment for language diversity on mobile devices*", In proceedings of the 3rd international conference on Virtual execution environments, June 13-15, 2007, San Diego, California, USA
[6] PREMIS Editorial Committee, 2008; "PREMIS Data Dictionary for Preservation Metadata, version 2.0", March 2008