

Visual and Sound Materials for UK Further and Higher Education

VSM Portal Demonstrator



Visual and Sound Materials Demonstrator Project

Appendices to the Final Report

Author(s): Leah Halliday, Rick Loup, Christine Rees, Ben Soares, Tim Stickland

Date: May 2008

Contents

Appendix 1: User Evaluation	1
Appendix 2: Content development	5
Appendix 3: Technical Documentation	7
Functional specification.....	7
Architecture	8
Database and API	13
Community Discussion Area (Forum)	15
Content Based Image Retrieval	16
MyVSM.....	17
Metadata Harvesting.....	18
Harvesting versus Distributed searching	20
Appendix 4: VSM Metadata Format	22

Appendix 1: User Evaluation

Usability testing was carried out in four institutions, two HEIs (University of Hull and University of Edinburgh) and two FEIs (St Helens College and Swansea College).

User support staff at EDINA designed the evaluation exercise and developed the resources (e.g. an observer scoring sheet, a task list, questionnaire to screen out unsuitable candidates, and instructions to all involved in the exercise). EDINA staff conducted the evaluation at the University of Edinburgh and St Helens College, while information professionals at Hull University and Swansea College conducted user evaluation at their own sites. They used the resources developed at EDINA with minor revisions as appropriate.

Nevertheless, there were some differences in the circumstances of the tests. At Edinburgh University and Swansea College, subjects were not required to login using ATHENS. They were pre-authenticated. At Edinburgh this authentication gave them permission to access all of the VSMPD target resources. At Swansea College the permission was not comprehensive but the resource for which they lacked authentication was not selected by the subjects. At St Helens College subjects were asked to bring their ATHENS credentials. They all recorded their ATHENS credentials in notebooks which they consulted when required to login, i.e. their credentials were not so familiar that the subjects had memorised them so it is not clear whether they would have had them to hand in a 'real' situation. At Hull University, subjects were not pre-authenticated. All of them logged in using ATHENS without trouble.

The way in which the test was conducted also varied. At the University of Edinburgh and St Helens College, the evaluator worked through the task list asking the subject to conduct each task in turn. At Hull University and Swansea College, the evaluator gave the task list to the subject who read the tasks as s/he conducted them.

The usability test was designed to test the interface and functionality of the VSM Portal Demonstrator (VSMPD) and how effectively users could complete a real-world task or project. To that end, each user was given a series of linked tasks to determine whether they could navigate and fulfil their requirements from start to finish, i.e. how well the different functions operate in sequence.

The results at three sites were largely similar; but users at Hull found the VSMPD easier to use than those at the other sites. It is not clear why this occurred. There are some differences in the way that the tests were conducted at different sites but nothing that is exclusive to Hull. The fact that subjects at Hull were all sufficiently familiar with their ATHENS passwords that they had no trouble logging in without assistance suggests that they are familiar with online information services. Perhaps subjects at Hull were simply more experienced users and thus were more confident than users at other sites; we cannot tell.

The evaluation resulted in a number of suggestions to improve the VSM portal demonstrator arising from the users and the observers/evaluators. These should first be considered for desirability and feasibility before being accepted for development. The suggestions, with the project's initial comments on desirability and feasibility shown in italics, are:

- The portal should indicate clearly what type of content is targeted and how it is selected; this statement should derive from a coherent collections policy

This recommendation presents no technical difficulties. We would add this statement to the new portal design suggested for future development.

- For a service, the VSM team should investigate fuzzy search functionality so that users' spelling mistakes are not terminal for the discovery process.

The Zebra database software can permit fuzzy searching. This is one of several advanced database functions we have suggested for future development.

- Presentation of results should be reviewed. If results are sorted by collection (as at present), there should be some indication of how they are sorted. The hyperlinked keywords following the collection title should be removed.

The issue of categorising results by collection requires further careful consideration. Presenting results categorised by collection may have no apparent purpose to end users, in which case negative reactions can be expected; these should not be dismissed, but may be outweighed by other factors:

1. *We know that end users have a low awareness of some content made available by JISC, and other services in the wider academic community; there may be significant benefits in making users aware of the existence of these collections.*
2. *One of the main motivations for content providers to commit resources to working with the VSM project in raising their profile; this gives them a legitimate interest in seeing that the “identity” of their content is maintained within the result set.*
3. *A merged result set would tend to give prominence to collections that generate large numbers of results. This is likely to reflect the size of the collection more than the quality of results, and relevance ranking will only partially mitigate this effect. A result set categorized by collection permits users to find results that are few in number, but which may be the most suitable for their needs.*

If categorization by collection is retained, we can indicate that collections are sorted by the relevancy score of the results (collection with more results are not ranked higher, as we think this is a poor indication of usefulness). The hyperlinked keywords were included in an attempt to summarize the content of the results offered by each collection. This feature has not fulfilled its purpose for most users, but in its absence there would be no indication for users of the nature of the records. It is interesting to note that an almost identical feature in the GetRef project (a federated search of bibliographic databases) was successful. We may be able to find another mechanism that describes the content and which may be more successful; if not, this feature will be removed.

An alternative route would be to present the user with a merged and ranked result set, whilst giving a discrete (marginal) indication of the constituent collections. These could also provide links to the results from that specific collection. Whilst this would probably technically have to be a new search, it could be presented as a "filtering" of the initial broader search.

- The VSM team should consider carefully whether collections lacking thumbnails should be included or excluded as targets for the VSMPD.

This will be considered in any future development, but it was not considered appropriate, in the current phase, to assert a new condition for content providers who had already agreed to work with us. As an alternative, we may be able to negotiate access to images that would permit us to generate and host thumbnails for a collection.

- The VSM team should consider presenting a search box at the top of every screen within the service.

This does not fit well with the current design of the portal demonstrator, but will be considered for the new portal design suggested for future development.

- The VSM team should consider carefully whether to include as VSMP targets collections which do not offer persistent URLs.

This will be considered in any future development, but it was not considered appropriate, in the current phase, to assert a new condition for content providers who had already agreed to work with us.

Lack of persistent URLs often (though not always) means a click-through to the actual item is not possible. Consideration of user frustration should be made against such inclusions.

- The VSM team should consider making the thumbnail image link to the correct record within the interface of the service where it resides.

This risks heightening the impression that content resides within the VSMPD service but as users have this impression anyway, the VSMPD will have to address it, i.e. to make it clear to users when they leave the VSMPD and go to another service. Steps have been taken to ensure that users are fully aware of when they “exit” the VSM Portal, by inserting a warning message (users may disable this if/when it becomes irritating). Making the thumbnail into an external link will be considered for the new portal design suggested for future development.

Such a feature is likely to by-pass user added comments within the VSMPD thus reducing the usefulness of this existing feature.

- The VSM team may usefully consider whether to provide users with the option to create their own login details for 'My VSM'. This removes its dependence on ATHENS, making it akin to services such as YouTube, Flickr etc. and allows users to select their own, easily memorable passwords.

We do not consider that this is appropriate. We understand that the need to log in may seem onerous, but this it is essential for functions that need to recognize users across separate sessions. Access to services in the JISC IE and beyond is mediated via a JISC mandated national framework (currently Athens and the UK Access Management Federation), with login mechanisms often provided by end users' institutions and integrated with local services (institutional portals, mail servers, etc.) We do not think that a JISC funded project or service should deviate from this policy, nor do we accept that it is genuinely easier for a user to register and create a separate account for a specific service. The observation may be an artefact of conducting usability testing outside of end users' normal working environment, which should be configured to default to appropriate authentication points.

- Users should have a toggle option that allows them to restrict their search so that they retrieve only content that they can see free of charge.

This is problematic. The goal is presumably to determine which content is free at the point of access, rather than free of any charge (such as subscriptions). To achieve this we would need to be aware of the subscriptions applicable to all end users, and this information is not available to us. We could differentiate between content that is freely available to all, and that which require a subscription, and allow users to filter search results on this basis; however a search run on that basis would exclude results from collections such as Film & Sound Online and Education Image Gallery, even though these are widely and freely available to end users. Though we accept that inaccessible search results are a real irritation, we concluded that it is more important to avoid the implication that users cannot access resources for which their institutions have paid a subscription.

- Once a user has created a folder or selected a specific folder within a session, that folder should remain open unless the user chooses otherwise.

This modification has been made to the portal.

- The 'default' folder should be renamed, e.g. 'general' or 'miscellaneous'.

The default folder has been renamed “General”.

- The VSM team should consider the merits of opening a new window for every result that a user selects (perhaps with a top limit).

User interfaces that open many windows are usually very badly received, and this design (dependent on a graphic user interface) is poor for accessibility. We think that such a design would generate a higher proportion of negative comments than the current design.

- A VSMP service could usefully present users with a style and examples of how to reference each media type in an academic essay or paper.

We will consider this in future development, though we are not sure that the VSM team are best placed to instruct end users regarding citation practices.

- The delete function in MyVSM should be made more transparent and straightforward, i.e. it should be possible to delete a record with a single keystroke.

We have simplified the deletion process, though it requires two clicks rather than one.

- Labels on functions in the VSMPD should be reviewed; plain English should be used throughout.
We have relabelled several menu items and links.
- The VSM team should consider how it might inform users of the limitations of services, e.g. that the quality and speed of services accessed through the VSMPD varies.
This will be considered for the new portal design suggested for future development.
- The VSM team should consider how to inform users that use of content accessed through the portal depends on the infrastructure at their host institution.
This will be considered for the new portal design suggested for future development.
- When the VSM team has an agreed collections policy and has identified a critical mass of collections to be included as VSMP Service targets, the team should implement a feature that allows users to search for content based on the terms of use (e.g. to include in a search only content that is available for liberal use within an educational context).
This recommendation presents no technical difficulties. For practical reasons we envisage that we would categorize collections in fairly broad terms (e.g. personal use only; educational use; any use) rather than attempt to describe licences in detail. We have introduced this already in a descriptive role, and will enable it to be used as a search filter when we have sufficient collections in each category to make it useful.
- The VSMPD interface should be redesigned to look more like a portal and less like a search engine. It should include graphics and adopt conventions that users consider 'intuitive' due to their familiarity. Plain English should be used throughout the interface.
We envisage a new design for the portal early in any future development.

Appendix 2: Content development

Progress with the negotiations initiated for a further 19 collections is as follows:

Agreed in principle:

- 1) Wellcome Medical Photographic Library, <http://images.wellcome.ac.uk/indexplus/page/Home.html>; images (Pixus)
- 2) Image Enriched Learning in Tourism (IELIT) project, <http://www.tourismimages.org.uk>; images
- 3) Creative Archive (BFI); moving pictures
- 4) Screenonline (BFI), <http://www.screenonline.org.uk>; moving pictures
- 5) Creative Archive - Open University, <http://www.open2.net/creativearchive>; moving pictures
- 6) British Library Archival Sound Recordings (JISC Digitisation project), <http://sounds.bl.uk>; sound

Agreed in principle, but dependencies:

- 7) MIDESS multimedia institutional repository project; images, moving pictures and sound; (see below)
- 8) Creative Archive (BBC); moving pictures; BBC project stalled
- 9) British Pathe Archive, <http://www.britishpathe.com>; images and moving pictures; awaiting ITN contract renewal

Contacted and awaiting outcome of approach:

- 10) BioScience image bank (JISC); images
- 11) Courtauld Institute, <http://www.artandarchitecture.org.uk/>; images (Pixus)
- 12) English Heritage - Images of England, <http://www.imagesofengland.org.uk>; Viewfinder; Pastscape; images
- 13) Arts on Film Archive (Arts Council England), <http://artsonfilm.wmin.ac.uk/filmcollection.html>; moving images

A further 6 content providers, from the commercial, public/government, and 'free' sectors were approached, but for a variety of reason will not be included.

To approach for any continuation of the demonstrator or pilot service

The project steering group made the following recommendations:

- Clinical Skills OnLine: video clips for health-related courses, <http://www.elu.sgul.ac.uk/cso/>
- Free stock photo collections that provide more general images, such as Stock Exchange, <http://www.sxc.hu/>, and morgueFile, <http://www.morguefile.com/>.
- Commercial image collections that would be of interest to education, but are not primarily image libraries, such as the Victoria & Albert Museum, <http://www.vam.ac.uk/>, and the British Museum, <http://www.britishmuseum.org/>.
- "Images for All" at the Royal Scottish Geographical Society, <http://www.geo.ed.ac.uk/rsgs/ifa/>.
- The EU project EASAIER (Enabling Access to Sound Archives through Integration, Enrichment and Retrieval <http://www.elec.qmul.ac.uk/easaier/index.html>) that includes the RSAMD (Royal Scottish Academy of Music and Drama), which will include tools for manipulating sound files.
- The Imperial War Museum, <http://www.iwmcollections.org.uk/>.
- The UK's regional film archives, <http://www.movinghistory.ac.uk/>.

- Community-created content (in repositories), with particular reference to the 'Defining Image Access' report which looks at semantic interoperability between image repositories and the Community-Led Image Collections (CLIC) study¹.

Institutional Repositories

Part of the remit of this project was to seek to include among our targets Institutional Repositories of multimedia objects. The VSM project was funded at a time when such repositories are few and are in development rather than established services. To help us explore the issues related to repositories of multimedia objects and with a view to including one among our targets, we worked with the MIDESS team based at Leeds University. At the end of the VSM Portal Demonstrator project, MIDESS was not sufficiently well developed to offer machine-to-machine access to its content through a portal. Furthermore, a variety of legal issues surfaced by MIDESS would prohibit delivery through a portal of much of the content included in the repository. Examples included uncertainty about the nature and extent of third-party copyright permissions obtained when content was created and limitations on consent secured with regard to medical images.

Copyright status of multimedia content may be uncertain simply because ownership is complex (e.g. where an object has been photographed and the photograph has been digitised should permission have been obtained from the creator of the object, the photographer or her employer, the person commissioning the photograph, the person digitising the photograph or partly by all of these? The answer is not necessarily obvious to the person seeking to use the digitised image). Materials used for teaching within an institution may have been created by a lecturer who has since left; the permissions trail for these may not be clear to those who continue to use them and thus the question of whether they are suitable for distribution through a portal is uncertain. Where documentation about the copyright clearance process is missing, it is often difficult to clarify the status of the material. The report from WorkPackage 7 of the MIDESS project outlines the complexity of establishing the copyright status of multimedia materials that are used in hard-copy in UK HE/FE².

¹ http://www.jisc.ac.uk/uploaded_documents/CLIC_Report.pdf

² http://www.leeds.ac.uk/library/midess/IPRreport_finalversion.pdf

Appendix 3: Technical Documentation

Functional specification

This specification represents the functionality of the VSM Portal at the end of the project (January 2008).

- A single search across all fields is provided. Search terms submitted by the user are treated as CQL (<http://www.loc.gov/standards/sru/specs/cql.html>); this is a standard query language which provides powerful searches for expert users, but also works intuitively for naive users. Queries expressed in natural language, and using conventions familiar to most users from many other web interfaces (*e.g.* using quotes around terms to be treated as an exact phrase), tend to produce results that match users expectations as well as any other scheme.
- Results may be limited by media type: any combination of still image, moving image and sound).
- Results may be limited by collection: any combination of collections may be selected. Users may select or deselect all collections as well as toggling the inclusion/exclusion of individual collections.
- The ability to re-run previous searches from a search history. Searches are only entered into the search history lost when they differ from the previous search (this avoids duplicating entries if a user resubmits the search form without making any change).
- Search results are presented in 3 layers. After submitting a search the user is always presented with the “collection list” layer, and may navigate through the “brief results” and “full results” to obtain progressively more detailed results. A breadcrumb trail is also provided to allow a user to jump between layers.

- Collections list. This is the first page the end user sees after submitting a search, and presents the results grouped by collection. The aim of this is to make the user aware of the existence of the separate collections, partly to give due credit to content providers, but also because smaller collections that produced fewer (but feasibly the best) hits could be obscured by sheer weight of numbers if the results from all collections were merged.

Information about each collection is displayed on this page, including a summary of terms and conditions. Text from a sample of the search results is analyzed for frequently occurring terms, which are displayed to attempt to convey a quick summary of the subject area covered by the results in each collection.

Each collection featured in the list is linked to a brief display of the records in the result set from that collection.

- Brief display. Nine items are displayed in this layer, with a thumbnail image (when available) and a brief description: title, media type, and a summary of the terms and conditions. All the records in this layer are from a single collection, the name of which is displayed in the page header.

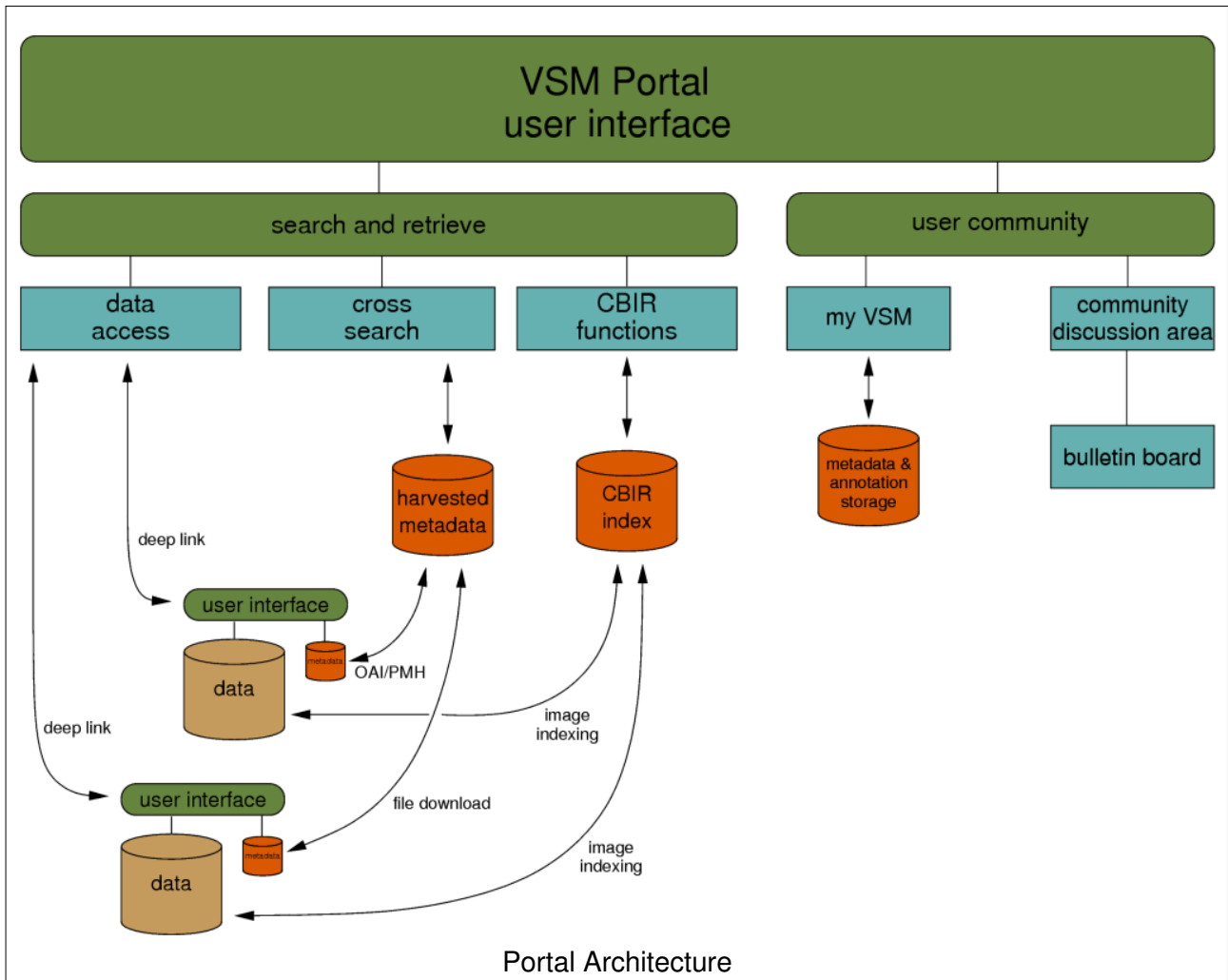
Each record featured is linked to its full display.

- Full display. This displays a single record with a thumbnail image (when available) and all available information: collection name, title and all other descriptive metadata. Two links are provided to the content provider's web site: one giving access to the item (full sized image, video or sound clip) and one to Terms and Conditions.
- Record metadata may be emailed to users. This includes the descriptive metadata plus the two URLs (to access the item, and to access Terms and Conditions) that appear on the full display page. This feature is available from the full display page only at present, though it could very easily be added to the brief display page as well.
- Records may be saved to a user profile, labelled “MyVSM” in the user interface. The user may organize saved record into folders, and add personal (private) annotations. When a record is saved the user is directed to the MyVSM page (see below), but if they opt to return to the search results via the main menu they are directed to the results page they last viewed (*i.e.* they do not need to navigate through the search results again, but may carry on browser from where they left).

- This feature is available from the brief display and full display pages.
- “MyVSM” may be accessed directly from the main menu. This allows users to view records they have previously saved; they may view individual records or an entire folder. Saved records may also be edited, moved between folders or deleted.
- A “find more images like this” function is available for collections which have been indexed by the content based image recognition (CBIR) software. This executes a search for images similar to a selected “seed” image, based upon colour, shape and texture parameters. This feature is available from the full display page only at present, though it could very easily be added to the brief display page as well.
- The bulletin board is integrated with the portal, can provide a number of forums, which can be added to or adapted to meet user needs. A link on the full display page allows users to start a discussion related to a particular item, and when discussion is already underway concerning an item, recent remarks are displayed along with link that allows users to join in with the discussion. Due to the space required for these features, it would not be practical to attempt to add them to the brief display page.
- Authentication is used where it is necessary to identify a user between sessions. This used EDINA mechanisms for Athens authentication, though in principle any system may be used. This has been applied to the MyVSM and Bulletin Board features.
- Authentication has also been added to the email feature; ideally we would not need to recognize users, but regrettably access control is needed to prevent the feature being abused for sending “spam” email.
- Features that require authentication are denoted by a key icon. When a user has logged in, a “Logout” link is added to the main menu so they may terminate their authenticated session; if clicked, an option to log out of the Athens single sign-on is also given.
- Context sensitive help is provide throughout. The content of the help pages is loaded from a separate set of static HTML files on the server, so these may be edited by staff who are not software engineers familiar with Apache::ASP.

Architecture

A technical plan was drafted prior to implementation, as documented in the main body of the Final Report. This section describes the design and software components of the VSM Portal at the end of the project (January 2008).



Software components used

Component	Software choice	Reason for choosing	Outcome
User interface	Apache::ASP http://www.apache-asp.org/	Offers fast and easy development, and is extremely flexible.	Some minor problems with installation of the latest version; all resolved within a week. Subsequently, and as expected, development was fast and flexible. The platform has been reliable.
Database: metadata hosted by the portal	Zebra http://www.indexdata.dk/zebra/	Provides structured text indexing and retrieval. Fast and scales well, provides powerful and flexible text search capabilities.	As expected this database proved very effective and offered very fast searching. Configuration and management options exceeded expectations. Now provides support for SRU/SRW as well as Z39.50.
Database wrapper (search/retrieval API)	Net::Z3950::ZOOM (CPAN) http://search.cpan.org/~mirk/Net-Z3950-ZOOM-1.18/lib/ZOOM.pod	Tested and found to function well. Could also provide the facility for adding a distributed search at a later stage, if required.	Performed as expected without problems.
Database: user data	Text files	Fast and robust; not as powerful as a relational database, but very simple with reduced overheads and dependencies. Indexing is not required.	As expected, all required functionality was available without difficulty, and excellent performance. We used the CPAN package Config::IniFiles to manage file input/output.

<i>Component</i>	<i>Software choice</i>	<i>Reason for choosing</i>	<i>Outcome</i>
Harvester	HTTP::OAI::Harvester (CPAN) http://search.cpan.org/~timbrody/HTTP-OAI-3.1.3/lib/HTTP/OAI/Harvester.pm	Tested and found to function well.	The package performed excellently, but unfortunately the OAI servers gave us severe problems, notably with resilience and performance, incorrect character encodings, and failure to comply with the PMH protocol. This required the coding of an OAI/PMH client with various workarounds that could be deployed for various servers. This used a variety of lower level CPAN packages to handle HTTP requests and responses, character encodings and XML parsing. The development of this client was relatively easy. Significant effort was used investigating the servers, and determining the problems and workarounds required in each case.
Metadata conversion	XML::Twig (CPAN) http://search.cpan.org/~mirod/XML-Twig-3.29/Twig.pm	Tested and found to function well. Particularly efficient for handling very large XML volumes.	Provided all the functionality we required with excellent performance when handling large XML files.
	Perl	Perl is an excellent tool for handling arbitrary text formats.	As expected, a flexible tool with quick development and very fast processing.
	Other	TBA	No other metadata formats were encountered.

<i>Component</i>	<i>Software choice</i>	<i>Reason for choosing</i>	<i>Outcome</i>
Authentication: Shibboleth Athens Locally registered users IP address checking	EDINA already has software for these mechanisms, deployed in Java and Perl based services.	Already available and tested.	At the current time only Athens has been implemented. Shibboleth can be added from existing code. IP address checking is not suitable as authentication is used in the Portal Demonstrator to identify specific individuals, not membership of institutions.
Bulletin board (BB) to provide repository for reviews of content, with capacity for discussion	Simple Machines Forum (SMF) http://simplemachines.org/	BB system with suitable functionality, including an API that will permit integration with the rest of the portal user interface.	Proved suitable. All the necessary functionality is available. In some areas the technical solutions for integrating with the Portal Demonstrator are not elegant, however they are simple enough to avoid serious maintenance problems.
Content based image recognition (CBIR)	LIRE (Lucene Image REtrieval) http://www.semanticmetadata.net/lire/	Documentation and online demonstration suggests it will perform the required function.	Functionality appears to be perfectly adequate, but following indexing of collections it is not clear it will scale well. Further development may address the performance issues, but it is also possible that an alternative package may have to be used.

Platform

This implementation is on Sun Solaris 2.8, though all the software components used are known to work on a wide range of platforms. The host server (SunFire 4800) has eight 900Mhz processors (UltraSparc III) and 16Gb of memory, but this specification is much greater than would be required for the VSM Portal Demonstrator as this server hosts a number of well-used JISC services. This hardware configuration is suited to running numerous software components concurrently, rather than being optimized to perform intensive tasks at high speed, but we have not specified a minimum hardware requirement since this would depend principally on the number of concurrent users and response times that would be required for a service.

A service could run on a dedicated server with much lower specification. The current server is well specified for memory, which is important for transforming and indexing metadata, so the most likely noticeable performance loss if using a lower specification server would be in the database building stage; this would still be entirely functional, but may take significantly longer. However, the current server does not have particularly fast processors, which is the main factor that could impact on the performance of the built database; this shows that excellent database performance is attainable without specifying high processor speeds, and suggests that a lower server specification should not be detrimental to the performance of a service.

Portal GUI infrastructure

Development of the web interface has been achieved through an Apache::ASP framework. Apache::ASP is an extension to the popular and proven Apache web-server and the mod_perl Apache module that allows Perl-based development within Apache. It implements the Active Server Pages API using embedded Perl only. It is a mature and featureful product that is actively developed.

The Apache::ASP framework enables rapid development of a dynamic user interface with in-built full service and session handling. It is capable of using cookies for session identification, or to fallback to an identifying query-string parameter making it suitable for the most stringent web-browser environment. Since it operates upon a mod_perl platform, the embedded code is pre-compiled, making the execution of embedded scripts highly efficient. Its basis in perl means that the majority of the wealth of existing perl modules are available for use within the interface development.

The Apache::ASP code is portable across different implementations of Apache::ASP and is not platform or installation dependent. This means it is a highly portable solution to web interface development, being usable on any platform that can sustain an Apache with mod_perl web server.

The VSM implementation is using Apache version 2.2.4, mod_perl version 2.0.3 and Apache::ASP version 2.59.

Database and API

Database

The database being used is Indexdata's Zebra software, version 2.0.14. This is a mature open source product that has a proven track record of being robust, efficient, flexible with a powerful set of features.

The native interface to the Zebra database is through a Z39.50 or SRU/W connection. This makes it standards compliant by default, and provides a machine-to-machine protocol for third parties to query the VSM metadata should the need arise. The VSM implementation of Zebra uses an SRU/W connection which provides a Web Services based protocol for querying the database and retrieving metadata records.

Zebra allows multiple named databases to reside in the same instance of the database server, and also allows live updating of individual databases using shadow indexes which can be rolled back should an error occur. This makes overall maintenance of the large number of collections used in VSM (each implemented as a separate database in Zebra) much simpler than otherwise might be expected.

The main downside of using Zebra is that it requires a non-standard and sometimes complicated configuration to specify how the metadata are indexed and also how a Z39.50 Bib-1 or CQL query maps to the configured indexes. However, since VSM is harvesting and re-formatting the metadata, they are made to be uniform across all collections meaning the configuration only has to be done once.

Some of the features that are being exploited by the VSM interface are:

Portability

Zebra can be compiled and used on a wide range of platforms, from most unix variants (including Linux and Mac OS X) to Microsoft Windows.

Shadow indexing for live updates

This makes maintenance of the various collections of metadata much easier to deal with, making a large proportion of home-made maintenance scripts unnecessary.

Standards based network search protocol

Zebra uses Z39.50 as its native search interface. This automatically provides standards-compliant machine-to-machine capability. SRU/W is easily configured on top of the initial Z39.50 set up, which provides a Web Service interface, more accessible to modern demands. It also allows use of the CQL query language, developed as part of SRU/W, which is a highly adaptable query language that can express from intuitive simple queries to complex machine-constructed queries.

Word proximity indexing to allow phrase searching

Proximity searching in most Z39.50 based database software is sketchy at best. Zebra's proximity searching allows intuitive sub-phrases of multiple words easily searched for by quoting the text.

Dynamic relevancy ranking with adjustable individual index weighting

Relevancy ranking is one of the most valuable features to users that a database server can provide. In Zebra, separate indexes can be weighted differently, and these weightings can be changed on a per-search basis, rather than at database build time.

Regular expression searching and fuzzy matching

Zebra allows complex regular expression searching of indexes, an unusually powerful feature. In practice this potential will only be touched on by allowing users to add wildcard characters to their search terms. Fuzzy matching is also useful when spelling is an issue.

XML and tag-formatted output

Zebra accepts XML as an input syntax, and will similarly output metadata in this format. It will also automatically create sensible "tag" formatted ASCII records that can be viewed through the Z39.50 interface (using a SUTRS formatted present request), though not through the SRW/U interface which requires XML.

Efficiency

Zebra has an extremely capable and efficient database search engine. Whilst it was originally thought that some advanced features might need to be off by default, and only used when the user specifically requests it (such as relevancy ranking or phrase searching), it transpires that these features can be on by default without any noticeable degradation in performance.

API for the user interface

Whilst the Zebra database server uses standards-based search and retrieve protocols for access, it was decided that we would abstract access to the underlying database behind a more general Perl API so that any desired change in the backend database could be accommodated without the need for any change in the user interface.

This API is designed so that the developer of the interface has easy and immediate access to functionality that would be expected by the user. Searches can be performed by supplying the search query of the user along with the list of collection names required to be searched, and flags for various

search switches. Alternatively, a list of metadata record identifiers can be supplied to the API. Both methods return a result set that can be dealt with in exactly the same way, although what actually happens behind the scenes is quite different. In the former case, a search is instantiated immediately and a genuine result set is presented back to the caller, whereas in the latter case (developed specifically to tie in with the CBIR functionality), a partial search is only performed as individual records are requested. This improves perceived efficiency. The API uses the CPAN³/Indexdata ZOOM module to contact the underlying Zebra database using SRW.

Whilst the API is for use by a Perl developed user interface, it would be simple to wrap the API with a SOAP interface using SOAP::Lite (a proven Perl package available via CPAN). This would allow a service oriented architecture to be used, and for a user interface to be implemented using any language capable of utilizing such an architecture.

Community Discussion Area (Forum)

The Community Discussion Area was implemented as a forum. This was done partly because of the wide availability of Open Source forum software. More importantly, many end users are familiar with forum software and the user interfaces, functionality and conventions they use.

Forum software

We selected the Simple Machines Forum⁴ (SMF) as a suitable product. This software is based on well known and reliable products: Apache⁵ web server, PHP⁶ embedded scripting and the MySQL⁷ database. SMF satisfies important requirements for integrating the forum with the portal:

- Template based customization, enabling the “look and feel” of the portal to be emulated by the forum.
- Generation of summaries of discussion threads as HTML fragments, which may be embedded into web pages generated by the portal.
- Generation of links which may be embedded into web pages generated by the portal, enabling end users to initiate or contribute to discussion threads from within the portal
- The ability to use external authentication (**Athens**, UK Federation, and others) to authenticate forum users.

The decision was made by checking documentation for the required features, making judgments of the reliability of the underlying components based on practical experience and a preference for Open Source over proprietary software. SMF appeared to be a suitable product, and this was confirmed with a test installation.

It is possible that other forum software may be suitable or superior, but we did not attempt an objective comparison between products. SMF was adequate for the needs of demonstrator.

Functionality

SMF provides a wide range of conventional forum features: the ability to create new discussion threads, to contribute to existing discussions, to set email alerts that notify end users of new posts to threads they are interested in, and so on. These features are documented at <http://www.simplemachines.org/about/features.php> Here we describe features and functions specific to SMF as we have integrated it with the portal.

³ <http://www.cpan.org/>

⁴ <http://www.simplemachines.org/>

⁵ <http://httpd.apache.org/>

⁶ <http://www.php.net/>

⁷ <http://www.mysql.com/>

Accessing the forum

The forum may be accessed via the *Forum* option on the portal main menu, which simply links the user to the forum user interface. In addition, certain discussion threads are associated with VSM database records (see below), so forum content may be read and end users may contribute from web pages generated by the portal.

Though the portal and the forum have separate user interfaces, hosted on separate web servers, we have modified the “look and feel” of the forum to emulate the portal as closely as possible. Ideally users should not feel as if the two components are separate services, but all a part of the same portal. It was not possible to produce an identical design, though the basic page template, logos, colours and fonts of the portal were successfully reproduced in the forum. SMF may be customized to a greater extent than this, and more development could be done in this area.

Authentication and registration

End users are required to authenticate before accessing the forum. At the current time Athens is supported, but this could be extended to the UK Access Management Federation or any other scheme. This authentication step provides a unique user identifier which may be used by the forum.

On the first occasion that a user visits the forum, they will be asked to register; this step collects an email address (required for email alerts provided by the forum) and allows the user to provide a name by which they are known in the forum (their user identifier remains private). A user need register only once; on each subsequent visit to the forum, they will be recognized by the user identifier provided by the Athens (or other) scheme.

Associating discussion threads with VSM database records

The forum permits threads to be started on any topic, so the user community may discuss any issues they wish. To integrate the forum with the portal, however, certain threads are directly associated with records in the VSM database (*i.e.* pictures, video or sound harvested from contributing content providers). These threads may be read within the forum, like any other threads, but they are also integrated directly with the portal. This is intended to widen participation, by bringing discussions to the attention of users who may not have been inclined to access the forum via the portal main menu.

Threads linked to a record are displayed on web pages generated by the portal; in the current design this is done on the “full record” page (*i.e.* when a user has run a search and navigated through results to display a single record with all available metadata). If there are a large number of contributions to the thread, only the most recent posts are displayed. Links are provided which allow the end user to view the thread within the forum user interface (this will display the thread in full, with functions such as email alerts when other users add new posts) and to post a new contribution to the thread.

Where there is no thread associated with a record (which of course will be the initial position for all records), the “full record” carries a link which invites end users to start a discussion.

Forum API

As indicated in the technical plan, an abstract API was implemented. This means the interface used by the portal contains nothing specific to the current implementation in SMF. It would be possible to implement the API as a SOAP interface if a formal Service Oriented Architecture (SOA) is required in future, but at the current time it is implemented as Perl package. This is a simpler and more robust architecture, with better performance, which may readily be converted to SOAP by using SOAP::Lite (a proven Perl package available via CPAN).

Content Based Image Retrieval

Content Based Image Retrieval (CBIR) is a feature that indexes the actual images associated in a way that allows it to match a given image to similar images found within the set of indexed images, given certain parameters. These parameters bound the likeness of an image based on its colour, texture and shape. CBIR is not very new, but it is also not widespread, and so there are no set user expectations

as to how to use this functionality. It is a feature that may turn out to be very useful to users, depending on the accuracy of the results. However, it was unproven that available CBIR search engines could deliver usefully accurate results, and so this feature within the interface was investigative of the underlying technology as well as how users might use such a facility.

The CBIR engine is a completely separate component from the VSM database (implemented with Zebra), using the actual data of the image objects (where available) from relevant collections. Whilst a service oriented architecture was decided upon at the initial stages, the exact software performing the indexing and searching was still to be decided, and so two CBIR engines were explored. The decision to use SOAP as the API allows us to switch the backend engine without needing to alter the client software (residing within the portal web interface and portlet) in any way.

The two options that were explored, GIFT⁸ and LIRE⁹, are both mature, open-source software with an active support community. GIFT is a C-based suite which uses its own XML markup language, MRML, for queries. LIRE is an extension to the popular Lucene¹⁰ engine, which is Java based and supported by the Apache Software Foundation.

The SOAP API was most easily implemented using an AXIS2¹¹ tomcat server attached to the LIRE API. There are only a few SOAP calls defined, providing all that is needed for sensible CBIR functionality by an end-user.

The LIRE engine indexes all images available to the VSM at the time of harvesting each collection, if a particular image is new or changed. These metadata are indexed against their VSM record identifier as used in the XML metadata indexed by the VSM database.

In fact the indexing of the images is done using the same collection level XML record produced by the harvester that is used by the VSM database. Most of the metadata elements are redundant for the CBIR engine, with only the record identifier and the URL to the thumbnail being used: the former as the obvious identifier string, and the latter to retrieve the actual image (or thumbnail) for use in indexing.

When a particular image is located by a user within the VSM web interface, the “More images like this” function passes the VSM record identifier to the CBIR engine via the SOAP API. The referenced image is used as a “seed”, and provides parameters that are used to initiate a search for similar images. This identifier is used by LIRE to look up the existing indexed CBIR metadata for the corresponding image, and then the similar set is retrieved, referenced again by the VSM record identifier. This is particularly efficient since all the images are pre-indexed.

The returned list of VSM record identifiers for similar images is then passed back to the VSM portal interface which can then look up the full metadata for each item using the API to Zebra. This set is presented to the user in exactly the same way as the results from a normal text-based search, and so navigation of the set should be intuitive.

The LIRE software appears to be successful in retrieving images with similar visual properties to the seed image. Scaling is more of an immediate issue, since searches become slow when large numbers of images have been indexed; at the current time, images from Scran¹² (a large collection) have not been indexed because the response time becomes unacceptably slow. It is possible that “tuning” certain parameters of the LIRE software may be sufficient to rectify this, or it may be that significant development may be needed.

MyVSM

This component of the portal permits users to save records between sessions, along with their personal annotations. We decided to implement this using text files as this is simple and robust, offers good

⁸ <http://www.gnu.org/software/gift/>

⁹ <http://www.semanticmetadata.net/lire/>

¹⁰ <http://lucene.apache.org/java/docs/>

¹¹ <http://ws.apache.org/axis2/>

¹² <http://www.scran.ac.uk/>

performance and is sufficiently flexible for our needs.

We used the Perl package Config::IniFiles (available from CPAN, <http://www.cpan.org/>) as an interface. Though we intended to create an abstract API, with a view to flexibility and the ability to add a SOAP interface later, this was not done due to time pressures. This is mitigated by two facts: firstly, Config::IniFiles provides an API with sufficient abstraction for our immediate needs, and secondly the functionality provided at this stage is quite simple. Nevertheless, if this area of the portal is developed further an abstract API will be a priority.

Functionality

Saving records

A user may save a record from the “brief record” or “full record” page (*i.e.* when a user has run a search and navigated through results to display one or more records). They may choose a title for the record (which defaults to a string created from the name of the collection in which the record was found, and the title of the record) and add personal annotations. Records will be saved in a folder named Default, unless the user selects another folder or creates a new folder.

Saving a record saves a *reference* to the record only. Only the record identifier, note of the media type (still image, moving image or sound), user derived data (title, folder and annotations) and a timestamp for the last modification are saved.

Viewing records

Users may view the records they have saved, arranged by folder. Since records are saved by reference, if the user views a saved record then the metadata are loaded from the VSM database. This ensures that if a record is modified, the user will always be presented with the most recent metadata; this could be particularly important if terms and conditions or URLs for accessing the item are changed. It also ensures that if a record is deleted by a content provider, possibly for legal reasons, no copy of the metadata persists within the portal and continues to be offered to the user.

In the event that a saved record has been deleted, it is flagged as unavailable but it is not deleted from the user’s saved records. Thus a record that is temporarily withdrawn, and then restored to the database, will become available again for users who had saved it.

Moving, editing and deleting records

Users may move saved records between folders, edit details (title and annotations) and delete unwanted records. For simplicity this functionality is all provided by a single web form (which is the form originally used to save the record).

Metadata Harvesting

The VSM database contains metadata provided by content providers. We use the term “harvesting” here to describe the process of obtaining these metadata, parsing the files and producing a common format for use in VSM.

Harvesting is a process that is carried out prior to indexing the metadata for the VSM database, so (unlike the other components described above) the harvesting software is not used directed by the portal user interface. If developed into a full service, harvesting would be conducted on a regular schedule to keep the VSM database up to date with the content offered by each provider. Depending on the frequency with which content providers updated their own databases, it is likely that harvesting would be carried out approximately monthly. This process would be automated; on the current Solaris platform, we would use the standard Unix *cron* daemon for scheduling.

Harvesting requires the following steps:

- Transfer of metadata file(s) from the content provider to the VSM host server
- Transformation of metadata file(s) into the format used in VSM

- Indexing VSM-format metadata in the database

Transfer of Metadata

The requirement for the successful completion of this step is straightforward: a complete and accurate copy of a content provider's metadata should be created in one or more files on the VSM host server. The process may be more complex.

The simplest mechanism available is to copy a file by making an HTTP request, or copying using a file transfer method such as scp (secure copy using the Secure Shell protocol). Provided the metadata files are neither very large nor numerous, a straightforward copy is certainly the easiest method to implement; it does require a little knowledge of the content provider's web server or other file transfer mechanism, but in practice obtaining these details is a trivial amount of work compared to the effort involved in transforming and indexing their metadata.

Other mechanisms exist for transferring metadata. In this project we encountered two mechanisms:

RSS

RSS is a mechanism for describing the content available on a web site, using a simple XML format. There are differing formats; for a summary see [http://en.wikipedia.org/wiki/RSS_\(file_format\)](http://en.wikipedia.org/wiki/RSS_(file_format))

RSS provides a description of an item and a URL for linking to a page providing further information. It is most commonly used for news items where updates are frequent, but it is perfectly suitable for linking to web sites which are updated less frequently or ones which are completely static.

Though the essential functionality (describing an item and providing a link for access) is a good match for the VSM portal requirements, RSS is too simple to provide rich metadata. In particular as it is designed for linking to web pages, data concerning media type and file formats are lacking.

OAI/PMH

The Protocol for Metadata Harvesting (PMH) developed by the Open Archives Initiative (OAI) is ideal for our harvesting requirements, and it has been implemented by the majority of content providers. Regrettably the quality of implementation is poor in many cases; servers may not comply with the standard, and may return XML with invalid encodings. It has proved necessary to develop an OAI harvesting mechanism that does not assume compliance with the standard, and which implements a number of "on the fly" fixes and workarounds.

This technical development is well within our capability, however it requires considerable investigation and writing of software specific to each target. It effectively negates the benefits of employing a standard, when each mechanism has to be investigated in detail to determine its behaviour, without documentation, and code specific to each one has to be written. In fact, content providers who use simple mechanisms based around file transfer or RSS have proved easier to work with than those who have sought to implement OAI/PMH.

Transformation of Metadata

This step involves parsing metadata files transferred from content providers, and producing metadata files in the standard VSM format.

An alternative method would be to configure the database to index files of each different format, and eliminate the stage of creating VSM format files. We rejected this for several reasons. Firstly, by maintaining VSM format files we would be able to quickly rebuild the database at any stage from a stable file format; this should be redundant, but redundancy is a desirable attribute if a resilient system is required. Secondly, we (correctly) anticipated that the transformation may be complex in places, and that it would be better achieved via a programming language than database configuration. Thirdly and lastly, the existence of files as an intermediate step makes testing and debugging simpler.

As anticipated in the technical plan, an object oriented approach was used. This minimized the code

that had to be written for each new collection. A new standard-compliant collection can theoretically be added without unnecessary duplication of code. Other collections may require code to work around standards-compliance issues, and some collections will inevitably require unique solutions. These eventualities can all be efficiently resolved with an object oriented structure, so the code base should prove to be extensible.

In the short to medium term, it would be productive to identify features of collections that are common and may be usefully dealt with by configuration rather than code. For example there is duplication of Perl code in some cases where similar XML formats occur in different collections (which was not apparent at the outset), in particular XPATH declarations for elements that are directly incorporated into VSM metadata elements; this would more elegantly be implemented with shared code and configuration files.

Indexing of Metadata

Metadata files in VSM format may be indexed by Zebra. Due to a problem installing XML DOM support for Zebra, it was necessary to remove the container element (vsm:collection) for indexing. This solution is inelegant, but in practice removing the top level XML element from a file is such a simple operation that resolving this problem is a low priority at the moment.

Harvesting versus Distributed searching

There are two main ways in which portals may enable searching of metadata. Firstly, metadata may be searched on the content providers' servers, using a protocol such as Z39.50 or SRW. Secondly, metadata may be harvested using a protocol such as OAI/PMH, and then searched on the portal server.

The first method, often called a “distributed search”, has certain advantages.

- It is not necessary to maintain a copy of all metadata on the portal server. If the portal searches large collections, the need for large volume of disk storage is avoided.
- It ensures that the search is always conducted on the most up to date metadata that are available.
- In cases where the metadata are the content providers sole source of revenue (such as Abstracting and Indexing databases), they may understandably be reluctant to implement a protocol that enables their entire collection to be downloaded.
- Access control can be applied by the content provider, ensuring that every search and retrieval of metadata is properly authorized.

A distributed search also has disadvantages:

- Because it involves connections to large numbers of remote servers, it may be slow and at time unreliable. To provide a reasonable service to the end user, the portal will have to implement an asynchronous search, and ideally allow the end user to terminate connections to unresponsive servers. This introduces a lot of complexity in the database client and user interface implementations.
- Database functions such as sorting and relevance ranking must be provided by the remote servers, and these functions may be unavailable or inconsistent. The nature of protocols used for distributed searching, which only enable the retrieval of relatively small numbers of records at a time, also prevents the portal from being able to sort or rank the result set within a reasonable time period.

Harvesting does not suffer the same disadvantages as distributed searching, but it introduces other requirements:

- Metadata must be retrieved and indexed into a database on the portal server, which requires that the metadata are made available in their entirety and the portal is prepared to set aside sufficient disk storage.
- If end users require authorization to access the metadata, this must be applied by the portal and so a high level of trust is required between the portal and content provider.

- Harvesting may be repeated sufficiently frequently that the metadata do not become seriously out of date.

The relative weight of these advantages and disadvantages depends entirely on the context. A distributed search is best suited to large collections, with frequent updates, and in which the metadata are of commercial value; these characteristics are typical of bibliographic collections. For smaller collections, with less frequent updates and freely available metadata, harvesting is likely to be the better option; these are characteristics typical of multimedia repositories.

No content provider working with the VSM Demonstrator project expressed an interest in distributed searching; many ruled it out completely. This allowed us to make a working assumption that we would hold all metadata in a local database; this made it possible to exploit the full functionality of the database system (Zebra) and provide features such as fuzzy searching, sorting and relevance ranking.

It is possible that in future a distributed search will have to be supported. The ZOOM Perl package offers support for searching asynchronously with Z39.50 or SRW, so searching remote databases as well as the local Zebra database should be a relatively straightforward development. More work will be required to resolve issues such as relevance ranking and sorted of results, which is currently done well by Zebra, but is unlikely to be performed in a consistent way (or at all) by remote databases. This will affect the way in which results are presented within the user interface. A further issue that will probably have to be addressed is slow responses or unavailability of remote databases, and how this may be handled without causing unacceptable delays or confusion for the end user.

Appendix 4: VSM Metadata Format

The VSM database stores metadata in a scheme based on Dublin Core (DC), with the necessary addition of some elements specific to VSM. The metadata are rendered as XML for communication between software components.

The root element is `vsm:collection`, which may contain the following elements:

`vsm:collectionID`

Mandatory element, with unique identifier for the collection, assigned by the VSM project.

`vsm:collectionName`

The name of the collection

`vsm:collectionLongName`

Long form of the collection name (often not needed)

`vsm:collectionDescription`

A description of the collection

`vsm:collectionLongDescription`

Longer description of the collection (often not needed)

`vsm:collectionMediaType`

The media type of the items in the collection, as DCMI type: *StillImage*, *MovingImage* or *Sound*. May be repeated for collection with more than one media type.

`vsm:record`

A record within the collection. Repeated for each record in the collection.

The list of `vsm:record` elements is the main body of each metadata file, and are mostly DC. A `vsm:record` may contain the following elements:

`vsm:id`

Mandatory element, with a unique identifier for the record, usually (but not defined to be) derived from a record identifier supplied by the content provider, prefixed with the `vsm:collectionID` value as a namespace. Though DC has an identifier element, this can be used for *any* form of identifier applicable to an item (*e.g.* a URI), which may not be unique; we preferred to have an unambiguous `vsm` element for this purpose.

`vsm:thumbnail`

The URL of a thumbnail for the record; DC provides no very clear way of representing this.

`dc:contributor`

This is used to store all references to people or organizations responsible for creating or providing the item; repeated for multiple contributors. This is rather vague, but source metadata are not sufficiently consistent for providing more details regarding roles. We regard it as unacceptable to attribute a false role to a person or organization as a result of misinterpreting metadata, and sought to avoid this (even if it results in some ambiguity).

`dc:description`

Suitable descriptive text is always available in the source metadata.

`dc:format`

MIME type(s) of the item(s) available; repeated where multiple formats are available.

Regrettably the source metadata are rarely detailed enough regarding formats for us to be able to include this element.

`dc:rights`

Used to reproduce any rights statements provide in the source metadata.

dc:source

The URL that may be used to access the item. Preferably this should be a “deep link” to a web page on the content provider’s web site which displays the item. If that is not possible, it may be necessary to link to the content providers’ home page or search form.

dc:subject

Used for anything that resembles a subject term, even if it was in another element in the original metadata. Generally repeated to include many subject terms. *e.g.* a dc:coverage may be provided in the source metadata, but this cannot be stored as coverage data, since collections are not consistent in their use of dc:coverage; but it can be added as a dc:subject. This ensures it is indexed in the VSM database, and that the record will be found if a user searches for the term, even though it is not possible to include a specific coverage search.

dc:title

A title of some form is always available in the source metadata.

dc:type

The DCMI type for the item: *StillImage*, *MovingImage* or *Sound*

The following page shows an example VSM metadata record rendered as XML.

```

<vsm:collection
  xmlns:vsm="http://edina.ac.uk/vsm/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/">
  <vsm:collectionDescription>
    ARKive is the Noah's Ark for the Internet era – a unique global initiative,
    gathering together into one centralised digital library, films, photographs
    and audio recordings of the world's species.
  </vsm:collectionDescription>
  <vsm:collectionID>
    arkive
  </vsm:collectionID>
  <vsm:collectionLongName>
    ARKive - Images of life onEarth
  </vsm:collectionLongName>
  <vsm:collectionName>
    ARKive
  </vsm:collectionName>
  <vsm:record>
    <vsm:id>
      arkive:A21C26C4-1464-4DF9-BE7F-E614ECF3B6EC:1
    </vsm:id>
    <vsm:thumbnail>
      http://www.arkive.org/media/A21C26C4-1464-4DF9-
      BE7FE614ECF3B6EC/Presentation.Streams/picture.jpg
    </vsm:thumbnail>
    <dc:contributor>birdphotography.com</dc:contributor>
    <dc:description>Akiapola'au - overview</dc:description>
    <dc:format>video/quicktime</dc:format>
    <dc:format>audio/x-pn-realaudio</dc:format>
    <dc:format>video/x-ms-wmv</dc:format>
    <dc:source xsi:type="dcterms:URI">
      http://www.arkive.org/species/GES/birds/Hemignathus\_munroi/Hemignathus\_munroi\_00.html
    </dc:source>
    <dc:subject>
      animal pictures
    </dc:subject>
    <!-- further dc:subject elements -->
    <dc:title>
      Akiapola'au - overview
    </dc:title>
    <dc:type>
      MovingImage
    </dc:type>
  </vsm:record>
  <!-- further vsm:record elements -->
</vsm::collection>

```

Example VSM metadata record rendered as XML