



# Automating Production of Cross Media Content for Multi-channel Distribution

**www.AXMEDIS.org**

# DE4.1.1
# Content Modeling and Managing

**Version:** 1.4
**Date:** 05/10/2005
**Responsible:** EPFL (revised and approved by DSI)

| |
|---|
| Project Number: IST-2-511299 |
| Project Title: AXMEDIS |
| Deliverable Type: Report and Prototype |
| Visible to User Groups: Yes |
| Visible to Affiliated: Yes |
| Visible to Public: Yes |

| |
|---|
| Deliverable Number: DE4.1.1 |
| Contractual Date of Delivery: M13 |
| Actual Date of Delivery: 10/11/2005 |
| Work-Package contributing to the Deliverable: WP4 |
| Task contributing to the Deliverable: T4.1.1, T4.1.2, T4.1.3, T4.1.4 |
| Nature of the Deliverable: Report and Prototype |
| Author(s): EPFL, DSI, UNIVLEEDS, EXITECH, SEJER |

**Abstract**
This Deliverable reports the AXMEDIS approach to content management in terms of technology to be adopted for the modeling, editing, viewing and authoring of objects content. The state of the art of existent technology is reported and the existent problems are highlighted. The work performed to extend the existent toolsets to meet the AXMEDIS requirements is described and a plan of foreseen enhancements is provided.

**Keyword List:**
Data Model, Editors, Authoring Tools, Viewers, MPEG-21 Terminal.

# Table of Contents

**AXMEDIS Copyright Notice**

The following terms (including future possible amendments) set out the rights and obligations licensee will be requested to accept on entering into possession of any official AXMEDIS document either by downloading it from the web site or by any other means.

Any relevant AXMEDIS document includes this license. PLEASE READ THE FOLLOWING TERMS CAREFULLY AS THEY HAVE TO BE ACCEPTED PRIOR TO READING/USE OF THE DOCUMENT.

1. **DEFINITIONS**
   - i. "**Acceptance Date**" is the date on which these terms and conditions for entering into possession of the document have been accepted.
   - ii. "**Copyright**" stands for any content, document or portion of it that is covered by the copyright disclaimer in a Document.
   - iii. "**Licensor**" is AXMEDIS Consortium as a de-facto consortium of the EC project and any of its derivations in terms of companies and/or associations, see www.AXMEDIS.org
   - iv. "**Document**" means the information contained in any electronic file, which has been published by the Licensor's as AXMEDIS official document and listed in the web site mentioned above or available by any other means.
   - v. "**Works**" means any works created by the licensee, which reproduce a Document or any of its part.
2. **LICENCE**
   1. The Licensor grants a non-exclusive royalty free licence to reproduce and use the Documents subject to present terms and conditions (the **Licence**) for the parts that are own and proprietary property the of AXMEDIS consortium or its members.
   2. In consideration of the Licensor granting the Licence, licensee agrees to adhere to the following terms and conditions.
3. **TERM AND TERMINATION**
   1. Granted Licence shall commence on Acceptance Date.
   2. Granted Licence will terminate automatically if licensee fails to comply with any of the terms and conditions of this Licence.
   3. Termination of this Licence does not affect either party's accrued rights and obligations as at the date of termination.
   4. Upon termination of this Licence for whatever reason, licensee shall cease to make any use of the accessed Copyright.
   5. All provisions of this Licence, which are necessary for the interpretation or enforcement of a party's rights or obligations, shall survive termination of this Licence and shall continue in full force and effect.
   6. Notwithstanding License termination, confidentiality clauses related to any content, document or part of it as stated in the document itself will remain in force for a period of 5 years after license issue date or the period stated in the document whichever is the longer.
4. **USE**
   1. Licensee shall not breach or denigrate the integrity of the Copyright Notice and in particular shall not:
      - i. remove this Copyright Notice on a Document or any of its reproduction in any form in which those may be achieved;
      - ii. change or remove the title of a Document;
      - iii. use all or any part of a Document as part of a specification or standard not emanating from the Licensor without the prior written consent of the Licensor; or
      - iv. do or permit others to do any act or omission in relation to a Document which is contrary to the rights and obligations as stated in the present license and agreed with the Licensor
5. **COPYRIGHT NOTICES**
   1. All Works shall bear a clear notice asserting the Licensor's Copyright. The notice shall use the wording employed by the Licensor in its own copyright notice unless the Licensor otherwise instructs licensees.
6. **WARRANTY**
   1. The Licensor warrants the licensee that the present licence is issued on the basis of full Copyright ownership or re-licensing agreements granting the Licensor full licensing and enforcement power.

2. For the avoidance of doubt the licensee should be aware that although the Copyright in the documents is given under warranty this warranty does not extend to the content of any document which may contain references or specifications or technologies that are covered by patents (also of third parties) or that refer to other standards. AXMEDIS is not responsible and does not guarantee that the information contained in the document is fully proprietary of AXMEDIS consortium and/or partners.
3. Licensee hereby undertakes to the Licensor that he will, without prejudice to any other right of action which the Licensor may have, at all times keep the Licensor fully and effectively indemnified against all and any liability (which liability shall include, without limitation, all losses, costs, claims, expenses, demands, actions, damages, legal and other professional fees and expenses on a full indemnity basis) which the Licensor may suffer or incur as a result of, or by reason of, any breach or non-fulfilment of any of his obligations in respect of this Licence.

7. **INFRINGEMENT**

1. Licensee undertakes to notify promptly the Licensor of any threatened or actual infringement of the Copyright which comes to licensee notice and shall, at the Licensor's request and expense, do all such things as are reasonably necessary to defend and enforce the Licensor's rights in the Copyright.

8. **GOVERNING LAW AND JURISDICTION**

1. This Licence shall be subject to, and construed and interpreted in accordance with Italian law.
2. The parties irrevocably submit to the exclusive jurisdiction of the Italian Courts.

**Please note that:**

- You can become affiliated with AXMEDIS. This will give you the access to a huge amount of knowledge, information and source code related to the AXMEDIS Framework. If you are interested please contact P. Nesi at nesi@dsi.unifi.it. Once affiliated with AXMEDIS you will have the possibility of using the AXMEDIS specification and technology for your business.
- You can contribute to the improvement of AXMEDIS documents and specification by sending the contribution to P. Nesi at nesi@dsi.unifi.it
- You can attend AXMEDIS meetings that are open to public, for additional information see WWW.AXMEDIS.org or contact P. Nesi at nesi@dsi.unifi.it

# 1 Executive Summary and Report Scope

The AXMEDIS consortium (producers, aggregators, distributors and researchers) aims at creating an efficient and complete framework to speed up and optimize content production and distribution with innovative methods and tools compliant with international standards - such as MPEG-21 - and de-facto standards. In the course of realization of such a platform for *production-on-demand,* content modeling and managing play a key-role since the heterogeneity of content formats and representations may create serious interoperability issues if not treated appropriately. In this sense, the AXMEDIS approach is based on the MPEG-21 concept of Digital Item which is the fundamental unit of distribution and transaction manipulated by the users of the framework [1].
Several are the technologies and the approaches currently adopted by the AXMEDIS partners according to their specific business sectors. This document provides then the state of the art of Content Managements Systems (CMS) which can range from simple software solutions to very complex architecture comprising multiple applications.
Three are the main elements of a CMS:

- Data model.

  In order to implement interoperable content manipulation, standard interfaces for content descriptions must be specified. In AXMEDIS such interfaces will be implemented using the XML language in accordance with the major trend of multimedia standard bodies, open initiatives and the industry. XML will be adopted as AXMEDIS file format and tools able to open, manipulate, save and possibly adapt such format will be implemented.
- Authoring toolset.

  Several are the tools necessary to enable content manipulation from its creation to the final distribution passing through several steps of editing/modification. These tools include:
  - *Viewers* able to open content objects and display the carried resources is terms of hierarchy of elements and sub-elements, expected behavior in time once decoded and visual organization in virtual bi and tri-dimensional spaces.
  - *Editors* enabling content creation and modification in user-friendly graphical environment. These tools would provide a powerful environment to potential content creators without the need to directly edit XML files. Moreover, several multimedia objects may need to be organized in complex "scenes" where the relationships among them are coded in a standard format. Being SMIL (Synchronized Multimedia Integration Language) a very popular and widely adopted format, appropriate SMIL editors and decoders will be implemented.
  - *Content (pre/post)processors*. An optimal content distribution may require the binarization of multimedia content in order to fully exploit the available resources in terms of bandwidth and storage space. This implies the need of pre-processors able to convert binary XML to textual format (and possibly resolve external references) and post-processors producing binary content ready to be streamed or downloaded.
- Decoding device (the player).

  In the MPEG-21 vision the end user may become content producer and distributor himself. The specification of the client terminal architecture and functionality is then part of the CMS specification. Differently from MPEG-2 and MPEG-4, in MPEG-21 a terminal architecture cannot be defined as an aggregation of functional blocks (decoders, buffers, demultiplexers, etc.) whose behaviors are driven by specified rules coded as standard descriptors. An MPEG-21 compliant terminal is rather a device able to correctly manipulate Digital Items by invoking a series of tools devoted to specific actions to be performed. These may include: content governance (DRM), adaptation and processing. The specification and implementation of an MPEG-21 compliant device requires then the specification and implementation of: (i) interfaces among the involved MPEG-21 components, (ii) mechanisms enabling the invocation of the required standard or proprietary decoders once the carried resources need to be accessed by the end user.

# 2 Introduction

WP 4.1 addresses the problem of content modeling using an MPEG-21 approach, providing the suitable research, development and then overall enabling technology to demonstrate this approach. Compatibility with existing CMS tools of AXMEDIS partners will be considered. This introductory section recollects the specification of research for WP4.1.

## 2.1 Specification of CMS Analysis for AXMEDIS Model definition and Content structure Mapping (DSI, UNIVLEEDS, EXITECH)

**Major partners involved**

DSI, EXITECH, UNIVLEEDS and all the content distributors in the Project

**State of the art**

Most content providers and distributors have their specific Content Management System (CMS), and larger companies even have multiple systems in place. Implementations are usually focused on Web content, documents, digital assets, or XML data. The state of the art of these large software systems is quite complex: it is rather difficult to compare them in terms of performance and data models since they are tuned for the specific needs of the company (publishers, distributors, aggregators, etc., have different needs in terms of metadata and files manipulated).

**Research and development plan**

In this task, the CMSs which are used by the project partners (content providers, aggregators and distributors) will be analyzed: XAURA of TISCALI, HP, WAN of OD2, XIM, etc. considering the several factors and aspects that have to be mapped to AXMEDIS objects and components. This activity will allow confirming and updating the structure of MPEG-21 digital item and DRM. The implementation of the Collector/Transcoder that will allow porting the content collected (crawled) from the CMSs into the AXMEDIS database will be addressed and realized into WP4.2. The aspects that will be taken into account include (but not limited to) the following issues:

- Content Formats
- Content production Workflow
- DRM models and interaction
- Database structure and interfacing tools
- Ingestion mechanisms
- Compositional tools for combining content components
- Communication capabilities
- Administrative information availability such as licensing, etc.
- Monitoring and versioning capabilities
- Automatic production capabilities, etc.
- Metadata added to content
- Database model

**Planned schedule**

- M7: analysis of considered CMSs to identify
    - their integration capabilities and
    - which information and metadata,
    - content type, and
    - DRM info they collect;

- M14: refined analysis of considered CMSs to identify
    - their integration capabilities and
    - which information and metadata,
    - content type, and
    - DRM info they collect

## 2.2 Specification of AXMEDIS Data Model (DSI, UNIVLEEDS, EXITECH and EPFL)

**Major partners involved**

DSI, EPFL, EXITECH, UNIVLEEDS and all the content distributors in the project

**State of the art**

The work of standards bodies and especially MPEG is currently in progress, in particular on MPEG-21. If this new standard will constitute a complete set of specifications for multimedia frameworks, what standard bodies are not going to provide is an implementation of the recommended tools able to demonstrate their effectiveness and their capability to fit into an overall system that meets real business models and needs. This means that in terms of state of the art, being the MPEG-21 specification still in progress, only non efficient pieces of software are available, coming from different sources and only intended to demonstrate functionality based on the development of the different specifications.

In this subWP, comparison with the state of the art will come during the research process through a continuous monitoring of the parallel activity of other research centers and consortiums working on the development of MPEG-21 and other equivalent systems.
At the moment of writing this deliverable, it is known that another IP Project consortium active in the 6$^{th}$ European Framework, ENTHRONE [2], is specifying the architecture of an MPEG-21 system including the implementation of a compatible MPEG-21 terminal. Available documentation from this project will be used to improve the AXMEDIS specification, while possible synergies with that activity may be investigated.

**Research and development plan**

The activity performed in T4.1.1 will allow confirming and updating the structure of MPEG-21 Digital Item and DRM. The implementation of the Collector/Transcoder that will allow porting the content collected (crawled) from the CMSs into the AXMEDIS database will be addressed and realized into WP4.2 (see section 3.2.1 below).
As from the AXMEDIS general requirements (Deliverable 2.1.1 [3]), the AXMEDIS Content Model has to remain compatible with MPEG-21 and at the same time will have to add features to facilitate content management and production, as well as other details. To this end, it is planned to add an informative structure called AXInfo to each resource.
The main goals of this task are two:

- definition/selection of the content model, which shall be flexible enough to allow content production on demand, creation of content by a) automatic formatting tools and b) on the basis of profiles, styles, context, and content structure available (complete objects and components)
- definition/selection of a common format for modeling and representation of digital objects, both in binary form and (for some of the object's attributes) in textual form; the format will have to take into account specific optimizations (and possibly variations) for data archival and transmission

A loader and saver in XML will be implemented, since the XML language will be used to define the AXMEDIS file format for any digital content. The digital content will be identified by an XML (possibly binarized) file. The XML schema will be defined to fit the requirements of the project (DE 2.1.1) as described above. Standard (mainly MPEG) and proprietary content formats will coexist inside the AXMEDIS file format.

**Planned schedule**

- M9: realization of the XML-based interface between content management layer, terminal and authoring
- M10: Definition of the AXMEDIS data model and database model, first version
- M12: Implementations of the first version of tools for AXMEDIS information editing and viewing into the AXMEDIS model as add on at the AXMEDIS Editor
- M12: production of the first version of the AXMEDIS database and management interface
- M24: production of the revised version of the AXMEDIS database and management interface
- M24: Implementations of revised version of tools for AXMEDIS information editing and viewing into the AXMEDIS model as add on at the AXMEDIS editor
- M36: production of the final version of the AXMEDIS database and management interface

## 2.3 Specification of Design and development of an authoring tool for AXMEDIS content (DSI and EPFL)

**Major partners involved**

DSI, EPFL

**State of the art**

Even more than for MPEG-21 tools in general, the state of the art in authoring tools for MPEG-21 format is very poor. A tool has been produced by ENIKOS [4] which partially covers the MPEG-21 features and standard. Other similar tools to edit wrapped multimedia files/resources are those derived from the WEDELMUSIC Project [5].
In general, the necessary tools have to be easy to use and have to permit to the content producer the creation of AXMEDIS objects and the verification of the their consistency in terms of DRM, metadata, relationships, links, etc. In addition, the authoring tool has to be capable of guaranteeing the security of the digital resources when these are manipulated. For instance, if a digital resource can be accessed and modified only according to some rights, the editing has to be performed in a secure environment so that these rights can be guaranteed. To this end, it has to be possible to set up a mechanism to delegate the resource manipulation to external editors according to restrictions and verifications of their respect. On the other hand, the authoring of AXMEDIS objects has to be also possible by using other editors (such as widely used editors running in Windows OS, Acrobat Exchange, Photoshop, etc.) by supporting them with an AXMEDIS plug-in to guarantee the security of resources. All these aspects will be studied and prototypes will be implemented.

**Research and development plan**

This task will contribute the design and development of an authoring tool for AXMEDIS content (also called the AXMEDIS Editor in the main architecture), including a plug-in interface for using external editors and separate plug-ins to enable the MPEG-21 approach in other content editors.
The AXMEDIS authoring tool (editor and associated viewer) will support the MPEG-21 standard with related composition and the nesting of levels; it will also support the navigation into MPEG-21 objects. Typical features such as drag and drop mechanisms will be available to manipulate and create objects, as well as scripting. As mentioned above, this task will also develop innovative plug-ins for other multi- and cross-media authoring tools. This development will be partly related to WP4.2 in regard to item indexing and annotation and to WP4.3 for what relates to lower level object type descriptors, composition and formatting of objects.

Making use of the first basic content tools and software supports developed above, an MPEG-21 compliant terminal interface (client tools and Player) will be implemented for validation of the approach mainly in task 4.1.4 (see next section) reusing some of the blocks developed for the authoring tool.

The design of the Editor for AXMEDIS has to be flexible to allow the addition of other tools characterized by different user interfaces, coming from other partners and from external companies such as:

- DRM editor and viewer, with capabilities of verification and simulation
- Hierarchy editor and viewer
- Behavior editor and viewer
- Visual editor and viewer
- Metadata editor and viewer, for instance the AXInfo editor and viewer, or the UNIMARC editor and viewer
- Workflow editor and viewer
- Etc.

Of course the implementation of the above tasks and extensive assessment by integration in the AXMEDIS Framework (WP5) will constitute a notable platform case set to address MPEG and the MPEG Industry Forum with useful contributions, possible amendment requests and test results.

To summarize, research and development belonging to this subWP are related to the:

- AXMEDIS Editor (the authoring tool)
- AXMEDIS plug-ins in other editors or viewers, such as Windows Media, Acrobat Reader/Exchange, etc.
- AXMEDIS editors and viewers listed above
- AXMEDIS interfaces for communication to, and use of external players
- AXMEDIS internal editors/players, for documents, video, audio, etc.

**Planned schedule**

- M10: study of the availability, integration, and security level of other editing and playing tools such as Windows Media, Acrobat Exchange, Acrobat Reader, Photoshop, Macromedia tools, etc.
- M13: first version of an authoring tool for MPEG-21, including some of the above mentioned editors and viewers (hierarchy, behavior, etc.) in early version
- M13: first version of plug-in interface for the AXMEDIS Editor
- M15: First version of some of the plug-ins for other editors/players
- M18: additional study of the availability, integration, and security level of other editing and playing tools such as Windows Media, Acrobat Exchange, Acrobat Reader, Photoshop, Macromedia tools, etc.
- M24: second version of authoring plug-in interface

## 2.4 Specification of Implementation of an MPEG-21 compliant client terminal (EPFL, DSI)

**Major partners involved**

DSI, EPFL, SEJER

**State of the art**

As for authoring tools, the state of the art of Player tools for MPEG-21 is poor. In the above section editors have been mainly presented. Typically the players are subset of the editor while in addition they have a nicer and more functional user interface, more sophisticated customization possibilities, and possibly support different user profiles. In the AXMEDIS multichannel architecture the project partners will have to demonstrate the usability of AXMEDIS model for PC, i-TV, PDA and mobile, also in accordance to the models of the demonstrators (WP9, please refer to the last sections of this document for details).
State of the art will be continuously monitored by attending MPEG meetings and maintaining contacts with other parties active in MPEG-21 development. As said above, it is known that other European Projects are currently working on the development of MPEG-21 client/player terminals. Like for the definition of the

data model, possible synergies with these other consortiums and research centers active in the same area, especially IST ENTHRONE, will be established and will be evaluated for development and data exchange purposes, with cross-platform validations and conformance tests.
Outside the MPEG scope, additional technology will be explored and monitored for multimedia and cross-media composition and synchronization, like for instance SMIL that has a certain cross-compatibility with MPEG-4 systems.

**Research and development plan**

The first 18 months activity of this task will allow the development of the player interface and functionality on one selected test terminal (PC platform), while the porting to one or two additional cases will be achieved in the continuation of the project (for mobile and/or PDA, and MacOS, to be selected). In this context contributions to MPEG are foreseen. This task also includes the study and the integration analysis and design for the implementation of a viewer based on Mozilla development environment, according to SEJER needs.

User-oriented interfaces and functionality will be implemented able to parse MPEG-21 and AXMEDIS information and to instruct the terminal to behave accordingly, possibly reusing blocks from task 4.1.3 (including some plug-ins and players) and developing/adapting some new ones for non PC platforms. In particular, media players used and developed by AXMEDIS partners will be considered and a layer as portable as possible will be conceived and developed allowing different levels of complexity and sophistication.

The work performed in this task can be mainly subdivided in subtasks:

- Short analysis of the state of the art and review of technologies and tools which may offer technology and tools for the AXMEDIS Player. This activity is also in relationship with authoring technology that may be adopted and further developed for behavioral and visual editors and viewers of the AXMEDIS Editor described in the previous section.
- Reuse/adaptation of basic content tools and software supports developed in T4.1.3.
- Implementation of an MPEG-21 compliant client tool for validation of the approach. This terminal will be initially developed on and for PC platforms, to port it later to one or two selected mobile platforms to allow full validation. The client tools will exploit the capabilities of other content player tools by means of suitable plug-ins and specific interfaces such as ACTIVE X for Windows, etc. User-oriented GUI and customization features will also be implemented. The first 18 months will allow the development of a terminal for one selected test platform (Windows PC), while the porting to additional one or two cases will be achieved in the continuation of the project. This subtask will also contribute the study, integration analysis and design for a viewer based on the Mozilla development environment.
- Definition of the validation process for content distribution to terminals (see specification for WP 4.6, 4.7, 4.8 later) in respect to AXMEDIS business models and requirements. This validation process performed with different distribution and terminal cases will constitute a core experiment for seamless and augmented use of multimedia resources across a wide range of devices.

Like in the case of the Editor and even more, the implementation of the above tasks and extensive assessment by integration in the AXMEDIS framework (WP5) will constitute a notable platform case set to address MPEG and the MPEG Industry Forum with contributions, possible amendment requests and test specification and results.

To summarize, research and development belonging to this WP are related to the:

- AXMEDIS Player and end-user GUI
- AXMEDIS plug-in in other players and viewers, such as Windows Media, Acrobat Reader, etc.
- AXMEDIS plug-in into Mozilla
- AXMEDIS interfaces for communication to, and use of external players
- AXMEDIS internal players, for documents, video, audio, etc.

**Planned schedule**

- M13: draft version of a Player tool for AXMEDIS/MPEG-21, with some of the internal media players and viewers also in draft version
- M15: first version of some of the plug-ins for other players and viewers
- M18: first test Player and terminal implementation, with additional player features (PC platform)
- M20: final selection of additional platforms for porting of the AXMEDIS Player
- M24: second, final version of the plug-ins for other players and viewers
- M36: implementation of additional test terminal interfaces, validation experiments for different content distribution test cases

# 3 AXMEDIS Object Manager (AXOM) (DSI)

In this prototype the main features regarding the management of AXMEDIS objects are included. The demonstrator gives evidence of the production of the following sub-components:

- AXMEDIS Data Model Schema definition
- AXMEDIS/MPEG-21 Data Model implementation
- Command manager
- XML Loader and Saver

The purpose is to show the typical way of programming an application which use the AXOM in order to manipulate objects. It shows the typical cycle load-manipulate-save. This test is intrinsically performed on the other demonstrators which perform this operation, but this specific demo is needed to understand how the applications are built against the object model which has been realized.

## 3.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | XercesC (only for SAX interface) |
| References to other major components needed | Configuration manager |
| Problems not solved | • Move of AXMEDIS Object portions between different objects<br>• Copy of AXMEDIS Object portions inside the object<br>• Efficient management of file offset reference during Save operation<br>• Add only an element, and not a sub-tree<br>• Memory leaks<br>• Missing AXOID management (temporary and definitive IDs)<br>• DateTime management<br>• Unicode compliance |
| Configuration and execution context | Presence of sample object to be loaded |
| Programming language | C++ |

## 3.2 AXMEDIS Object Manager: The problems

Some of the desired functionalities have not been yet realized:

- To move an object portion into another object: the “move” command has been implemented only with references to the same object portions. This action can be performed by cut & paste.
- To copy a sub-tree of the object into another position of the object itself.
- The identifiers used inside the newly created objects are not managed at any level: mechanisms in order to obtain temporary AXOIDs have to be included together with the link to the AXOID generator service.
- The DateTime has not been managed. A candidate library has been identified (Boost); it includes datetime management and locales, but it has not been integrated yet.
- Unicode has not been taken into account until now. Especially when parsing data from XML only simple (8 bit) string are loaded in memory and manipulated. Further extensions have to be introduced in order to manage Unicode string data.

Other functionalities are present but they are not implemented in an efficient way:

- After a save operation the model is destroyed and reloaded after, this simplify some on-going update of file offset streams on digital resource (which reside on disk space and not in memory). A solution has been conceived based on the controlled updating of the target file offsets, but not yet implemented.
- The adding mechanism is not allowed to insert full hierarchy of object, but only leaf objects. This ensures a protection check at every add command. A new command can be build to allow adding recursively a hierarchy of objects.
- Memory leaks are due to the uncorrect usage of XMLTranscode function of XercesC library

## 3.3 AXMEDIS Object Manager: Work performed

Nowadays several achievements have been reached in this area, in particular, regarding the data model schema :

- A profiling of the MPEG-21 DID [6] standard has been created to represent AXMEDIS objects. This profile specifies how AXMEDIS objects have to be represented w.r.t. the MPEG-21 DIDL XML schema. The AXMEDIS Data Model Schema does not have the same degrees of freedom MPEG-21 DIDL has, however it is enough flexible to cope with the given requirements (DE 2.1.1 [3]) an AXMEDIS object is easier manageable than a generic MPEG-21 DI.
- During the definition of the AXMEDIS Data Model Schema, lacks of expressiveness have been found out in the MPEG-21 DIDL IPMP schema [7]. In particular, it was evident the difficulty to associate accessible metadata to MPEG-21 DIDL element once they have been protected. Due to the previous observation, suggested by DSI at 72nd MPEG Meeting in Busan (Korea) (M12084 - AXMEDIS EC project and data model), an MPEG-21 core experiment named "CE on the indexing of IPMP protected DIDL content" (N7198) has been activated. DSI, FUPF and EXITECH have taken active part to the CE thus obtaining the amendment of MPEG-21 IPMP DIDL during the 73rd MPEG Meeting in Poznan (Poland). In particular a placeholder for accessible metadata has been added to the XML representation of protected DIDL element (N7426). This placeholder will be used in AXMEDIS to exposes public metadata of protected content.
- A set of useful business-level metadata has been defined. Such metadata have been structured using an XML schema named AXInfo. AXInfo is a mandatory set of metadata which describes an AXMEDIS object thus enabling automatic processing (formatting, protection, etc…) of AXMEDIS content. Metadata contained in AXInfo have been defined on the basis of several requirements gathered from content producer, integrator and distributor.

Regarding the AXMEDIS data model:

- On the basis of the actually available MPEG-21 standard, a generic object model for MPEG-21 complaint document has been designed and realized. In particular, the model consists of the following parts:
    - A base class, named *MPEG21Element*, which represent the base class for all the elements of all the MPEG-21 schemas which have to be represented in our object model. This base class mainly provides generic functions for the navigation and the manipulation of an XML-like document.
    - A generic factory class which, accepting a namespace, an element name and a set of attributes, returns an instance of the class which represents the given element from the given namespace. This factory is composed by a set of sub-factory, one for each namespace which should be managed. These sub-factories should be initialized using a static function at the start-up of an application.
    - Actually, two hierarchies of classes representing as much MPEG-21 namespaces have been created and are fully integrated with the mechanism above. In particular, DIDL hierarchy and DII hierarchy have been realized. Other three hierarchies have been made (IPMP DIDL, IPMP General Descriptor and IPMP Descriptor) but they are not still fully complaint with the above mentioned feature.

The described model implements several relevant features, some of which are:
    - Manage any kind of content which can be contained in a statement element (text, MPEG-21 XML and generic XML)
    - Manage embedded assets (e.g. within resource elements) as stream of bytes because an asset could be huge w.r.t. available memory in a device, e.g. a film encoded in MPEG2 could easily reach dimensions in the order of gigabytes.
    - Usage of event to report changes in the object model. That is, each object of the MPEG-21 model throws an event every time something changes, e.g. an attribute has been set, a child element has been removed, etc. These events are received by a "document" class instance (i.e. and object which represent the MPEG-21 document as a whole) which sends the events to those object which have been previously registered as document listeners.
- An upper-level model has been developed to represent the AXMEDIS data model. It is based on the MPEG-21 model, i.e. it is a "view" over an MPEG-21 model. Since AXMEDIS Data Model Schema is simpler than the MPEG-21 schemas, the AXMEDIS object model is simpler than the MPEG-21 object model. In particular, the AXMEDIS object model is composed of three main classes (instead of the fifteen and more classes of the MPEG-21 model): one representing an AXMEDIS object, one representing a resource and one representing generic metadata. The latter is specialized for the mandatory type of metadata adopted by AXMEDIS: AXInfo, Dublin Core and MPEG-21 DII [8]. This model allows manipulating an AXMEDIS object without minding to the complexity of the underlying MPEG-21 object.

It has to be noticed that the proposed MPEG-21 data model architecture is extremely flexible and easily expandable. In fact, who wants to enhance the data model with elements belonging to a (not yet implemented) MPEG-21 namespace should simply create:

- a hierarchy of classes representing these elements
- a factory class which is able to instantiate the above mentioned classes on the basis of name and attributes
- specific loader(s), if needed, for those elements (or group of elements) which needs specific actions during the load
- a writer which is able to write in XML a given instance of a class belonging to the related hierarchy.

The classes of the latter three points have to be registered to specific manager during the start-up of the application. In particular the classes have to be respectively registered to *MPEG21ElementFactory*, *MPEG21Loader* and *MPEG21ElementWriter*.

Regarding the AXMEDIS Command Manager:

- An uncoupling layer has been placed between the user and the data models (MPEG-21 and AXMEDIS); in this case "user" stands for a software developed to interact with an AXMEDIS object. This uncoupling layer, named *AxObjectManager*, allows controlling and regulating the access to the content since each action passes through specific check points. These check points will be used to integrate DRM and IPMP mechanism in an easy way.
- A set of commands to interact with the *AxObjectManager* have been created (actually two sets of commands exist, one operating on the MPEG-21 model and the other operating on the AXMEDIS one). These commands allow to do all the fundamental actions such as navigate the document; manipulate the document structure, access assets, etc. A command is an active object, i.e. a command has to expose the method *execute* which implements the real function of the command. The *execute* method accepts three parameters which allow the command to interact with the model. These parameter cannot be obtained from outside the *AxObjectManager* therefore the *execute* method can be effectively called only by an instance of *AxObjectManager*.

The technology which is needed to load and save has been development with the following features:

- An XML loader for MPEG-21 DI has been designed and realized. The loader is based on the SAX2 technology thus allowing to save computational time and space. The loading mechanism has been thought as the cooperation of different loader each of which has specific knowledge. In particular, a

generic loader has been created named *MPEG21ElementLoader*. This loader uses the above mentioned general factory to create an MPEG-21 element each time it receives an event of start element from the SAX2 parser. Since some elements/set of namespace require specific techniques to be loaded, the generic loader can be configured to delegate its task to another object every time the predefined conditions are verified.

- To cope with the issue of AXFW portability, a generic XML writer has been realized named *XMLWriter*. It exposes methods which allow writing an XML document such as *writeStartElement*, *writeEndElement*, etc. Moreover, *XMLWriter* automatically manages the namespace declarations avoiding the user to worry about this problematic task.
- Based on the *XMLWriter*, a set of writers have been created for the MPEG-21 model hierarchy. A writer is a class which owns the knowledge about how to write in XML the elements which belong to a given namespace (DIDL, DII, IPMP DIDL, etc…). Those writers are driven by a generic writer. The specific writers have to be registered in the generic one at the application start-up. Please notice that the pattern applied to the writers hierarchy is the same applied to the factories hierarchy.

# 4 AXMEDIS Object Preprocessor and Postprocessor (EPFL)

The set of tools implementing the AXMEDIS Objects Preprocessing and Postprocessing provide two main functionalities. On one hand, conversion from textual XML to binary form and vice-versa has to be implemented in order to allow both objects presentation in a human readable (and editable) format and objects download or stream through bandwidth-critical channels. On the other hand, when objects are converted from a binary format to textual XML on the end-user terminal, the existent references to external objects must be resolved.

## 4.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | Xerces |
| References to other major components needed | |
| Problems not solved | • Support of the Xinclude schema for references resolution |
| Configuration and execution context | |
| Programming language | C++ |

## 4.2 Object Preprocessor and Postprocessor: State of the art

As a textual format, XML tends to be rather verbose since it has not been designed to work ideally in a real-time, constrained and streamed environment such as those found in the i-TV, in the multimedia or in the mobile industry. As long as structured documents (HTML, for instance) were basically composed of only few embedded tags, the overhead induced by textual representation was not critical. Applications are now dealing with larger and highly structured XML documents. More and more XML files are exchanged in many fields such as in i-TV, database synchronization or enterprise application integration (EAI). In these cases, the XML textual representation is verbose and its processing very inefficient.
Therefore, optimized binary encoding mechanisms have been developed in order to reduce the bandwidth needed to transmit XML objects when necessary. To overcome this lack of efficiency of textual XML, MPEG-7 Systems has defined a generic framework to facilitate the carriage and processing of MPEG-7 descriptions: BiM (Binary Format for MPEG-7) [17]. It enables the streaming and the compression of any XML documents. Among the advantages of BiM there is its flexibility. In fact, a BiM decoder can deal with evolution of XML languages. Technically, at encoding phase a level of compatibility is chosen for the

bitstream. The encoding process adds necessary information to ensure that an old decoder will be able to skip unknown part of the bitstream. This feature allows also XML private extensions to be easily inserted within the original XML document without breaking interoperability. If forward compatibility is not needed, the redundancy is removed and the bitstream becomes more compact. Given the proliferation of MPEG standards and other external organizations referring to MPEG-7 Systems BiM technology, it seemed that a dedicated standard would considerably ease the referencing and maintenance of the standard independently of the MPEG-7 Systems specification. These are the reasons which led to the creation of a new project of the MPEG group (ISO/IEC WG11): the so called MPEG-B (23001-1) MPEG-B -- Part 1: Binary MPEG format for XML.

Open source implementations of BiM include:

- the BIM Open Source project which is entirely Java based and composed by four different subprojects. The first project is the BIM Database Standard Edition, a personal information manager. The second project is the BIM Dynamic Development Environment. The third project is the BIM Telnet Client. The telnet client is a smaller scale project. It is useful for storing aliases, triggers, patterns, and some artificial intelligence in a basic telnet interface. The fourth project is the BIM Obfuscator used to obfuscate Java code variables and methods. It basically protects name integrity across packages.
- The MPEG-B reference software is available but its efficiency is very limited. It can be used for testing the output of other implementations but it could hardly be integrated in larger frameworks which may need to use it as a plug-in.
- The most efficient BIM implementation is the EXPWAY BiM commercial implementation called BinXML™ [20]. This product is very efficient in terms of compression performances provided. On the MPEG-7 test set, BinXML™ reaches an average compression ratio of 85 %. The average number of bits per elements and attribute is 2.93 bits (min = 0.003, max = 13, standard deviation = 1.76) against 136.7 bits for the original XML files.

The only open source implementation in C++ of BiM is then the MPEG reference software. If its efficiency should not be adequate to the AXMEDIS purposes another solution called XMill is available. XMill is a special-purpose compressor for XML data that typically achieves twice the compression rate over existing compressors, such as gzip. MILL is an open source program delivered under a BSD License, it is written in C++, and it is available for all 32-bit MS Windows (95/98/NT/2000/XP), and all POSIX (Linux/BSD/UNIX-like OSes) [21] platforms.
Similar to gzip, XMill is a command-line tool that works on a file-by-file basis. A given file with extension '.xml' is compressed into a file with extension '.xmi'. Any other file without extension '.xml' is compressed into a file by appending extension '.xm'. Reversely, the original file is obtained by replacing extension '.xmi' with extension '.xml' or by removing extension '.xm'. Alternatively, the user writes the output to the standard output and optionally read from the standard input.

A widely deployed open source project for XML validation, authoring and manipulation is the Xerces toolset. Xerces (named after the Xerces Blue butterfly) provides XML parsing and generation. Currently, there are two validating versions, in Java and C++, implementing the W3C XML and DOM (Level 1 and 2) standards, as well as the de facto SAX (version 2) standard. The parsers are highly modular and configurable. There is also some support for XML Schema (draft W3C standard). Xerces-C++ is a validating XML parser written in a portable subset of C++. It provides a shared library for parsing, generating, manipulating, and validating XML documents.

## 4.3 Object Preprocessor and Postprocessor: The problems

At the 71st MPEG meeting held in Hong Kong (China) the MPEG-21 DID standard [6] was amended to substitute the REFERENCE element with the W3C Xinclude schema. This is due to the fact that in December 2004 W3C published the XML Inclusions (XInclude) 1.0 Recommendation (see W3C XINCLUDE). The functionality provided by the REFERENCE element in the first edition of ISO/IEC 21000-2 can be substantially provided by XInclude. XInclude also provides functionality that extends beyond that of the REFERENCE element. For example, the document author can provide a fallback in case the referenced elements are unable to be retrieved. Hence in this second edition the REFERENCE element has been removed and similar functionality is achieved using XInclude.
Some important differences between REFERENCE and XInclude have to be considered when using XInclude instead of REFERENCE. These include:

1. A REFERENCE element operates on its parent element, in that the content of the referenced element is included into the content of the parent element of the REFERENCE element. Whereas an XInclude include element operates on itself, in that the included content replaces the XInclude include element.

2. XInclude processing can result in addition of xml:base and xml:lang attributes to the top-level included elements.

3. XInclude processing does not include the attribute value inheritance of the REFERENCE element.

The above mentioned amendment to the MPEG-21 schema requires an extension of the Xerces capabilities in order to support the validation of the Xinclude schema. In particular, the Xerces Schema Validator needs to be extended to accept INCLUDE element coming for XInclude Namespace. A normal schema validator cannot manage this element since they are xml:base and xml:lang attributes. A special validator has to be provided in order to validate the XInclude syntax and the whole schema by considering the referred elements

## 4.4 Object Preprocessor and Postprocessor: Work performed

The work for the development of the AXMEDIS Reference Resolver is based on the Xerces toolset for the authoring and manipulation of XML files. The application opens an XML file and searches for references inside it. So far, a reference is defined as something like this:

```
<reference file="C:\referenced_file.xml"  XPath="node[2]/something[0]/node[3]/something">
```

where:

- the `file` attribute is the referenced file.
- the `XPath` attribute is the path inside the referenced file of one specific node. This path is in format XMLPath.

Then the Reference Resolver opens all the referenced files and searches for the referenced nodes and, finally, it substitutes the link or reference with the referenced element.

The following snapshot displays the user interface of the reference resolver.

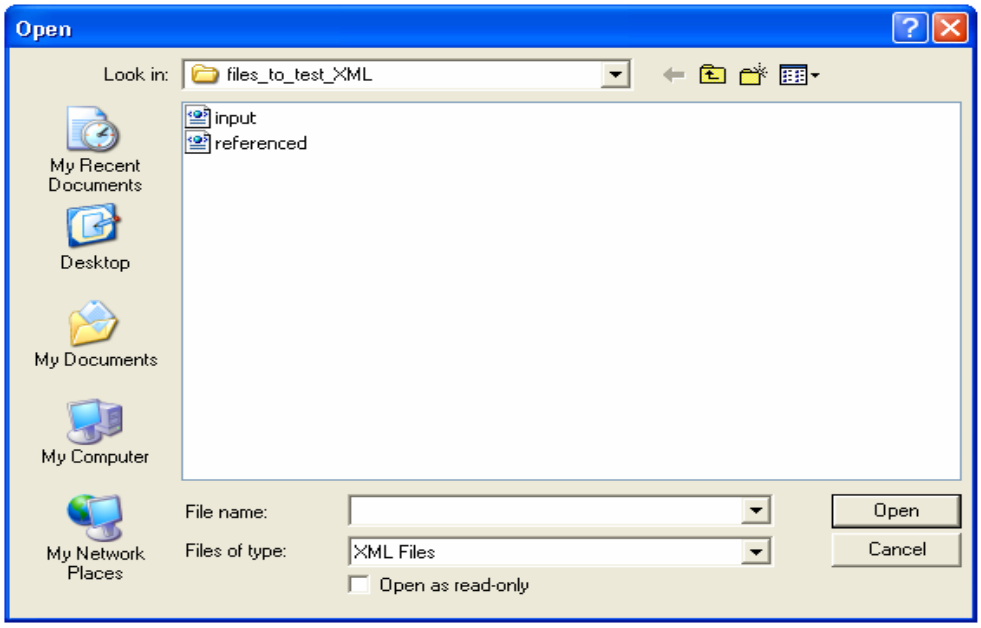



**Figure 1 - The user is required to open the input file which is referencing the "referenced.xml" file**

Suppose the input XML is named as input.xml and the referenced XML file is named as referenced.xml. The input XML file has the following content.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE personnel>
<!-- DOCTYPE personnel SYSTEM "personal.dtd"-->

<!-- @version: -->

<personnel>

  <person id="Big.Boss" >
    <name><family>Boss</family> <given>Big</given></name>
    <email>chief@foo.com</email>
    <link subordinates="one.worker two.worker three.worker four.worker
five.worker"/>
  </person>


  <person id="one.worker">
    <reference file = "referenced.xml" XPath = "/AUCTIONBLOCK/ITEM/BIDS/BID[0]" />
    <name><family>Worker</family> <given>One</given></name>
    <email>one@foo.com</email>
    <link manager="Big.Boss"/>
  </person>

  <person id="two.worker">
    <name><family>Worker</family> <given>Two</given></name>
    <email>two@foo.com</email>
    <link manager="Big.Boss"/>
  </person>

  <person id="three.worker">
    <name><family>Worker</family> <given>Three</given></name>
    <email>three@foo.com</email>
    <link manager="Big.Boss"/>
  </person>
```

```
  <person id="four.worker">
    <name><family>Worker</family> <given>Four</given></name>
    <email>four@foo.com</email>
    <link manager="Big.Boss"/>
  </person>

  <person id="five.worker">
    <reference file = "referenced.xml" XPath =
"/AUCTIONBLOCK/ITEM/BIDS/BID[3]/BIDDER" />
    <name><family>Worker</family> <given>Five</given></name>
    <email>five@foo.com</email>
    <link manager="Big.Boss"/>
  </person>

</personnel>
```

The referenced XML file has the following content:

```
<?xml version="1.0"?>
<AUCTIONBLOCK>
   <ITEM>
      <TITLE>Vase and Stones</TITLE>
      <ARTIST>Linda Mann</ARTIST>
      <DIMENSIONS>20x30 inches</DIMENSIONS>
      <MATERIALS>Oil</MATERIALS>
      <YEAR>1996</YEAR>
      <DESCRIPTION/>
      <PREVIEW-SMALL SRC="images/burl-s.jpg" WIDTH="300" HEIGHT="194" ALT="Vase and
Stones"><!--The small image--></PREVIEW-SMALL>
      <PREVIEW-LARGE SRC="images/burl.jpg" WIDTH="640" HEIGHT="413" ALT="Vase and
Stones"/>
      <BIDS>
         <BID>
            <PRICE>3300</PRICE>
            <TIME>9:19:32 AM</TIME>
            <BIDDER>John</BIDDER>
            <TIMESTAMP>1315</TIMESTAMP>
         </BID>
         <BID>
            <PRICE>3200</PRICE>
            <TIME>8:18:31 AM</TIME>
            <BIDDER>Andrew</BIDDER>
            <TIMESTAMP>1308</TIMESTAMP>
         </BID>
         <BID>
            <PRICE>3100</PRICE>
            <TIME>2:48:08 PM</TIME>
            <BIDDER>Chris</BIDDER>
            <TIMESTAMP>1307</TIMESTAMP>
         </BID>
         <BID>
            <PRICE>3000</PRICE>
            <TIME>2:47:58 PM</TIME>
            <BIDDER>opening price</BIDDER>
            <TIMESTAMP>1298</TIMESTAMP>
         </BID>
      </BIDS>
      <MTIME>
         <TIMESTAMP>1315</TIMESTAMP>
         This is an example of mixed content
      </MTIME>
   </ITEM>
</AUCTIONBLOCK>.
```

In the input XML file there are two parts associated with the referenced XML file. They are the following in the input XML file.

```
<reference file = "referenced.xml" XPath = "/AUCTIONBLOCK/ITEM/BIDS/BID[0]" />
```

and

```
<reference file = "referenced.xml" XPath =
      "/AUCTIONBLOCK/ITEM/BIDS/BID[3]/BIDDER" />
```

After resolving, the output file replaces these two parts with the corresponding elements in the referenced XML file. For example:

```
<reference file = "referenced.xml" XPath = "/AUCTIONBLOCK/ITEM/BIDS/BID[0]" />
```

This part is replaced by the following code from the referenced XML file. The result is demonstrated in the following snapshot.

```
<BID>
  <PRICE>3300</PRICE>
  <TIME>9:19:32 AM</TIME>
  <BIDDER>John</BIDDER>
  <TIMESTAMP>1315</TIMESTAMP>
</BID>
```

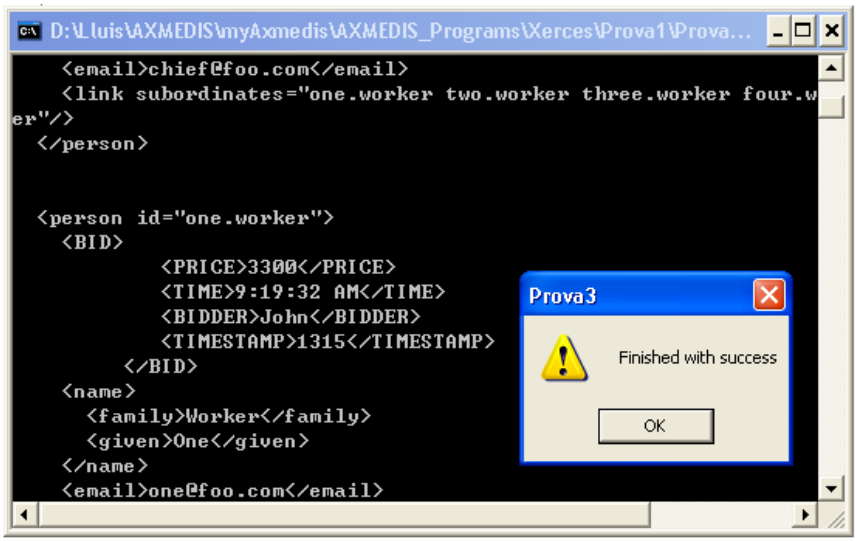



**Figure 2 - Output of the reference resolver after processing input.xml**

Following the recent modifications of the MPEG-21 DID standard the current version of the resolver described above needs to be modified and extended accordingly.

# 5 AXMEDIS Editor (DSI)

The AXMEDIS Editor should allow the user to create and manipulate AXMEDIS objects.

## 5.1 Technical Details

| | |
|---|---|
| Reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | wxWidgets |
| References to other major components needed | AXOM, Hierarchy Editor & Viewer, Internal Image Viewer, Internal Audio Player, Internal |

| | |
|---|---|
| | Video Player, Internal Document Viewer |
| Problems not solved | |
| Configuration and execution context | |
| Programming language | C++ |

## 5.2 Editor: Work performed

The AXMEDIS editor allows to:

- load and save an unprotected AXMEDIS object as a xml file
- create a new AXMEDIS object
- visualize and manipulate the hierarchy of both the AXMEDIS and MPEG-21 (see Hierarchy Editor and Viewer)
- view resources (images, documents, video, audio) in an AXMEDIS object using the internal resource viewers
- view metadata in xml (not using the MetadataViewer)

The following is the AXMEDIS editor window for a new object, on the left are present the hierarchy views (AXMEDIS and MPEG-21) and on the right the different views on an element.

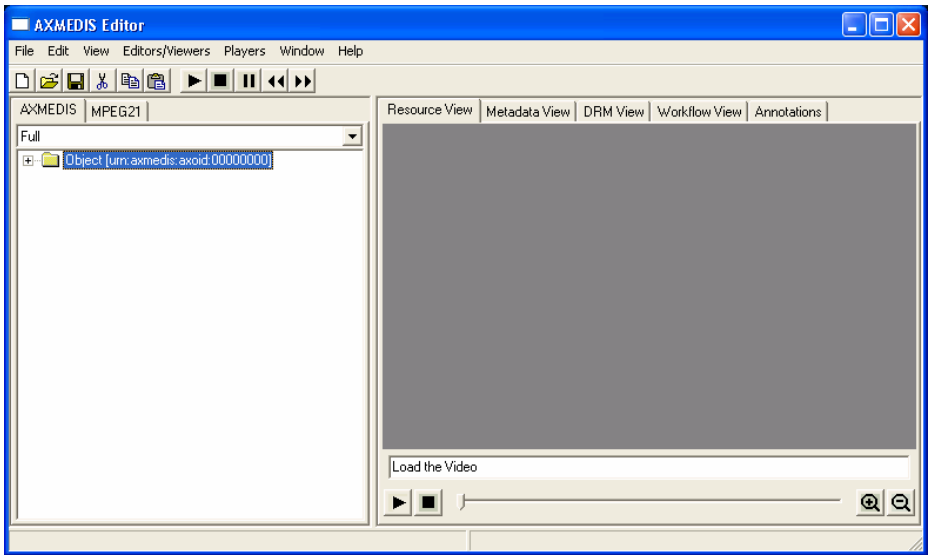



**Figure 3 - AXMEDIS editor window for a new object**

In the following picture an AXMEDIS object was loaded and a resource is opened (double click)

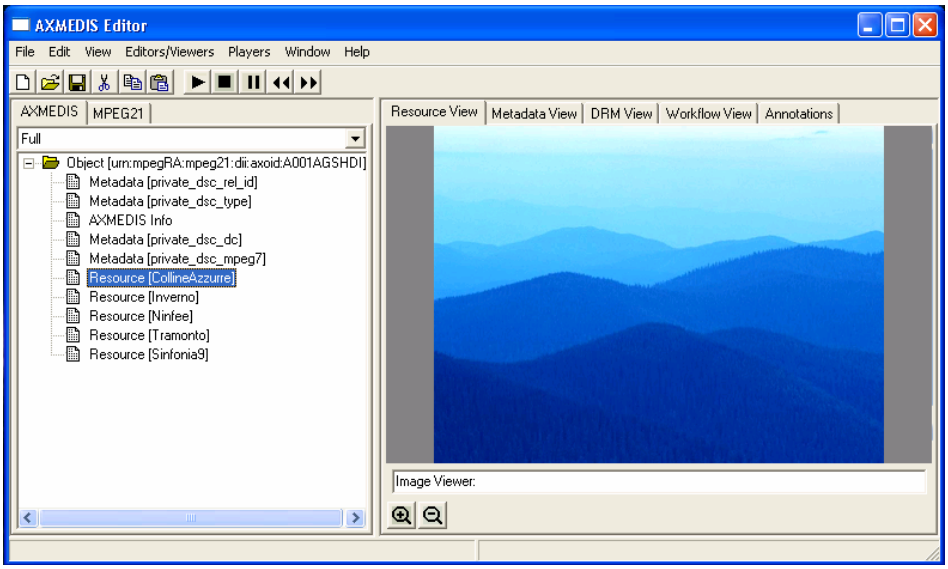



**Figure 4 - An AXMEDIS object is opened and the resource is rendered**

In the following image is opened a metadata element of an object, by now the xml description is presented (debug only)

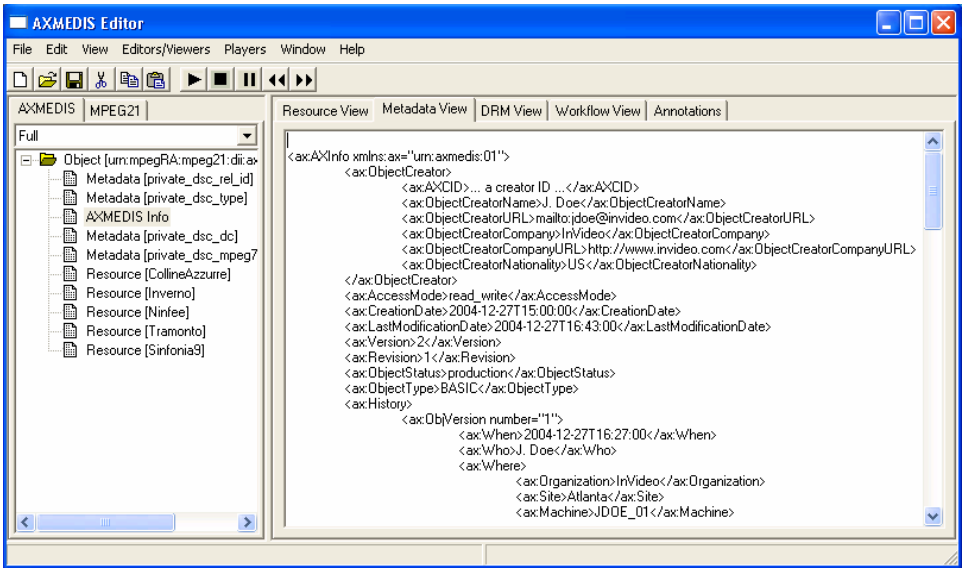



**Figure 5 - The AXMEDIS editor displays a metadata element.**

In the following image the contextual menu associated with an element is shown

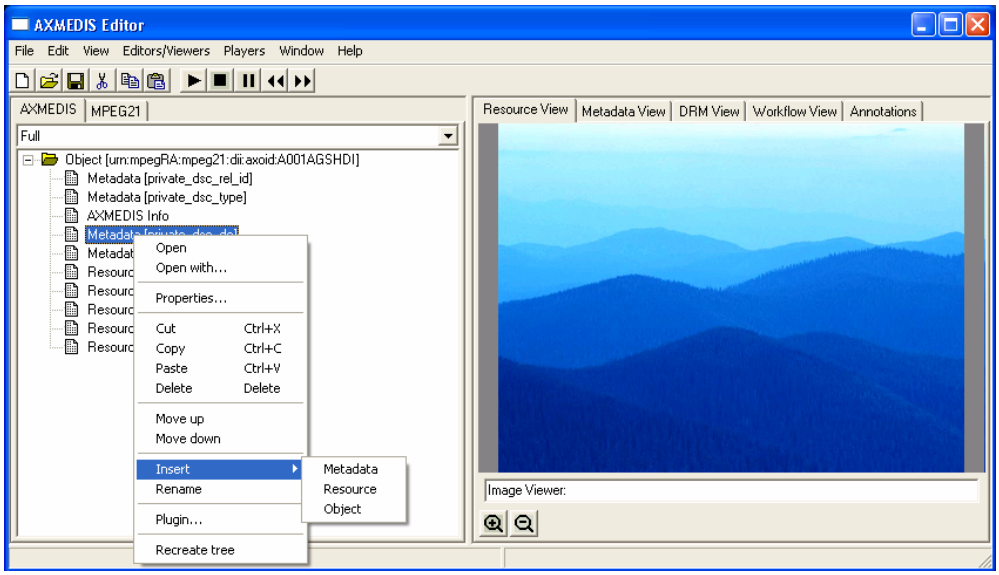



**Figure 6 - Contextual menu**

In the following image properties of a resource are shown

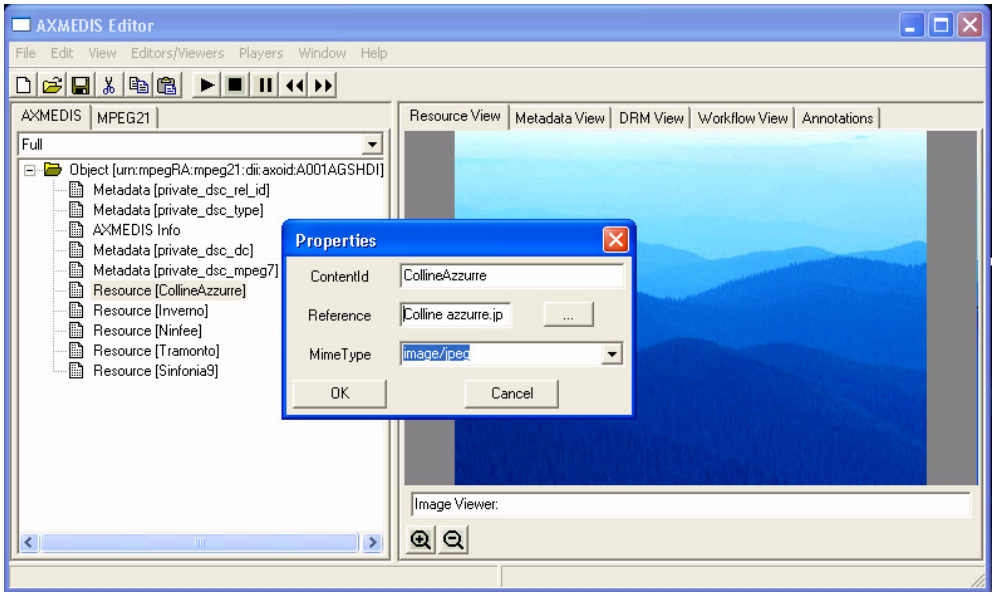



**Figure 7 - Resource properties are shown**

In the following image the MPEG-21 hierarchy is shown and the MPEG-21 resource is opened

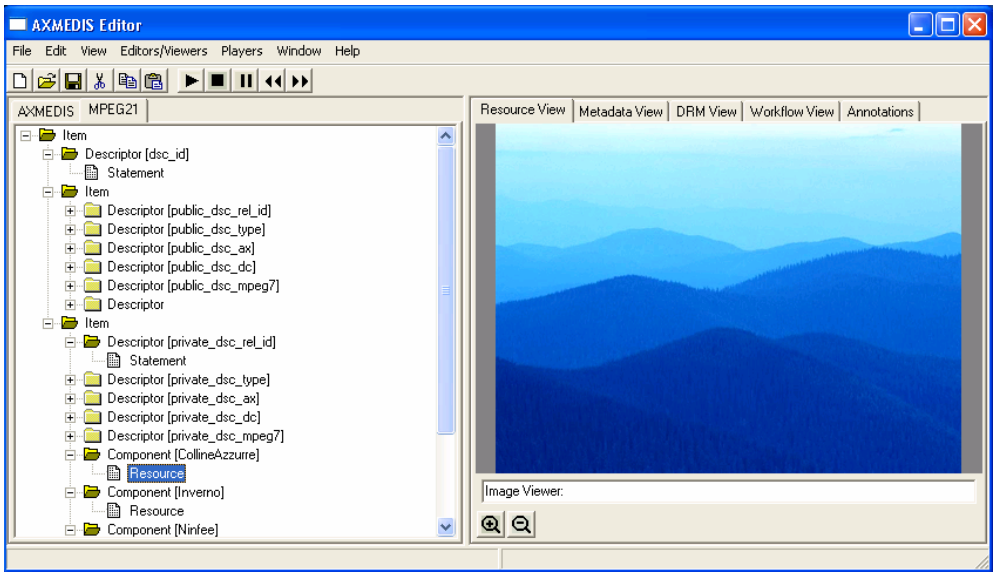



**Figure 8 - MPEG-21 hierarchy and resource rendering**

# 6 Hierarchy Editor and Viewer (DSI)

The Hierarchy Editor and Viewer show the AXMEDIS and MPEG-21 hierarchy and allow manipulating them.

## 6.1 Technical Details

| | |
|---|---|
| Reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | wxWidgets |
| References to other major components needed | AXOM |
| Problems not solved | |
| Configuration and execution context | |
| Programming language | C++ |

## 6.2 Hierarchy Editor and Viewer: Work performed

The AXMEDIS Hierarchy Editor and Viewer allows to:

- visualize the AXMEDIS hierarchy as tree structure
- add a new AXMEDIS resource, AXMEDIS metadata or AXMEDIS object to an AXMEDIS object
- remove any element inside the object
- move the elements up or down
- drag & drop elements to move elements inside the object or from an object to another one
- drag & drop a file on the hierarchy to automatically add a resource
- copy & paste

The MPEG-21 Hierarchy Editor and Viewer allows to:

- visualize the MPEG-21 hierarchy as tree structure
- add a new MPEG-21 element (container, item, descriptor, etc.)
- remove any element inside the object
- move the elements up or down
- drag & drop elements to move elements inside the object or from an object to another one

The following is an example of AXMEDIS Hierarchy:

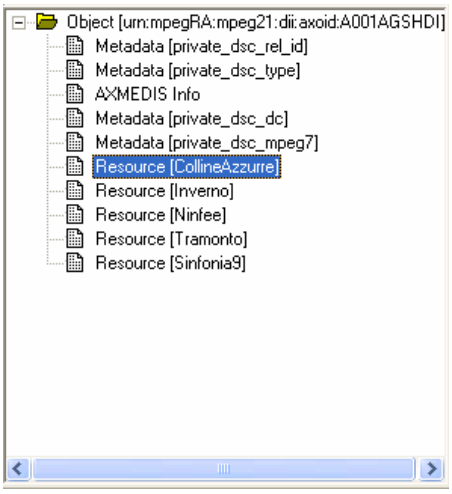



**Figure 9 - Example of AXMEDIS hierarchy**

The following is an example of MPEG-21 Hierarchy:

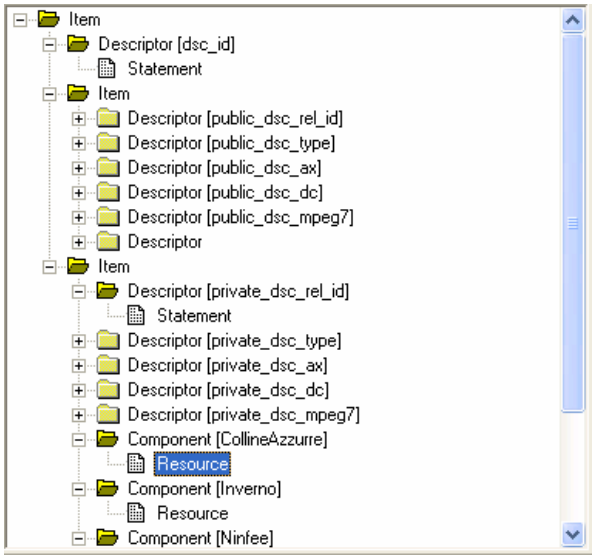



**Figure 10 - Example of MPEG-21 hierarchy**

see AXMEDIS Editor section for examples using the Hierarchy Editor & Viewer

# 7 Visual Editor and Viewer (EPFL)

The Visual Editor will arrange the media objects (video, photo, text, etc) on the screen. The Visual Editor allows object placement in a 2-D (or possibly 3-D) environment. Moreover, it permits managing (i.e. moving, deleting, adding, etc…) object subparts which have spatial properties or constraints. The layout description will be done according to the SMIL W3C standard [9].

## 7.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | wxWidgets 2.4.2, Xerces |
| References to other major components needed | AXOM |
| Problems not solved | • Attachment of media content to shapes<br>• More precise editing functionality (attributes) |
| Configuration and execution context | |
| Programming language | C++ |

## 7.2 Visual Editor and Viewer: State of the art

SMIL authoring tools are implemented by different companies and consortia. Some of them are commercial tools whereas some are also available as free software or even open source toolsets.
SMIL authoring tools implemented in C++ could not be found as open source projects. A powerful SMIL authoring tool commercialized by Oratrix [10] – GriNS2 - has been examined and can be used as reference implementation supporting most of the functionality needed by AXMEDIS. On the other hand, the most interesting open project is LimSee2 developed by INRIA [11] and it is implemented in Java. LimSee2 can also be used to test the conformance of the implemented Visual Editor and Viewer. The following sections provide a brief overview and some snapshots of the GRiNS and LimSee2 features.

### 7.2.1 GRiNS Pro Editor for SMIL 2.0

GRiNS Pro Editor for SMIL 2.0 is a SMIL authoring application developed by Oratrix Software Development Company for creating, editing and maintaining streaming media documents for the RealOne player and also for other SMIL 2.0 compliant players, such as Internet Explorer6.0 and 3GPP. GRiNS Pro Editor can take source materials such as audio, video, image and text assets, and integrate them into a presentation which can be distributed via the Web.

GRiNS has an internal layout editor which manages visual and auditory resources. When working with template-based presentations, the template contains a number of regions to organize screen and other resources in the presentation. When the content creator uses these regions, he will typically not need to visit the Layout view. For more complex presentations, or when creating custom templates, the Layout view provides a means of creating, sizing, editing and placing new regions.

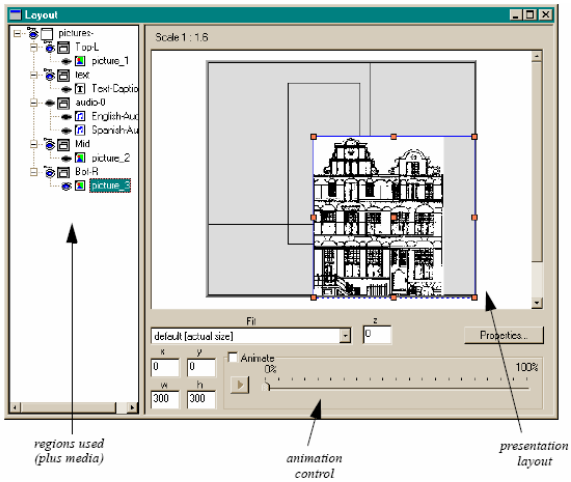



**Figure 11 - Layout of the GRiNS editor**

When activated, the Layout view shows a window similar to the GR*i*NS Player window and a separate editing window for selecting layout regions.
More information is available on the Oratrix Web site [10].

### 7.2.2 LimSee2

LimSee2 is an open authoring application for SMIL 1.0 and 2.0 languages conducted by the WAM team (Web Adaptation Multimedia) at INRIA, Grenoble, France.

LimSee2 has a multi-view solution (Layout view/Attributes view/Structure View/Timeline view) that renders the structure of the SMIL document at different levels during the authoring process: timing and synchronization, spatial layout, XML tree etc. The different views are synchronized (a modification in one view is immediately rendered in all the other views) and provide functionality that allows a user to manipulate and fine-tune a SMIL document without requiring a full knowledge of the language.

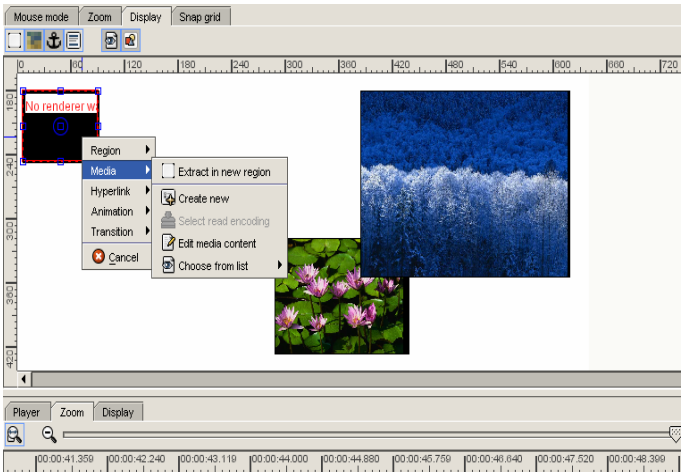



**Figure 12 - Layout of the LimSee2 editor**

The spatial layout of a SMIL document can be edited in LimSee2 in a 2D canvas, which constitutes a WYSIWYG (What You See Is What You Get) environment for a fixed time in the temporal scenario. SMIL regions can be easily moved, resized or created in a few clicks, media content can be directly previewed (for images, texts and videos), and region z-indexes can be adjusted with an intuitive drag-and-drop mechanism. The 2D canvas also provides traditional features such as a zooming tool or a customizable snap grid.
More information is available on the LimSee2 Web site at INRIA [11].

## 7.3 Visual Editor and Viewer: The problems

The above described toolsets are one a commercial application (GRiNS) and the other a free JAVA application (LimSee2). In the open source community few are the projects aiming at developing SMIL editors in C++ and they are all in a very early stage (still planning, no source code produced). AXMEDIS need then to build its own SMIL editor, an open C++ application, with functionality similar to the one of LimSee2 or GRiNS.

## 7.4 Visual Editor and Viewer: Work performed

### 7.4.1 The Visual Editor

The work on the AXMEDIS Visual Editor has been started. It is based on the open WxWidgets library for cross-platform graphical user interfaces implementation. The state of the Visual Editor is the following:

- The program allows the user to draw rectangles or circles on the screen with the mouse. The shapes can also be resized, moved or deleted.
- Every shape corresponds to an SMIL Region that will be a placeholder for any media content: text, video, animation.
- The program is able to save the documents in XML-SMIL format.
- The library Xerces-C++ is used to deal with the XML files.
- The GUI is implemented with wxWidgets version 2.4.2.

The following picture shows a snapshot of the available tool.

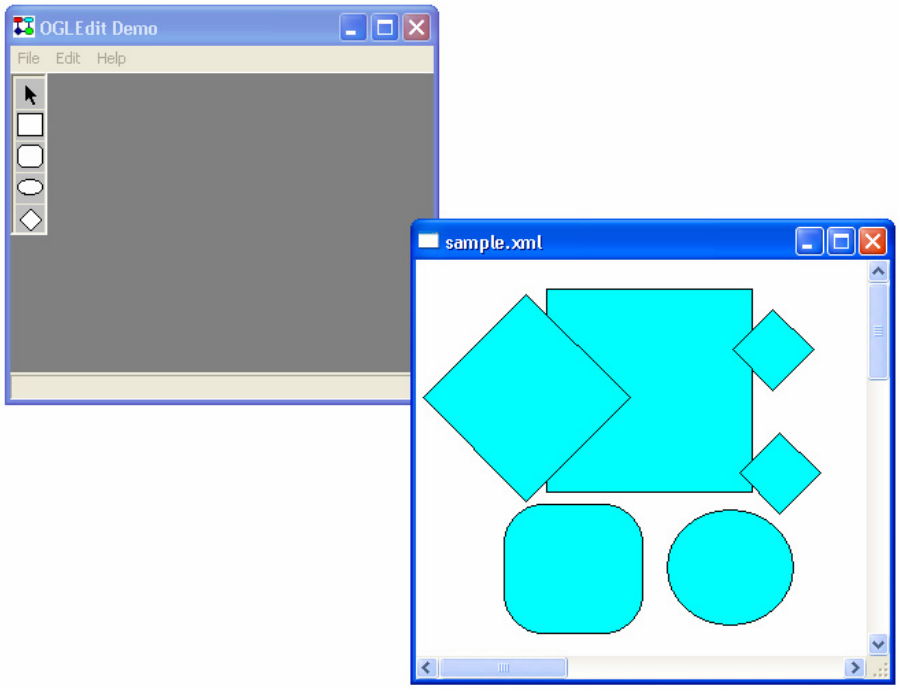



**Figure 13 - Visual editor and viewer layout**

Currently, the functions of the tool are rather elemental; they need to be extended in order to implement a powerful SMIL editor comparable to LimSee2 or Grins.
The main functionality in course of development and integration are now briefly described.

Every shape corresponding to an SMIL region and used as a placeholder for any media content (text, video, animation), shall be linked to the chosen resource through straightforward commands (file browsing or visual drag&drop). The following snapshot shows how the visual editor shall be used to add a media object to a rectangular area.

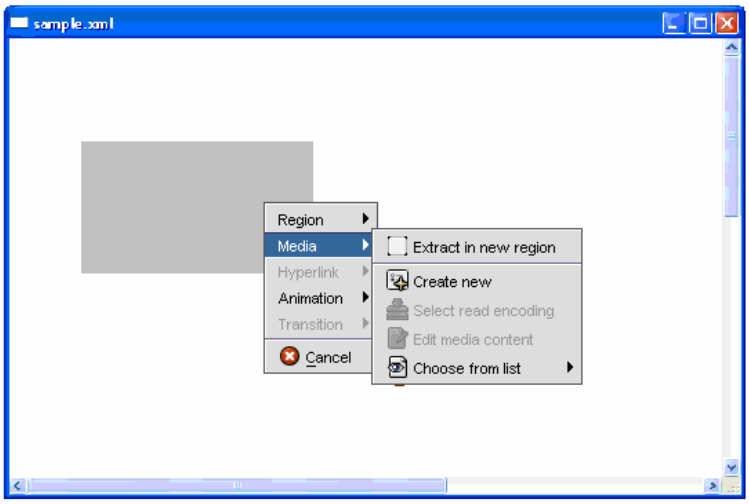



**Figure 14 - Contextual menu to be used to add objects to the scene**

After choosing a given region, a media object can be linked by right clicking the corresponding shape and choosing *create new* media object. At the same time the business logic will add the source of the media object to the SMIL file. The following is a snapshot of a sample SMIL file of the media object which defines the position and the source of the image "Water.jpg" associated with the region.

```
<img xmlns="http://www.w3.org/2001/SMIL20/Language"
     fit="meet"
     dur="5"
     src="Water.jpg"
     region="region_dufbu"
     id="img_rlto"/>
```

**Figure 15 - Excerpt of a sample SMIL file**

### 7.4.2 The Viewer

The editor described above needs to be integrated with a SMIL player to be invoked for example when a content creator needs to pre-view the authored content. This will enable the AXMEDIS editor to directly play SMIL objects without the need of calling an external player. A SMIL player has been implemented both to this end and to be distributed as (part of) the AXMEDIS PC player.
Section 14.2.2 describes in detail the current version of the stand-alone SMIL player based on the AMBULANT open source project. This version of the player will be integrated in the Visual Editor in a modular way allowing the extraction of a down-scaled version of the tool providing only the player's features. This may be implemented by means of the definition (or un-definition) of a few environment variables enabling the compilation of only those parts of the code implementing the SMIL player functionality.
For more details on the current version of the stand-alone player please see paragraph 14.2.2.

# 8 Behaviour and Functional Editor and Viewer (EPFL)

The Behaviour Editor together with the Visual Editor will provide the user with the infrastructure to produce multimedia presentations. A multimedia presentation can be composed of many media objects (text, audio, video, vector graphics...). The user will use the Behaviour and Visual Editors to organize the media objects in space and time. The Behaviour Editor will complement the Visual Editor by adding time boundaries to the media objects. This means that every media object will be visible only for a period defined by the user. The simplest example for this is a slide show: the user specifies a group of slides and each slide is only visible during a slot of time defined by the user.
The Behaviour Editor will use a subset of the SMIL language to describe the multimedia presentation.

## 8.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Not available yet |
| List of libraries used | wxWidgets 2.4.2 |
| References to other major components needed | AXOM |
| Problems not solved | No open source projects are available in C++. Development has to start from scratch. |
| Configuration and execution context | |
| Programming language | C++ |

## 8.2 Behavior and Functional Editor: State of the art

At the heart of any SMIL implementation lays a module in charge of managing the Timing and Synchronization of the decoded scenes. Orchestrating the timing of the various media elements in the presentation is, after all, what makes SMIL different from traditional Web media.
The following paragraphs describe how timing and synchronization management is managed in the two already examined SMIL authoring suites: GRiNS and LimSee2.

1. GRiNS:

As one of the four views supported by the GRiNS SMIL authoring environment, the Timeline View is for editing the SMIL file in terms of time structure. The hierarchical structure is intended to represent the coarse timing relationships reflected in the `<par>` and `<seq>` constructs. While the attributes associated with an element can be used to define more fine grained relationships which are often difficult to visualize with the hierarchical view alone. For this reason GRiNS also supports a timeline projection of a presentation. Unlike some systems which use the timeline as the initial view of the application, the GRiNS timeline displays relationships derived from the hierarchical structure. This means that the user is not tied to a particular clock or frame rate since the actual timing relationships will only be known at execution time. When working with the timeline view the user can define exact start and end offsets within `<par>` groups by using a synchronization arc shown as an arrow.

2. LimSee2:

LimSee2 is one of the most efficient authoring tools in terms of editing the SMIL time structure. The timeline view is a faithful representation of the temporal behavior of the document. In this view the author can perform all editing operations related to synchronization between elements. The mouse cursor informs the user on the editing operations he can perform on the selected SMIL element. When the cursor takes the shape of a cross, an user action moves the object along the time axis (thus changing the begin and end attributes); if on the other hand it looks as a double arrow, the action changes the length of the object (duration attribute). Durations expressed as relative time placements between elements and represented by red arrows can be modified directly by moving the mouse while it is on these arrows.

It is important to notice that every editing action implies a consistent update on all the elements of the document. For instance changing the duration of an object can induce changes of time positions of other objects of the same composite or of ascendent composites. This consistency checking is continuously performed during user actions thanks to the use of the Cassovary constraint solver. This feature allows users to perceive on the whole document and in real time all the consequences of the modification they are performing.

## 8.3 Behavior and Functional Editor: The problems

Since GRiNS is a commercial software and LimSee2 is a Java application AXMEDIS needs to build its own SMIL Behavioural and Functional editor, as an open C++ application, with the same class of functions that LimSee2 or GRiNS have. On the other hand the tools mentioned above can be used as reference implementations to test the compliance of the SMIL content authored with the AXMEDIS behavioural and functional editor.

The Behavioural editor allows the organization and synchronization of different media object along a timeline (see Figure 16). According to several feedbacks received from experts in SMIL authoring, it is very important to have the possibility of finely "tune" the media objects once deployed along the timeline. In other words if it is important to have a graphical interface allowing the "visual" representation of the media object in time – like the one shown in Figure 16 - , it is at the same time essential to have the possibility to specify the value of the different time stamps (start of decoding, pauses, end) by directly inserting numerical values. When designing and implementing this component it is then necessary to carefully consider the appropriate harmonization of these two complementary features.

Starting from the existent visual editor prototype described in § 8.4 a Behavioural Editor will be integrated to control the SMIL temporal structure based on a timeline scale. The following snapshot (realized as mock-up) shows the graphical interface of the tool used for editing different media objects based on this timeline. In the depicted situation, two are the video objects, one the audio and one the texture.

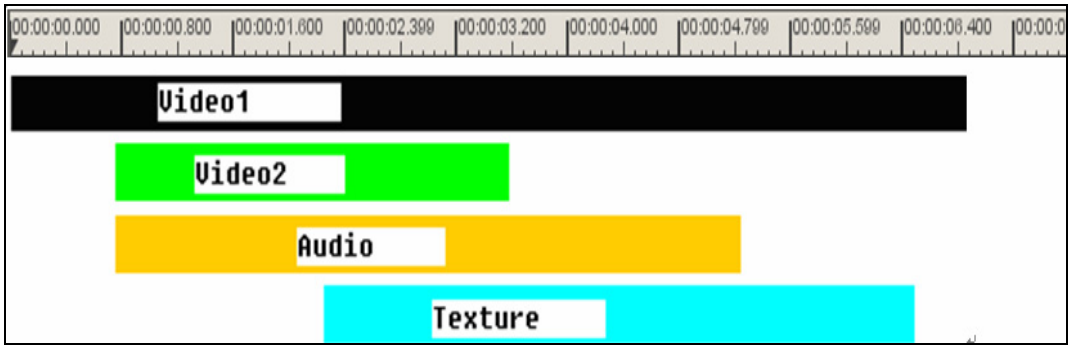



**Figure 16 - Example of layout of a behavioral editor and viewer where several media objects are placed along the timeline**

Work is in progress to implement both the visual timeline for objects deployment and the interface enabling the insertion of numerical values for time stamps.

# 9 Descriptions and Comments Editor and Viewer (EPFL)

Annotations and Comments View will permit to display information including annotations and comments within AXMEDIS objects or parts of them. Annotations and Comments will be logically included as elements in AXMEDIS objects without actually modifying their contents. Such View will be user customizable, i.e. users will be able to select which types of media have to be made accessible for each component.
The main functionality implemented by this component can be summarized as follows:

- annotations and comments can be created, edited and deleted by means of graphical actions such as contextual menus generated by right-clicks, drag-and-drop, etc
- annotations and comments can be accessible or not. The interface will enable the user to select for each kind of media, the set of annotations and comments to be made accessible.
- plug-ins handling specific media types (text, audio, still pictures) reasonably expected are supported since the different annotations and comments may have several natures.
- the metadata associated to media can also be printed and visualized in a human understandable format.

## 9.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Not available yet |
| List of libraries used | wxWidget 2.4.2 |
| References to other major components needed | AXOM |
| Problems not solved | Integration and interaction with Visual/Behavioural editor |
| Configuration and execution context | |
| Programming language | C++ |

## 9.2 Descriptions and Comments: The problems

The most critical point from an architectural point of view is the harmonization of this component with the visual editor and viewer. In fact, the insertion of description and comments should be a part of the process of SMIL objects creation to be smoothly integrated with the graphical set of functionality. It is then of the utmost importance that the user can invoke this functionality straightforwardly when editing SMIL objects and that the concerned annotations are appropriately integrated in the produced SMIL content.
Two are then the issues to be faced:

- Interaction with the visual viewer and editor;
- Support of a consistent integration of the textual - and possibly audiovisual - comments and annotations in the produced SMIL file.

The descriptions and comments editor and viewer will be integrated with the visual and behavioural editor and viewer. The insertion of comments and annotation should be done directly from the editors' interface by means of a procedure whereby the scene in course of authoring can be "freezed" (no other editing action can be performed on it) and comments can be added. This could be implemented by right-clicking on the chosen shape and by selecting the desired action by means of a contextual menu. Figure 17 shows how this functionality will be integrated into the Visual Editor.

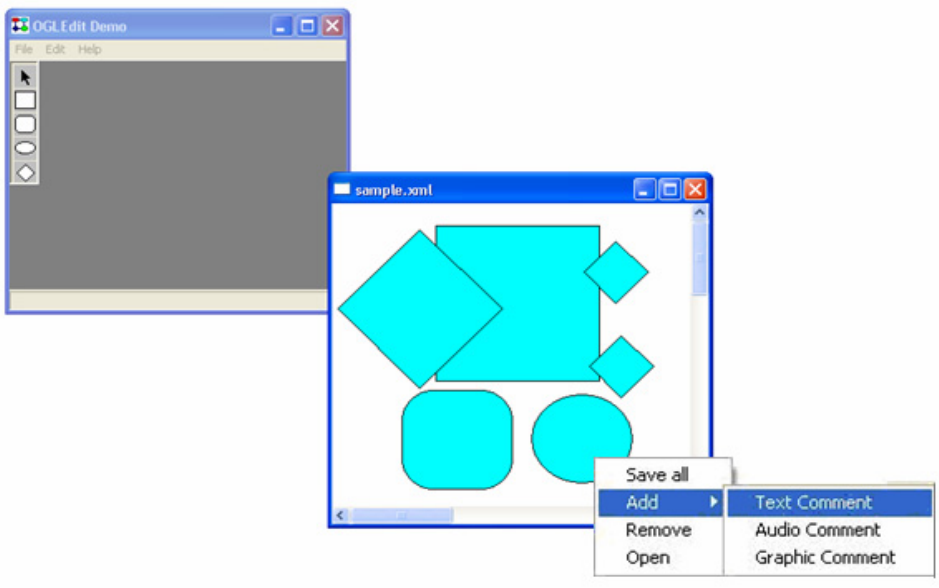



**Figure 17 - Contextual menu allowing the insertion of comments**

# 10 Metadata Editor and Viewer (UNIVLEEDS)

The AXMEDIS Metadata Editor and Viewer aim to adequately display and edit metadata information contained within an AXMEDIS object. Due to the complex nature of AXMEDIS objects, there may be one or more metadata sections with different schemas, including MPEG-21, Dublin Core, and AxInfo.

- Metadata Viewer shall be able to adapt itself (e.g. by analysing the data-related XML schema) automatically to the metadata structure;
- Specific set of valuable metadata, such as authoring MPEG-7 metadata, should be included into AXMEDIS Editor basic release;

For Metadata visualisation, two possible approaches can be done: (1) using the above editor without the manipulation and save functionality activated; and (2) generating a HTML file (with CSS) and use a browser to display the file

The Metadata Editor main aims to add, delete, move and edit elements for Dublin Core and AxInfo metadata with validation. This is achieved by utilising the methods and functionalities provided by Xerces and the AXMEDIS MPEG-21 compliant model. The Metadata Editor and View will be integrated into the AXMEDIS Editor as described in section 6 of the report.

## 10.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | |
| References to other major components needed | xerces-c_2_6D.dll |
| Problems not solved | • Adding and deleting elements with validation with schemas<br>• Adding and deleting attributes with validation with schemas |
| Configuration and execution context | |
| Programming language | C++ |

## 10.2 Metadata Editor and Viewer: State of the art

Currently there is an influx of high performance, fully compliant XML parsers including Xerces and Expat, as it is further discussed in the deliverable DE4-3-1 [12]. While there are editors and viewers available for creating, specifying and editing XML, tools are required for editing various metadata standards (e.g. Dublin Core). This is an ongoing problem currently being addressed by working groups and specifically for the Dublin Core standard, an ongoing series of workshops to address these problems are ongoing by the Dublin Core Metadata Initiative (DCMI) Workshop Series [13].

The AXMEDIS metadata editor is required to apply manipulation functionalities such as adding, deleting and moving XML elements and their attributes with schema validation. This manipulation applies not only for the Dublin Core, but AXMEDIS AxInfo and other user defined metadata for the AXMEDIS objects. This requires the development of generic functionality that can be applied to multiple metadata standards including the user defined metadata. The metadata editor and viewer will utilise the new tools developed for the transcoding of metadata as discussed in DE4-3-1 and will also utilise the functionalities provided by the Xerces [14] libraries. Future development will utilise the new AXMEDIS MPEG-21 compliant model as discussed in section 3, AXMEDIS Object Manager.

## 10.3 Metadata Editor and Viewer: The problems

Some of the desired functionality has not been implemented:

- Addition and deletion of elements with validation to keep the metadata as valid XML using the schemas.
- Addition and deletion of element attributes with validation to keep the metadata as valid XML using the schemas.
- User defined filtering of the tree structure

## 10.4 Metadata Editor and Viewer: Work performed

The work on the Metadata Editor and Viewer has been started and the first prototype with metadata viewing and basic editing functionalities is near completion for demonstration at the AXMEDIS2005 Conference. The main stages of the Metadata Viewer and Editor are as follows:

- Visualize the XML as a tree structure
- Loading and saving of the metadata XML file using the library Xerces-C++
- Parsing XML is SAX2 event based
- Separate test GUI with metadata editor/viewer wxWindows Frame implemented with wxWidgets (version 2.4.2)
- Editing wxWindows Dialog for editing loaded elements and attribute values
- GUI functionalities including local and global expansion of the tree view and opening the editing dialog box.

Planned future enhancements include:

- Adding and deleting metadata elements with validation using XML schemas
- Adding and deleting metadata attributes with validation using XML schemas
- Filtering viewing elements by namespace or schema identification
- Integrate Metadata Editor and Viewer frame into the AXMEDIS Editor and Viewer
- Creating a HTML view of the metadata elements for viewing object metadata in a browser

In the following image the metadata view is shown together with the advanced editing dialog for editing the element values and the attribute details.

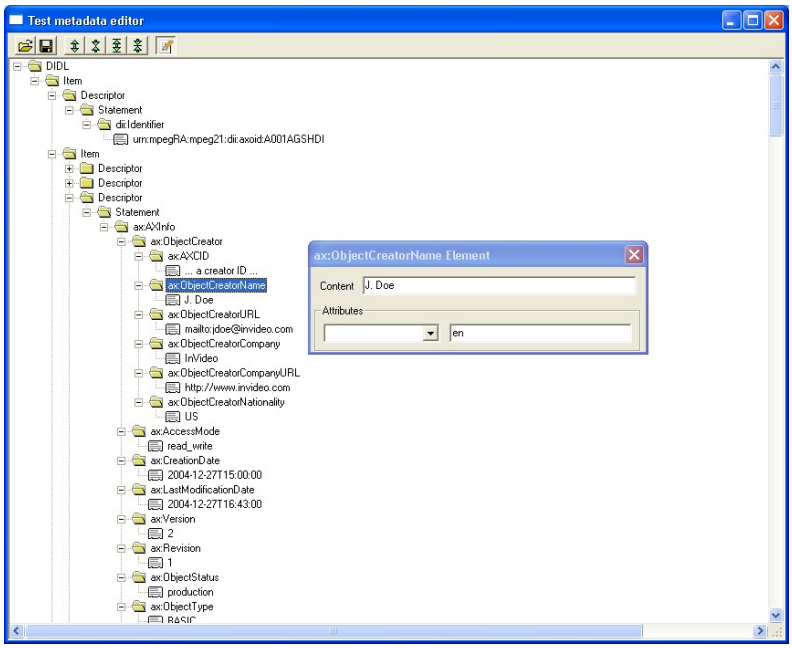


**Figure 18 - Metadata view with editing tools for values and attributes.**

# 11 AXMEDIS ActiveX Control (DSI)

The AXMEDIS ActiveX control allows exploiting AXMEDIS objects in windows applications and in web pages.

## 11.1 Technical Details

| | |
|---|---|
| Reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | MFC, wxWidgets |
| References to other major components needed | AXOM, Hierarchy Editor & Viewer, Internal Image Viewer, Internal Audio Player, Internal Video Player, Internal Document Viewer |
| Problems not solved | |
| Configuration and execution context | |
| Programming language | C++ |

## 11.2 ActiveX Control: Work performed

The initial prototype of AXMEDIS ActiveX Control allows to:

- load an AXMEDIS object
- display the AXMEDIS hierarchy (for end user) and the resources in the object
- hide/show the hierarchy
- get the content count (how many resources/objects are present in the object)
- open a content in the resource view
- get some minimal information on the content
- control the execution of the content play/stop

The following picture shows the ActiveX used in the ActiveX Control Test Container

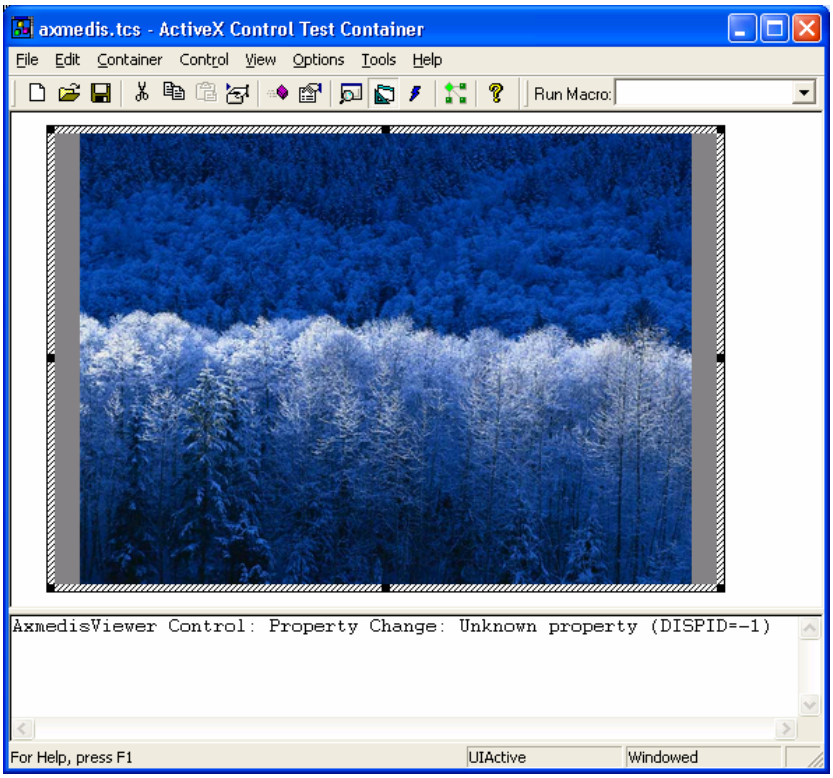
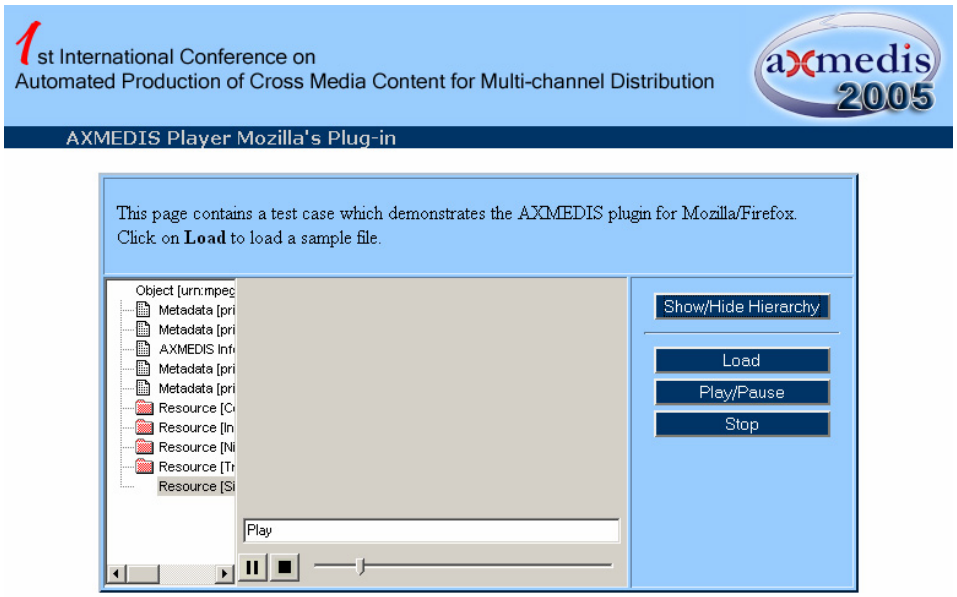



**Figure 19 - ActiveX Control Test Container**

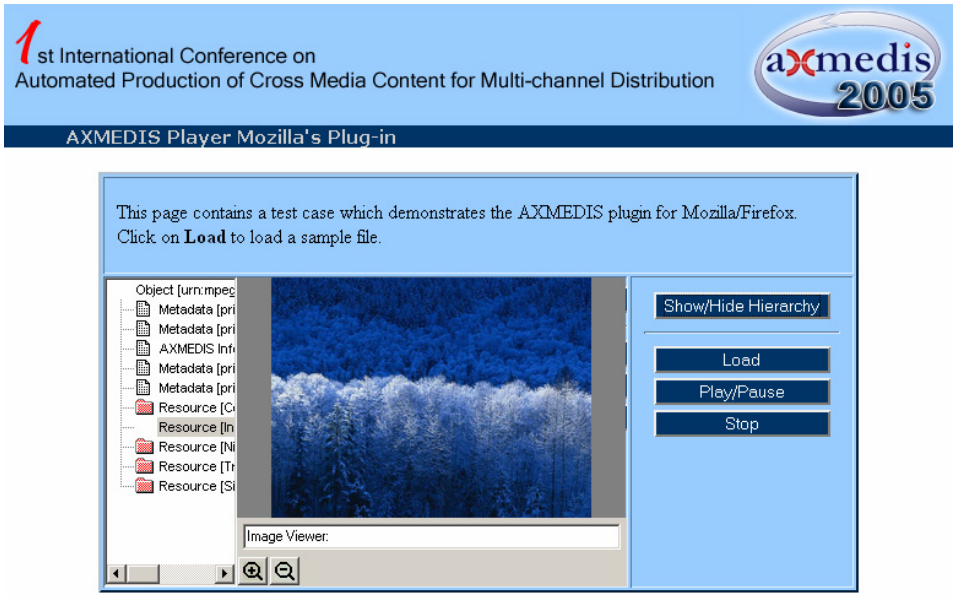
See the AXMEDIS PC Player prototype as another example of AXMEDIS ActiveX use.

# 12 AXMEDIS Plug-in into Mozilla (SEJER)

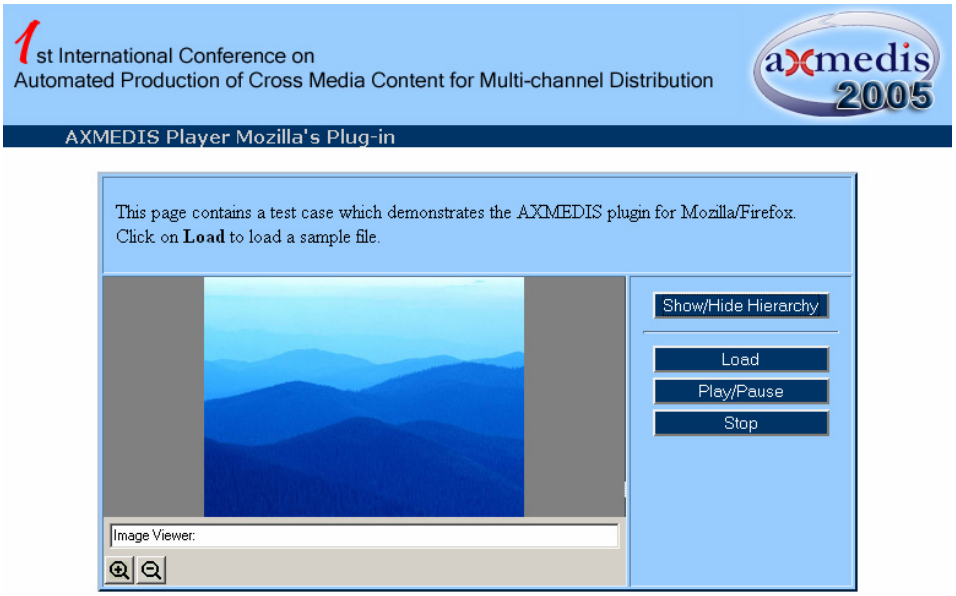
The AXMEDIS plug-in into Mozilla [15] is intended to integrate the AXMEDIS player inside HTML pages. The three screenshots below demonstrate the rendering of several contents, with the hierarchy visible or not.




Playing music




Displaying a picture




Displaying a picture with the Hierarchy panel hidden

**Figure 20 - Rendering of several contents into Mozilla**

## 12.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | Gecko plugin SDK |
| References to other major components needed | xerces-c_2_6D.dll |
| Problems not solved | Raising events that can be listened by JavaScript in the HTML page |
| Configuration and execution context | 1) Copy the following files in the "plugins" directory of firefox or mozilla:<br>- xerces-c_2_6D.dll<br>- nsIAxMozillaPlugin.xpt (from the bin\win32 directory)<br>- npaxmozilla.dll (from the bin\win32 directory)<br>2) Open Mozilla/firefox<br>3) Open the test page: test.html |
| Programming language | C++ |

## 12.2 Plug-in into Mozilla: The problems

The code of the plug-in is based on the sample “scriptable” from GeckoPluginSDK. The first problem was to have mozilla/firefox to recognize the plug-in, even after the library has been copied into the plugin directorory of the application. In order to do that, the library name **must** start with “np”

Then, when the code from the AxActiveX was inserted into the plug-in, three problems arise:

- the resources of the player were not displayed correctly. The HINSTANCE of the application has to be set to the one of the library;

- the initial layout of the controls composing the player is not correct until the user resizes the plug-in. This was corrected by forcing the plug-in to layout at the end of the initialization;
- the background of the player was not drawn correctly. Ignoring the `WM_ERASEBKGND` solved the problem.

An issue remains: the plug-in should be able to raise events that JavaScript code in the HTML page should listen to. Little documentation is available for this specific task so a solution is still under investigation.

### 12.3 Plug-in into Mozilla: Work performed

The work on the AXMEDIS plug-in into Mozilla has been started and is almost in sync with the ActiveX Control.
The following properties and methods have been implemented:

```
attribute boolean viewHierarchy;
readonly attribute short contentCount;
short load(in string filename);
string getContentType(in short contentRef);
void openContent(in short contentRef);
void play();
void stop();
```

The plug-in is also able to read the value of the viewHierarchy and the path to the file to load from the parameters in the HTML page.

## 13 AXMEDIS plug-in into Multimedia Players (DSI)

In this area several experiments have been performed on Windows Media Player trying to customize the default behaviour of the player in order to:

- enforce AXMEDIS digital rights;
- render multimedia content packaged in an AXMEDIS object.

For more details on Customizing Windows Media Player see Report in Appendix (file Customizing-WMPlayer.pdf included with this deliverable).

### 13.1 Technical Details

| | |
|---|---|
| reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | Windows Media SDK |
| References to other major components needed | AXOM |
| Problems not solved | • Impossible the customization of Windows Media DRM<br>• Impossible customization of WM player behaviour in order to enforce digital rights<br>• Poor set of functionality offered to play well-know resources in a render plug-in |
| Configuration and execution context | Windows Media Player Preset with Plug-in Component |
| Programming language | C++ |

## 13.2 Plug-in into Multimedia Players: The problems

Two main issues have been identified in examining the Windows Media Technology:

- It is not possible to control play of resource in terms of avoiding unauthorized operations. No customization are possible in terms of DRM.
- It is possible to create a render plug-in and a UI plug-in, but is not possible to demand play of well-know resources as soon as they have been extracted by the composed AXMEDIS object. Only a device canvas is at disposal to be drawn by the render plug-in.

## 13.3 Plug-in into Multimedia Players: Work performed

Several experimentations have been performed with the Windows Media Technologies: a prototype of customized Windows Media Player has been realized. The customization has been focused on the digital rights enforcement and on AXMEDIS object decoding and rendering on the WM Player.

# 14 AXMEDIS PC Player (DSI, EPFL)

The AXMEDIS PC Player application allows opening and rendering AXMEDIS objects in the end user terminal. The AXMEDIS player will use the AXMEDIS internal Players to render the different types of content: audio, video, text, and SMIL presentations.
In these terms, given the nature of an AXMEDIS Object, the Player shows a strong relationship and interconnection with the concept of MPEG-21 Client Terminal.
A Visual Basic application has been realized to test the ActiveX, it could be considered as a preliminary prototype of the PC Player.
Furthermore, the Internal SMIL Player (see DE5.1.1, AXMEDIS Framework Infrastructure) can at the same time be exploited to synchronize the several media objects according to precise spatial and temporal stamps, and as such constitutes the core of the scheduling and playback functionality of the AXMEDIS Player.

## 14.1 PC Player: State of the art

Other multimedia tools, mainly based on proprietary formats, are currently spreading on the market. They are based on frameworks that are not as flexible and powerful as MPEG-21, however they constitute a state of the art in the domain, since as noticed above MPEG-21 is still in development stage and current tools are poor. Reviewing these frameworks is important as it provides a direct investigation about what currently means a state of the art player terminal in terms of media support, security, flexibility and other aspects; this is useful to learn about successful features of client terminal players for PC platforms and also to show how much advanced the MPEG-21 based AXMEDIS client will result.
We review here the main multimedia frameworks in their actual state. We do not speculate about their future, or about their possible evolutions. The landscape is likely to evolve quickly, some of the main actors will probably disappear, and new actors will also appear.

### 14.1.1 Flash

Flash is a proprietary system developed by Macromedia, Inc. [22]. It is composed of an authoring tool, and a viewer available for free on the most widely available platforms (Macintosh and Windows based). The viewer can be integrated in an HTML page, so Flash content can be easily integrated in Web content.

Flash is based on vector graphics. In conjunction with scripting (Actionscript), and animation (timeline based as well as scripting based), this has made of Flash the most widely used multimedia framework for the Web. The very low bit rate induced by the use of vector graphics, scripting, and timeline based animation has made of Flash a very convenient format for Web-based animation, storyboards, high quality graphics, and a suitable alternative to HTML for the development of Internet web sites.

**Media support**

The support of audio can be considered as being poor in Flash. The only compression scheme supported is mp3, together with a proprietary compression scheme known as “Nelly Mosser” for which no information is

available to our knowledge. Flash supports also uncompressed schemes, PCM based, such as WAV or AIFF. For structured audio, no format – even MIDI - is supported. Video support in Flash has continued to evolve since its introduction in Flash MX and Flash Player 6. Flash Player 7 greatly improves video quality, supports higher frame rates, and provides additional opportunities for loading dynamic media at runtime.

**System support**

Flash supports the operating systems of Windows and MacOS. It cannot support the Unix, Linux, BSD system.

**Authoring and production**

Flash benefits from a proprietary authoring tool developed by Macromedia.

**Interactivity and animation**

Interactivity and animation can be implemented in Flash by using ActionScript, a proprietary scripting language

**Openness and extensibility**

Flash is in principle a closed system on the client's side. No extensions can be developed, no decoders can be added, and no interactivity other than interactivity defined on the authoring side with the ActionScript scripting language can be defined to enhance the standard viewer (this is not to be confused with the extensibility functions available in the new Flash MX 2004, which are available in the Flash authoring application).

Flash is open to XML, and able to exploit XML data in a client-server architecture, via http-based protocol, or via XML socket based, real-time exploitation of data. With this functionality, it is for example possible to imagine a Flash client application exploiting XML data available on line, for example XML-based metadata such as RDF, or Dublin Core, or even MPEG-7 metadata in their XML format.

The Flash file format is itself now open, as well as some parts of the source code, and many developers are developing new Flash based solutions. For example, the NorthCode company [23] has developed SWFStudio (http://www.northcode.com/swfstudio/), a software which makes possible to build stand-alone executables from Flash content. In this configuration, it becomes possible to build plugins to Flash executables. The same society has developed a plugin development kit in order for other developers to build their own extensions to Flash (with the restriction that this works only with Flash stand alone applications – it is always impossible with the Flash standard client).

**Security and privacy**

Flash implements a browser-like security sandbox scheme in order to ensure the security and privacy of Flash movie and the client machine. The sandbox defines a limited space in which a Flash movie running within the Macromedia Flash Player is allowed to operate. Its primary purpose is to ensure the integrity and security of the client's machine, and as well as security of any Macromedia Flash movies running in the player. Basically, the sandbox idea is the following: A Macromedia Flash movie executes inside a sandbox. Any information inside the sandbox can be communicated only to the domain from which the movie came. Access to information within the sandbox from outside of the sandbox is severely limited.

### 14.1.2 Windows Media

Being preinstalled with every version of Microsoft Windows sold, Windows Media Player is becoming increasingly widespread on the web.

**Media support**

Windows Media Player supports most of media formats, but it does not support the RealNetworks content such as the .ra, .rm, .ram media file. It cannot support the QuickTime content like .mov, .qt format. As for the MPEG-4 (.mp4), Windows Media Player gives no support.

**System support**

Windows Media Player supports the operating systems of Windows and MacOS. It cannot support the UNIX, Linux, BSD system.

**Authoring and production**

Production of Windows Media content can be done in multiple ways: by the mean of Windows Media Encoder, or by the mean of the toolkits provided by Microsoft for this purpose. This toolkit can be accessed by the mean of the C++ language, the Visual Basic language, or even by the mean of an HTML interface.

**Interactivity and animation**

No support of interactivity – scripting, controls – is directly available in Windows Media.

**Openness and extensibility**

Customization of the Windows Media Player is possible by using the Software development Kit provided to this end by Microsoft. By using the SDK, it is possible to develop a customized end-user interface driving the Windows Media content, in any language supported by the Windows Media SDK (C++, Visual Basic, HTML, .net with C#…).

**Security and privacy**

Windows media frameworks implement the Windows Media digital rights management (DRM) platform to protect and securely deliver a la carte and subscription content for playback on a computer, portable device, or network device. Windows Media DRM is comprised of multiple components including Windows Media DRM for Portable Devices and Windows Media DRM for Network Devices, as well as an updated Windows Media Rights Manager SDK. These components allow for the seamless flow of content to almost any device, offer the widest range of purchase and rental options for digital media, and ensure the security of premium content as it flows from device to device. Windows Media DRM works in the following five steps: 1.Packaging 2.Distribution 3.Establishing a license server 4.License acquisition 5.Playing the digital media file to secure content providers to protect their content and maintain control over the entire process of the media distribution.

### 14.1.3 QuickTime

**Apple's products**

They are the following:

- QuickTime 7 Pro [24] enables H.264 video creation, audio and video capture, multi-channel audio creation and multiple files export. It is an easy-to-use tool for creating AAC audio files and 3G files for mobile viewing, editing videos and exporting movies.
- QuickTime MPEG-2 Playback Component provides QuickTime users with the ability to import and play back MPEG-2 content, including both multiplexed and non-multiplexed streams. It is suited for content creators with projects such as Professional content production and transcoding video content (from MPEG-2 video to MPEG-4 for example).

- QuickTime Broadcaster is a tool for producing live broadcast events.

**Media support**

QuickTime 7 Player supports a wide-range of industry-standard audio formats, including AIFF, WAV, MOV, mp3, MP4 (AAC only), CAF and AAC/ADTS. For structured audio, QuickTime supports MIDI. There is no support for audio effects or 3D audio. Multichannel audio is supported by QuickTime 7 up to 24 audio channels, enabling standard surround formats. QuickTime 7 supports H264.

**System support**

QuickTime supports the operating systems of Windows and MacOS. It cannot support the UNIX, Linux, BSD system.

**Authoring and production**

Adobe's Premiere or Macromedia Director can generate QuickTime content. There are also a number of production tools available, like FinalCut Pro for instance. Apple does not provide authoring tools, but software like Adobe's Premiere or Macromedia Director are able to produce QuickTime content, generally by the mean of a plug-in. Apple's QuickTime-related products (QuickTime Pro, QuickTime MPEG-2 Playback and QuickTime Broadcaster) are not literally authoring tools, but provide however a few creating, encoding and editing functionalities.

**Interactivity, animation**

No scripting language is available for defining interactivity, but interactivity can be defined by using the QuickTime Software Development Kit provided by Apple.

**Openness, extensibility**

Extensions to QuickTime can be defined on the user's side by using the QuickTime Software Development Kit provided by Apple. It provides interfaces in C or Java QuickTime content can be embedded in a web page, but only a restricted set of functions are available from scripting languages such as JavaScript, making QuickTime not very well suitable for development of interactive content on the Web. Timeline-based, raw graphics animation is provided by authoring tools such as Adobe Premiere or Macromedia Director.

**Security and privacy: tbd**

### 14.1.4 Real

RealNetworks media are limited to audio, from speech, mono-channel to surround, channel music, and video. There is no native support for interactivity, vector graphics, but the Real Player [25] supports the W3C's standard for synchronized multimedia SMIL, and thus interactivity, animation and support of vector graphics can be integrated this way.

**Media support**

Real plays every major media format Including, AVI, MP3, RealAudio, RealVideo, WAV Audio, and Windows Media. This feature makes it a very popular media player.

**System support**

Real not only supports the operating systems of Windows and MacOS, but also support the UNIX, Linux. But it cannot support BSD system.

**Authoring and production**

The Helix producer enables encoding of streaming media (audio, video), in the native Real formats, with different bit rates. The Helix producer cannot generate SMIL animations – but can generate the included media.

**Interactivity, animation**

Real Media does not include any native support for interactivity or animation, but these functionalities can be integrated in SMIL animations by integration of Flash or SVG content.

**Metadata**

There is no support for metadata in Real Media.

**Openness and extensibility**

Real Media does not include any native support for scripting or extensibility, but the SMIL support for these features must be taken in account.

**RealPlayer**

- Audio: The Real Player includes support for its audio proprietary format as well as for mp3 format. It is also possible to include MIDI content.
- Vector graphics, animation: Real Player supports the integration of Flash content (only Flash 3 and Flash 4), with some restrictions such as for audio content, which must be integrated using another channel (mp3, or rm). Interaction with Flash content is also supported, enabling in this manner capabilities of interactions with timeline from the user. It is for example possible to develop a simple user interface in Flash, composed of some buttons for playing, stopping, fast reviewing or forwarding an audio track, but this kind of interaction will be limited to interaction with the timeline.
- Real Player supports also integration of SVG.

**Security**

Real implements the Helix DRM platform for secure media content delivery over PC and non-PC devices, including mobile devices and home equipments. Helix DRM includes a set of products and services enabling business models through secure rights managed distribution of movies, music and other digital content. Helix DRM provides secure media packaging, license generation and content delivery to a trusted media player base across all major platforms to multiple devices. It extends the RealPlayer and Helix Platform open architecture to accommodate the incorporation of a wide range of rights management systems. It integrates into existing infrastructures and back-end systems, supporting a broad set of business models including purchase, rental, video on-demand, and subscription services.

The above media players are all developed for the multiple kinds of content files which cover over the range of document, images, audio, video, multimedia, etc. Here, we also want to enclose two famous audio players iTunes and Winamp with the purpose of learning their good features of Graphical User Interface which could be a good example for the development of Axmedis Player.

### 14.1.5 iTunes

iTunes [26] is also a digital media player developed by Apple Computer, for playing and organizing digital music and video files. Additionally, the program connects to the iTunes Music Store (sometimes referred to simply as "iTunes" or "iTMS") which allows users to purchase digital music files that can be played by iTunes. The player has gained and maintained a reputation as being easy to use while still providing many features for obtaining, organizing, and playing music. iTunes is also the principal way to manage the music on Apple's popular iPod digital audio player. The program is freely downloadable and is also supplied with Mac OS X as well as Apple's iLife home-application suite.

**Media Support**

iTunes can currently encode to MP3, AIFF, WAV, MPEG-4 AAC, and Apple Lossless, and can play anything QuickTime can play (even video formats, as long as they have audio). In order to play other formats such as the Ogg Vorbis audio format iTunes requires addition of QuickTime components. However, the extensions for Ogg Vorbis does not work with QuickTime version 7 and Mac OS Tiger installed. In May 2005, video support was introduced to iTunes with the release of iTunes 4.8. But Video support in iTunes is still limited at this point iTunes is so far still incompatible with the most common video formats such as .MPEG and WMV.

**System support**

iTunes supports Windows and MacOS. It cannot support the UNIX, Linux and BSD system.

**Authoring and production**

iTunes was developed from SoundJam MP, a popular commercial MP3 application distributed by the Macintosh software company Casady & Greene.

**Openness and extensibility**

On the Macintosh, iTunes is tightly integrated with Apple's iWork suite of applications and the rest of the applications in iLife.

**Security and privacy: tbd**

### 14.1.6 Winamp

Winamp [27] is an audio player made by Nullsoft, part of Time Warner. It is skinnable, multi-format freeware. Originally, MP3 playback was based on the AMP decoding engine by Tomislav Uzelac et al. In later versions this was replaced with Nitrane, a proprietary decoder created by Nullsoft and subject of a lawsuit from Playmedia Systems, Ltd. After an out of court settlement and licensing agreement, Nullsoft switched to an ISO decoder from Fraunhofer Gesellschaft, the developers of the MP3 format

**Media Support**

Winamp support a wide range of audio format file including MIDI, MOD, MP3, Ogg Vorbis, WAV, WMA and many other audio formats.

**System support**

Winamp only supports Windows. It cannot support MacOS, UNIX, Linux, and BSD system.

**Authoring and production**

Winamp was first released by Justin Frankel in 1997.

**Security and privacy: tbd**

The features of these above media players are summarized in the following tables.

**General Features**: Basic general information about the players: creator/company, license/price etc.

| | Creator | First public release date | Stable version | Software license | Proprietary format |
|---|---|---|---|---|---|
| **Flash** | Macromedia | December 1996 | 8.0 | Proprietary | Flash |
| **Director** | Macromedia | 1988 | MX 2004 | Proprietary | DCR |
| **iTunes** | Apple Computer | January, 2001 | 4.9 | Proprietary | Apple Lossless |
| **QuickTime** | Apple Computer | December, 1991 | 7.0.1 | Proprietary | QuickTime |
| **RealPlayer** | RealNetworks | 1995 | 10 | Proprietary | RealAudio, RealVideo |
| **Winamp** | Nullsoft | June, 1997 | 5.094 | Proprietary | NSV |
| **Windows Media Player** | Microsoft | November, 1992 | 10 | Proprietary | WMA, WMV |

**Operating Systems Support**: The operating systems on which the players can run natively (without emulation).

| | Windows | Mac OS X | Linux | BSD | Unix |
|---|---|---|---|---|---|
| **Flash** | Yes | Yes | No | No | No |
| **Director** | Yes | Yes | No | No | No |
| **iTunes** | 2000/XP/2003 only | Yes | No | No | No |
| **QuickTime** | Yes | Yes | No | No | No |
| **RealPlayer** | Yes | Yes | Yes | No | Yes |
| **Winamp** | Yes | No | No | No | No |
| **Windows Media Player** | Yes | Yes | Planned | No | No |

**Main Features**:

| | Audio playback | Video playback | Outbound streaming | Skinnable | Media Database |
|---|---|---|---|---|---|
| **Flash** | Yes | Yes | Yes | No | Yes |
| **Director** | Yes | Yes | Yes | No | Yes |
| **iTunes** | Yes | Yes | Yes | No | Yes |
| **QuickTime** | Yes | Yes | No | Partial | No |
| **RealPlayer** | Yes | Yes | No | Yes | Yes |
| **Winamp** | Yes | Yes | No | Yes | Yes |
| **Windows Media Player** | Yes | Yes | No | Yes | Yes |

**Audio Formats Support**: Information about what audio file formats the players support.

| | Lossy compression | | | | | | Lossless compression | | |
|---|---|---|---|---|---|---|---|---|---|
| | **MP3** | **WMA** | **RealAudio** | **Vorbis** | **AAC** | **AC3** | **APE** | **FLAC** | **ALAC** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Flash** | Yes | Yes | No | No | No | No | No | No | No |
| **Director** | Yes | Yes | Yes | Yes | Yes | ? | ? | ? | ? |
| **iTunes** | Yes | Partial | No | No | Yes | No | No | No | Yes |
| **QuickTime** | Yes | No | No | No | Yes | No | ? | No | Yes |
| **RealPlayer** | Yes | Yes | Yes | Yes | Yes | ? | ? | ? | ? |
| **Winamp** | Yes | Yes | No | Yes | Yes | No | No | No | No |
| **Windows Media Player** | Yes | Yes | No | No | No | No | No | No | No |

**Video Formats Support**: Information about what video file formats the players support.

| | MPEG-1 | MPEG-2 | MPEG-4 | WMV | RealVideo | Theora | Flash |
|---|---|---|---|---|---|---|---|
| **Flash** | Yes | Yes | No | Yes | No | No | Yes |
| **Director** | Yes | ? | ? | No | Yes | ? | Yes |
| **iTunes** | Yes | No | Yes | No | No | No | Yes |
| **QuickTime** | Yes | No | Yes | No | No | No | Yes |
| **RealPlayer** | Yes | ? | ? | No | Yes | ? | Yes |
| **Winamp** | Yes | No | No | Yes | No | No | No |
| **Windows Media Player** | Yes | Yes | No | Yes | No | No | Yes |

**Container Formats Support**: Information about what container formats the players support.

| | AVI | ASF | QuickTime | OGM | Matroska | MP4 |
|---|---|---|---|---|---|---|
| **Flash** | Yes | Yes | No | No | No | No |
| **Director** | Yes | ? | Yes | ? | ? | Yes |
| **iTunes** | Yes | ? | Yes | ? | ? | Yes |
| **QuickTime** | Yes | No | Yes | No | ? | Yes |
| **RealPlayer** | Yes | No | No | ? | ? | ? |
| **Winamp** | Yes | No | No | No | No | Yes |
| **Windows Media Player** | Yes | Yes | No | No | No | No |

## 14.2 PC Player: Work performed

Two prototype players have been developed. The first one exploited the features of Microsoft Foundation Classes and ActiveX in order to rapidly develop a user friendly graphical interface able to open and display objects and the associated AXMEDIS hierarchy.
The second player is based on an open source SMIL player called AMBULANT [18] which has been "embedded" into a WxWidgets application implementing the user interface. The implementation of this player permitted to appreciate the flexibility of the SMIL language for the creation of multimedia scenes composed by several synchronized media objects.
The following sections provide some details and screenshots of the developed applications.

### 14.2.1 ActiveX based PC player

**Technical Details**

| | |
|---|---|
| Reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | MFC |
| References to other major components needed | AXMEDIS ActiveX |
| Problems not solved | |
| Configuration and execution context | |
| Programming language | Visual Basic (.NET) |

The preliminary prototype of the AXMEDIS PC Player allows to:

- load an AXMEDIS object
- browse the content in the object (only resources)
- hide/show the AXMEDIS hierarchy
- play/stop the content fruition (for audio/video)

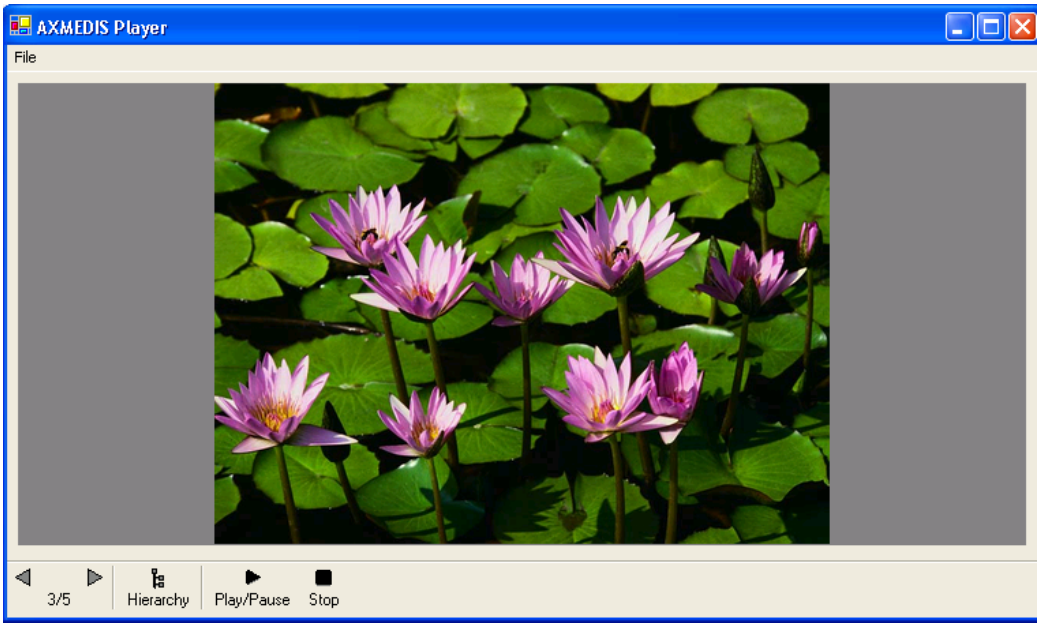



**Figure 21 – AXMEDIS PC player rendering an ActiveX image**

Note: in the figure above the ActiveX is only the image and not the buttons

When the Hierarchy button is pressed, the AXMEDIS Hierarchy is shown:

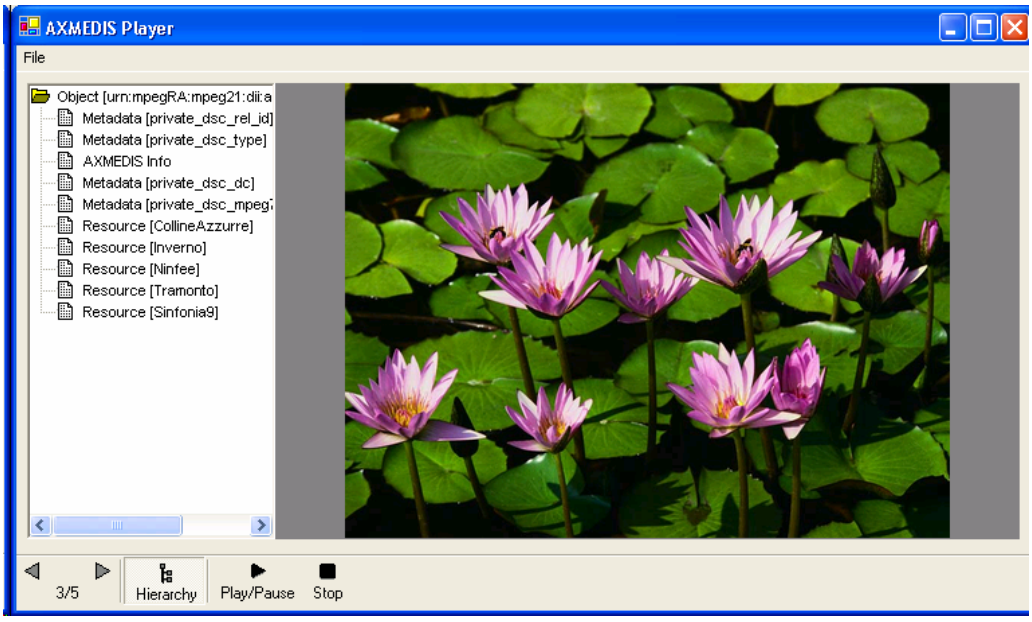


**Figure 22 – AXEDIS hierarchy shown with the resource**

### 14.2.2 SMIL Player

The purpose of the SMIL player is twofold. On one hand it will be used as internal viewer of the AXMEDIS editor when a creator of SMIL content will need to have a preview of the authored objects. On the other hand the very same core functionality of the SMIL player will be needed on the final user's PC terminal as interface for accessing and consuming content. This section details the features of the developed SMIL player implemented as a stand alone application still to be integrated in the Visual editor and viewer.
The version of the SMIL player to be distributed to end users will be produced by automatically extracting only the subset of features of the complete SMIL editor which are related to SMIL object decoding and rendering.

**Technical details**

| | |
|---|---|
| Reference to the AXFW location of the demonstrator | Private information |
| List of libraries used | wxWidgets |
| References to other major components needed | AXOM |
| Problems not solved | Complete wxWidgets porting, support of main AXMEDIS components necessary for integration, internal media types support |
| Configuration and execution context | |
| Programming language | C++ |

The developed prototype of the AMBULANT SMIL player running embedded in a wxWidgets application can be used by users to perform the standard operations of a player: play, stop, pause, close and open (refer to DE5.1.1 [28]).
The application has been built by modifying a wxWidgets sample called *minimal*. The source code for this application has to be placed the directory **$WXWIDGETS_INSTALL_DIR\wxWindows-2.4.2\samples\minimal_modified** where the Visual Studio 7 Solution file is **wxWindows-2.4.2\samples\minimal_modified\minimal.sln**
To use AMBULANT the application must be linked to **libambulant_win32.lib** because AMBULANT is implemented as a static library.
The main steps necessary to integrate the AMBULANT player in a wxWidgets application can be highlighted by comparing the two files **minimal\minimal.cpp** and **minimal_modified\minimal.cpp**. This comparison will show that the lines of code added to the original application are not many.
The most important modifications to *minimal.cpp* performed to integrate AMBULANT are the following:

```
dummy = new gui_player(s_player_callbacks, u);
```

The line above creates an instance of the AMBULANT player.
The **u** parameter contains the file name of the SMIL file that will be played.
The **s_player_callbacks** parameter is a class that contains a HANDLE to the window where the SMIL presentation will be displayed (a HANDLE is the identifier of a window for the Windows OS. A HANDLE = DWORD = int = 32 bits).

```
m_canvas = new MyCanvas( this );
```

The line above creates the wxWidgets window in which the SMIL presentation shall be displayed.

```
s_hwnd = (HWND) GetHandle();
```

The line above returns the HANDLE of the window in which the SMIL presentation has to be displayed. AMBULANT will need this HANDLE and, to retrieve it, it shall call the function `new_os_window()` of the class `s_player_callbacks` (derived from `gui_callbacks`).

Other modifications to *minimal.cpp* have been performed to add a toolbar with the standard player buttons - play, stop, pause - and to add some items in the menus such as **File→Load**.


**Figure 23 – Sample SMIL presentation decoded and rendered by the SMIL player**

# 15 Bibliography


[1] MPEG-21 Multimedia Framework Part 1: Vision, Technologies and Strategy. ISO/IEC 21000-1
[2] ENTHRONE Web Site: http://www.enthrone.org/
[3] AXMEDIS Deliverable DE 2.1.1 "User Requirements and use cases"
[4] ENIKOS Web Site: http://www.enikos.com/home.shtml
[5] WEDELMUSIC Web Site: http://www.wedelmusic.org/
[6] MPEG-21 Multimedia Framework Part 2 – Digital Item Declaration (DID). ISO/IEC 21000-2
[7] MPEG-21 Multimedia Framework Part 4 – Intellectual Property Management and Protection (IPMP). ISO/IEC 21000-4.
[8] MPEG-21 Multimedia Framework Part 3 – Digital Item Identification (DII). ISO/IEC 21000-3.
[9] SMIL overview on the W3C consortium Web Site: http://www.w3.org/AudioVideo/
[10] Oratrix Web Site: http://www.oratrix.com/
[11] LimSee2 Web Page on the INRIA Web Site : http://wam.inrialpes.fr/software/limsee2/
[12] AXMEDIS deliverable DE 4.3.1 " Content Composition and Formatting"
[13] Dublin Core Metadata Initiative Web Site : http://dublincore.org/
[14] Xerces Web Page on the Apache Web Site : http://xml.apache.org/xerces-c/
[15] Mozilla Web Site: http://www.mozilla.org/
[16] AXMEDIS Deliverable DE 4.6.1 "Content Distribution via Internet"
[17] MPEG-7 Part 1: Systems - ISO/IEC 15938-1
[18] The AMBULANT Project Web Site: http://www.cwi.nl/projects/Ambulant/distPlayer.html
[19] The BIM Open Source Web Site: http://bimopensourcesolutions.freewebspace.com/
[20] EXPWAY Web Site: http://www.expway.com/bim.php
[21] XMill Web Site: http://sourceforge.net/projects/xmill
[22] Macromedia Inc. Web Site: http://www.macromedia.com/
[23] Northcode Company Web Site: http://www.northcode.com/
[24] QuickTime Web Site: http://www.apple.com/quicktime/pro/
[25] Real Player Web Site: http://www.real.com/
[26] iTunes Web Site: http://www.apple.com/itunes/
[27] Winamp Web Site: http://www.winamp.com/
[28] AXMEDIS Deliverable DE 5.1.1 "AXMEDIS Framework Infrastructure"