Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE4.3.1.2
Content Composition and Formatting, 1st update

Version: 1.3
Date:15/09/2006
Responsible: DSI (Ivan Bruno) (revised and approved by coordinator)

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: DE4.3.1.2 Contractual Date of Delivery: M24 (end of August 2006) Actual Date of Delivery: …… Work-Package contributing to the Deliverable: WP4.3 Task contributing to the Deliverable: WP4.2 Nature of the Deliverable: Report and Prototype Author(s): DSI, IRC, XIM, HP
Abstract: The report deals with problems related to the Automatic Content Production and more in general with content processing, therefore the current status of the Content Processing prototype and tools are reported. The report is structured in sections dealing with different aspects of content production: Automatic Content Processing Area based on rules and a distributed system, using JavaScript language for rules and definition of AXMEDIS Data Types for JavaScript Engine, Adaptation Tools and Algorithms for content processing, formatting, etc… Finally the state of the art in the field of automatic formatting and adaptation was investigated to introduce the AXMEDIS formatting tools: a templates-based system with functionalities for automatic layout selection and optimization . **Keyword List:** Content production, Javascript, Adaptation tools, formatting, composition, transcoding

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

This report is focussed on describing the AXMEDIS Content Production Area (AXCP) prototype, tools, work done and analysis of the state of the art regarding formatting and media transcoding tools. This document is structured in sections dealing with:

- AXMEDIS Content Processing Area (under responsibility of DSI): this section describes the current status and the work done in the development of tools in AXMEDIS Content Processing Area. This area copes with the problem of automatic content production, adaptation and protection of AXMEDIS object and their publication on a P2P environment (AXEPTool). The actual prototype is based on rules that include a procedural description using the Javascript language (script) and schedule information that describe the rule firing inside the AXMEDIS Rule Engine. Such engine is based on a Javascript engine derived from SpiderMonkey by Mozilla. To cope with the amount of needed resources (computational, time, etc…) during the content processing activity, an early version of a distributed environment was defined and based on GRID computing. Therefore the Rule Engine was decomposed in two main component: the AXMEDIS Rule Scheduler and AXMEDIS Rule Executor. The AXMEDIS Content Processing Area results are further illustrated by means of UML diagrams, tables, GUI definitions and snapshots, description of a Grid communication protocol and the XML schema for rules.
- JS Classes for the JavaScript Engine: the composition and formatting process by means javascript required the extension of the object data type inside the javascript language. The set of classes and functions that currently extend the capabilities of the Spidermonkey Javascript engine are described.
- Content Composition and Formatting: this section performs a state of the art analysis to identify limits and problems of pre-existent tools for automatic adaptation and formatting; then, architecture and status of AXMEDIS tools are reported. The actual prototype is based on SMIL for document structure description and XSLT for layout definition; techniques for layout optimization are sketched. UML diagrams and screenshots illustrate architecture and functionalities of these tools.
- Workflow for the integration with the AXMEDIS content processing and factory tools.
- Adaptation and Transcoding Tools: in this part the work done and analysis on external tools and libraries for content processing, content format transcoding, content formatting and adaptation (digital audio, video, animation, multimedia, metadata, etc…) are discussed. Part of them was used to implement and customise the set of functions and algorithms to call inside the automatic content processing area.
- Two related works (in italian) are contained in Annex1 and Annex2 (DE4.3.1.2Annex1 and DE4.3.1.2Annex2). They are technical reports produced at the University of Florence:
    - in the Annex1 is described an architecture for the automatic formatting of a SMIL presentation, selecting templates and style-sheets suitable for the resources in input;
    - in the Annex2 is described a prototype that performs the layout optimization of a SMIL presentation. The prototype uses Genetic Algorithms to optimize positions and dimensions of the regions that compose the presentation.

2 Introduction

The problems related to automatic content production should be decomposed into composition and formatting. In addition, these aspects have to be integrated with those related to content production workflow. Content workflow monitoring and management is considered in WP4.4, which deals with the P2P model behind AXEPTool.

2.1.1 T4.3.1-2 Composition and Formatting algorithms and tools (DSI)

Major partners involved

DSI with ILABS, UNIVLEEDS, XIM, DIPITA

State of the art

Compositional and formatting tools belong to the set of digital content production tools. They have to help content designer to:

- efficiently collect needed components, using advanced query options
- find/produce alternatives for those components that may present distribution problems (e.g. files too big, IPR or usage clearance issues, etc.)
- structure components, highlighting the semantic relations among them
- bind content structure to some presentation styles
- format broadcast/broadband-quality content for delivery to a variety of channels, eventually requiring repurposing or even re-authoring
- support different delivery channels according to various formatting styles and constrains reported in the final user's device profile

Automatic content production problems are decomposed in two separate phases: composition and formatting. The **Compositional** and **Formatting Engine** will respectively perform such phases. Such engine can use services provided by some tools involved in AXMEDIS object production.

Composition is the action of putting together content components to create a new digital item in an almost automatic manner, based on a set of user-defined rules. The final result is a new composite AXMEDIS Object. The compositional activity should allow composing different kinds of raw assets such as Text, Images, Audio, Video (actual shot), Animations (synthetic), etc… coming from AXMEDIS database. The composition may produce homogenous (where all digital resources are the same type) or heterogeneous (different kinds of digital resources) composite AXMEDIS Objects.

A challenging part in the automation of composition will be the management of metadata as in such activity the user will be highly relaying on objects metadata to identify, select and operate on digital assets. This is already the habit in most editorial environments and tools. In more detail it is important to note that metadata are used during the search phase, and are crucial in the classification and archival phase. Research concerning automatic extraction of high level metadata, and related state of the art, is covered in T4.2.2 and specified in the related section.

Formatting is the process of exploiting digital resources in a combined AXMEDIS Object to some integrated visualization (editorial) format for distribution to the end user. A simple composite object comprising several parts (e.g., an audio, a video and a document) may be formatted in different ways according to various formatting styles (graphic layout, spatial constraints, file size, quality limitations, content temporal scheduling, speech generation from text, synchronization between audio and images, etc.)

and adapted to produce a "final content" (Digital Item Adaptation) to be distributed via different channels to users' platforms such as i-TV, mobile, PC, etc…
Generally speaking, a formatted multimedia object can be seen as a unique object characterized mainly by:

- *Spatial relationships* – They are defined in the spatial domain and describe how visible resources are organized in the visualization space (layout, margins, etc…)
- *Time relationships* – They are defined in the time domain and describe not only the basic orchestration (i.e. when media items appear and disappear from the screen), but also which synchronization relations need to be maintained (e.g. audio streams lip sync with a given video stream) and how it is possible to control speed and other basic behaviors of animated content (in time domain).

Other relevant aspects to be taken into account are the navigation structure and hierarchies inside the object. For example in many multimedia objects there are at least two different sets of possible navigation paths that allow the user to exploit the content. Usually there is a linear or sequential path and a hyper-textual one. This basically allows defining some additional relationships to be added to the previously mentioned ones and that may be present or not:

- *Navigation and connection relationships* – They are defined in the content structure domain and provide information on the content components structuring and navigation path and relations. Such relationship may be limited to object structure, but can also connect several different object or content sources. In the latter case there is a direct connection with the *Context and dependency relationships.*
- *Context and dependency relationships* – They are defined in the metadata structure domain and provide information on the content context and dependencies relations.

This latter point may account for objects level of complexity and (in case of occurrence) imply some limitation to the results achievable through automatic formatting.
All above-mentioned relationships and aspects (spatial, time, and hierarchical) will be analyzed in this task and linked to the corresponding views developed in T4.1.3.

All the formatting activity shall be based on content features, generic user profile and needs, specific user profile (in case of formatting on demand), formatting style, optimization parameters, end-user device profile, interactivity level and paradigms, content type and features, metadata, categorization, business information (price, localization, etc.), temporal evolution, DRM rules, delivery time, etc…
The formatting will have to take into account also the specific problems of the distribution channel:

- Content location and time to delivery (*time for processing and eventually storing of content to be manipulated).*
- Business and transaction models, thus DRM of content
- Delivery model: streaming, download, off-line distribution, etc.
- Delivery format: MPEG-4, simple audio files, documents, video, etc.

For example a user working on a PDA or a mobile wants to play a video clip, which requires a broadband network to be played; the formatting process has to adapt this video clip so to satisfy transmission constraint and to provide to the user a service with an adequate quality.

As with the composition activity, AXMEDIS aims to automate the formatting activity as far as possible, by allowing the user to define formatting rules (based on the above characteristics) which will then be applied to multimedia objects to prepare them for the required distribution channels, business models, delivery formats, etc.

For the analysis of the state of the art the following activities have been performed:

- Review of commercial and non commercial tools
- Analysis of the state of the art in adaptation algorithms for different media
- Monitoring and analysis of the standardization regarding DIA in MPEG-21
- Collecting the internal state of the art, what the partners are using for content composition and formatting
- Compilation of a list of references
- Identification of the main tools on the market, verification of their features and capabilities
- Preparation of a review document of the state of the art
- Identification of constraints imposed by the
    - distribution channel
    - final format for delivery
    - client power and profile

Research and development in composition and formatting (DSI)

The MPEG-21 and WEDELMUSIC models are compositional models in which any kind of content can be integrated. In addition, in both models the DRM aspects and several relationships between the content components included can be defined.

In this area the following aspects have been considered and developed:

- definition of a language for content composition/formatting process description
- implementation of an engine for processing scripting for content composition/formatting
- language dedicated to the layout of the content format.
- definition and implementation of the rule structures for AXMEDIS Composition and Formatting Rules Editor
- definition and implementation of AXMEDIS Composition and Formatting Process Engine
    - the engine is based on GRID technology
    - scheduling capabilities
    - process control capabilities
    - composition and processing rule profile processing according to the node capabilities for process allocation.
- elicitation and analysis of the expertise and current manual processes used in the production workflow
- connection of composition/formatting algorithms and tools to the engine and the editor:
    - Based on technical metadata and licensing aspects
    - For automatic definition of DRM rules (Possible Available Rights) based on profiles: distribution, device, channel, user, player tool, etc.
    - Produced by example, the user could imagine to provide an example of composition and the tools could process the other files accordingly
    - actions on sets such as the merge and selections
    - engine interpretation and execution of scripting for automatic processing of composition and formatting
    - the formatting angine is based on reasoning on resource descriptors, user profile, device profiles, style descriptors, template descriptors, etc. In addition an optimization phase has been realized based on Genetic Algorithms for optimizing the identification of layouting paramters.
- connection of composition/formatting algorithms and tools with several available integration and optimization technologies, such as:
    - For graphic formatting tools to allocate in the visual and temporal domains different visual sources, integrated with optimization issues
    - For integration of accessible algorithms and solutions for the comparison with the innovative algorithms that will be developed
    - For integration of content formatting engine with commercial or open source tools via plug-ins
    - For integration of adaptation and content processing algorithms for several media

    - For integration of adaptation and content processing, including synchronization and editing
    - For managing relations between textual information in multimedia objects
    - For accessing to databases and making queries automatically
    - For accessing to network and system resources
    - For controlling the exploitation of resources in the GRID nodes
- research activity on Content Composition and Formatting and integration into the several workflow processes of content producers, aggregators and distributors
- adaptation of the content producer partners' workflows to support AXMEDIS framework and integration of the new tools into these workflows; this will include:
    - creating initial composition and formatting rules for each content producer partner
    - integrating AXMEDIS tools via plug-ins and interfaces with the producers' existing production tools and WFMS (where currently used)
- testing the model for the production/collection of test cases
- verification and production of several test cases derived from the defined Test Plan
- validation of the new environment for content composition/formatting by using content produced by WP8. This involved each content producer partner in:
    - working through the Test Plan, running test cases
    - identifying and documenting any missing rules, formatting language elements or critical functionalities in the prototype AXMEDIS tools

2.1.2 T4.3.3 Workflow Support (IRC)

Major partners involved

IRC, with HP, XIM

Knowledge Engineering Pre-requisites

The design and specification of the Workflow Integration to be undertaken as the planned activity for the forthcoming phase was based on three earlier preparatory stages of activity undertaken to accomplish a systematic requirements analysis, state-of-the-art research review and knowledge engineering-led transactions and states analysis. Thus in setting out the rationale for the proposed research work as part of the planned activity we shall suffice to give only brief references to some of the above results as these were fully reported in documents submitted earlier. Thus this section will focus on the salient aspects of the requirements and specification of the plug-in interfaces between the two selected Workflow Management Supports (WFMSs) to be integrated with the relevant AXMEDIS components. Accordingly the present specification is based on the conclusions of three distinct planks of preparatory research carried out by the workflow group (IRC, HP, XIM); outlined as follows:

A) Established the state-of-the-art from the standpoint of the relevant business semantics, workflow knowledge and scenarios that are applicable to the three key sectors targeted for exploitation of AXMEDIS Framework, namely e-book, e-media, e-music production and distribution workflows.

The required domain knowledge elicitation was supported by the design and distribution of a questionnaire and a series of follow-on semi-structured interviews with selected practitioners from each of the above sectors. The results were documented in the multi-sectorial Workflow Requirements Elicitation and Domain Knowledge Analysis part of DE2.1.1a, *User Requirements* and use cases.

This led to

- a novel analysis of AXMEDIS Object lives and their LifeCycles
- the analysis of the metadata and descriptor states required for the workflow-triggered traversal of reasoning over such LifeCycles hierarchy that takes place in the normal course of AXMEDIS Object tracking and control

- and how this involved three distinct interaction spaces i.e. three workflow-centric inferencing environments re Object states reasoning, tracking and control

It was then suggested that the above three possible sub-spaces of AXWFM-AXMEDIS interaction could help clarify and streamline the design of AXWFM-AXMEDIS interfaces as well as direct the focus of any metadata and database partitioning (i.e. AXWFM DB- AXdb complementarity) that could be efficiently deployed using the canonical 10P-stamp situation assessment classes of Object metadata states; with its two sub-fields namely *purpose* and *period* used for filtering the focus on metadata in a context-sensitive manner. This was set out in the respective Workflow Sections of DE2.1.1a, *User Requirements* and use cases.

This work in turn led to suggestions re the design of the AXMEDIS Object Schema in terms of the specific states and descriptors required to be included with the Object metadata (AXinfo) and/or in AXWFDB and/or AXdb or in the PMSdb to allow reasoning over all aspects of Object history and full Object traceability and tracking.

This foundational requirements engineering work led to the conceptualisation of the prototypical scenarios of workflow usage and thus the derivation of the 23 Use Cases and associated Test Cases for workflow interaction. This in turn supported the rationale for a four-Channel set of interfaces for a streamlined integration of workflow and AXMEDIS.

B) Established the state-of-the-art of the available workflow technologies both as open-source and proprietary products and performed comparative analysis of their compatibility for integration with AXMEDIS framework. This has led to a rationalization of the choices re the adoption of the candidate workflow systems for integration with AXMEDIS Framework. This was reported in the respective Workflow Section of DE2.1.1a, *User Requirements* and Use cases.

C) Established the knowledge engineering and software development analysis base from the standpoint of the need for specification of the sub-systemic workspace boundaries for states representation, and, *the technology and the protocols* for the above integration leading to the specification and design of the transaction Semantics, Syntax and Protocols for the interfaces required to achieve the various integration scenarios as specified.

In what follows we will first outline the state-of-the-art functional requirements from all workflow participants (Users, AXFWM, AXMEDIS components) as it has helped set the guidelines for the research and development work to be performed in the next phase to achieve the Workflow Integration within AXMEDIS Framework. Thereafter we will briefly refer to the rationale for the planned research in terms of the choice of existing workflow systems to be adapted and interfaced to AXMEDIS Framework and the knowledge engineering-led choices re the selected sub-systemic boundaries for the required 17 distinct interface instances to be delivered within the following four-channel architecture for the workflow integration:

i. AXWFM and AXMEDIS Workflow Editor,
ii. AXWFM and AXMEDIS Rule Editor/Viewers,
iii. AXWFM and AXMEDIS Engines
iv. AXWFM and AXMEDIS Query Support Interface

We shall conclude with an outline of the design specification and the technical environment of its implementation as part of the future planned activities together with the associated schedule of milestones for the deliverables

Established Requirements for the Integrated AXWFM-AXMEDIS
This is the outline of the requirements now established for AXWFM Integration as follows:

i. Operate within the key Operating Systems (OS); for example the Windows, Linux, Mac

ii. Interact with AXMEDIS Object Manager to access Objects and track/update their status (i.e. allow workflow metadata visualisation, editing, automated updating and storage).
iii. Monitor the progress of assigned process activities and be capable of managing more than one workflow process instance so as to provide workflow support for multi-agency co-design and co-production of multimedia content based on open-source distributed products through LGPL, BSD or similar licences.
iv. Provide time-and-status metadata updates that remain accessible to other Enterprise Project Management Applications, such as SAP for example (OPTIONAL)
v. Provide a seamless service interface (API) to be used for developing the plug-ins for AXMEDIS-native tools (e.g. tools for Content Production, Formatting, Packaging/Bundling and Distribution) for the range of operating systems selected above, i.e. specifically to provide interfaces for the following tools and engines as listed below:

a) Editor
b) Rule Editor/Viewers for various tools
c) Composition and Formatting Engine
d) Programme and Publications Engine
e) Protection Tool Engine
f) P2P Active Selection Engine
g) Collector Engine
h) Publication/Loading Rules/Selections Editor
i) Publication Tool Engine of AXEPTool
j) Loading Tool Engine of AXEPTool
k) Administrative Information Integrator
l) Administrative Information Manager
m) Accounting Manager and Reporting Tool
n) User Query Support

AXMEDIS is expected to deliver a number of innovations to the media production workflow:

- Support single and distributor customisable workflow for all channels - content should need to be authored only once
- Support for peer-to-peer production workflow – collaborative production and integration of content across remote ad hoc virtual organisations that may form and re-form on a project-by-project basis
- Support artificial intelligence-assisted workflow - repetitive tasks, in particular re-purposing for multiple channels, is to be automated
- Enable integration into existing workflows - AXMEDIS is to provide a framework to support workflow, rather than dictate a single, inflexible model. This will allow production houses to retain their unique styles and original creative aspects of their current workflows

The interface will be developed to ensure maximum compatibility with the requirements of the various parts of AXMEDIS Framework to deploy the workflow environment for example for automated content composition and formatting, or content distribution through various channels such as B2C over P2P networks.

The Workflow is to be tightly integrated with AXMEDIS Viewer/Editor tools, in order to be able to automatically streamline editing/viewing activities inside publishing and distribution processes. With this kind of integration, the Workflow automatically launches AXMEDIS Editors/Viewers on AXMEDIS Objects.

The AXMEDIS Viewer/Editor tools can be executed outside the Workflow environment. This is to accommodate those enterprises that wish to work with AXMEDIS Objects but without necessarily adopting

structured processes – this is to be supported through the provision of a simple AXMEDIS Object check-in/check-out interface.

On the Client side, the AXWF User Interface will be the home interface for all users and actors involved in the product development and distribution processes. Through the AXWF User Interface the actors logs-in and see all the workitems in which they are individually committed and via accessing any workitems they can perform the actions required through launching the appropriate tools. All actions performed by the actors/users are logged by the AXWF in the AXMEDIS Object repository, as well as the new revisions and status changes.

Through the AXWF User Interface it must be possible to create new or delete existing Objects or Components which in turn will create or delete (sub) process instances. Once the actor/user has selected a workitem to work with and an editing activity to be performed (e.g. editing, composing, formatting, etc an Object/component), the AXWFM will lock the Object/component in the Object repository and will copy it to a work area for exclusive access by the user where it will be processed by the proper tools (authoring/formatting/rendering/composing/ packaging/bundling).

Accordingly the key patterns of deployment in scenarios for the integration of any AXMEDIS-adopted workflow management system (AXWFM) with AXMEDIS Framework involve the AXWFM launching the execution of various plug-ins as follows:

- AXMEDIS Object Editors/Viewers via the AXMEDIS Editor WorkFlow Plug-in
- AXMEDIS Compositional/Formatting Engine via the AXMEDIS WorkFlow Engine Plug-in
- AXMEDIS Collector Internal Engine via the AXMEDIS WorkFlow Engine Plug-in
- AXMEDIS Program and Publication Engine via the AXMEDIS WorkFlow Engine Plug-in
- AXEPTool Loading Tool Engine via the AXMEDIS WorkFlow Engine Plug-in
- AXEPTool Publication Tool Engine via the AXMEDIS WorkFlow Engine Plug-in
- AXMEDIS Loader/Mover via the AXMEDIS WorkFlow Query and Database Interface
- AXMEDIS Query Support via the AXMEDIS WorkFlow Query and Database Interface
- AXMEDIS Compositional/Formatting Rule Editor via the AXMEDIS WorkFlow Rule Editor Plug-in
- AXMEDIS Program and Publication Rule Editor via the AXMEDIS WorkFlow Rule Editor Plug-in
- AXEPTool Publication/Loading Rule Editor via the AXMEDIS WorkFlow Rule Editor Plug-in
- AXMEDIS Protection Rule Editor via the AXMEDIS WorkFlow Rule Editor Plug-in

Thus the AXWFM Workflow Manager will be capable of interacting with the AXMEDIS Object Manager via the AXMEDIS Editor WorkFlow Plug-in

It is expected that every AXMEDIS tool will update the AXMEDIS Object tracking information, while performing its actions. However, when the check-in / check-out interface is used, it is up to the Workflow Object Manager to update such information. Unfortunately the Workflow cannot know exactly what action the user performed between the Check-in and Check-out stage. Accordingly a field is to be provided for users to describe the work that was done on the Object concerned; thus allowing status updates.

Specification of AXWF-AXMEDIS Interface Types

In summary four types of interfaces are expected to form the basis of AXWFM interactions within the AXMEDIS framework. These types will depend on the characteristics of the different applications or tools that the AXWFM must exchange information with.

These interfaces are outlined below:

a) AXWFM-AXMEDIS native tools interface: the AXMEDIS client tools form a crucial part of the AXMEDIS framework and will therefore present a closely knit form of interaction. These native

tools which deal with aspects related to authoring, formatting, rendering, composing, packaging, and bundling applications all perform specific actions on an AXMEDIS Object.

b) AXWFM-CMS interface: External CMSs must be allowed for as users must not be impeded in their work. Interfaces to such systems are necessarily less well integrated relative to native AXMEDIS tools. An essential part of this type of interface will be the check-in and check-out operations of an AXMEDIS Object by an authorised actor. The former operation performs tasks related to maintaining Object consistency by locking it in the repository and authorising its download exclusively to such actor. The latter operation releases this lock once the upload of the manipulated AXMEDIS Object has been successfully completed and status / revision entries have been inserted in the AXWFM.

c) AXWFM-AXMEDIS Server Engines interface: The web services technology can be used to integrate the AXWFM and server-based AXMEDIS applications such as the AXMEDIS Object Manager, Program and Publications Engine, Protection Tool Engine, Publication Tool Engine of AXEPTool among others. Using web services, the AXWFM can invoke the proper methods over HTTP(s), enabling one to interface any server-side component, regardless of language or platform or location and vice-versa.

d) AXWFM-Query support interface: this type of interface is a particular case of the AXWFM-AXMEDIS server engine interface which is to be based on the web services technology. In this case an authorised user may perform AXMEDIS-related queries through the AXWF user interface. The user must be afforded the possibility of performing advanced searches on characteristics connected to the AXMEDIS Objects by invoking external search engines and thus retrieving the information being sought.

State of the art

This part gives an outline of the rationale for the choice of the two selected workflow systems (i.e. the open-source workflow, Openflow, and the proprietary workflow system BizTalk). Later we shall describe the framework for the design of the plug-in input/output structures to be realised such that they are compatible with the protocols and method invocations of the adopted workflow systems.

The main WFMSs that have been examined are as follows:

i. Openflow
ii. JBoss jBpm
iii. ObjectWeb Bonita
iv. Enhydra Shark.
v. OpenWFE

Rationale for the Choice of Openflow

The Open-Source Workflow tool was to be selected on the basis of the following criteria:

- Best coverage of the functions needed by Axmedis
- Openness of the interfaces
- Usability considerations
- Robustness, based on project references

The Rationale for choice of BizTalk

a) Following the extensive process of Multi-Sector Requirements Knowledge Elicitation and Analysis as undertaken in the initial stages of AXMEDIS project together with the state-of-the-art review of all the major WFMSs it was concluded that:
b) there is no dominant WFMSs currently deployed in the multi-media Production/Distribution sector
c) MS NET BizTalk offered the greatest scope for this AXMEDIS integration.

Research and development plan

Research and development in workflow (IRC)

AXMEDIS technical annex set out the following Objectives in the present context:

- Adaptation of the content producer partners' workflows to support the AXMEDIS framework and integration of the new tools into these workflows
- Definition of the expertise and current manual processes used in the production workflow in order to create tools based on artificial intelligence technologies
- Research activity on Content Composition and Formatting and integration into the several workflow processes of content producers, aggregators and distributors
- Consolidation of tools and workflow developments to be compliant with the overall AXMEDIS framework

The research and development activity during this period mainly included the examination of the specification of the plug-ins from the point of view of the integration technology required for the delivery of a fully interoperable set of the plug-ins with the APIs available from Openflow.

This is to serve the integration with AXMEDIS native engines, tools and the Query Support Interface. This implies the design of the plug-ins for full Connectivity, messaging Communication, Command and Control (C4) as well as AXMEDIS Object transfers across the relevant interfaces in particular such that the workflow will be able to provide seamless interaction with all the relevant AXMEDIS components involved in any of the *eight generic Workflow Scenarios* that are already proposed and established by the workflow team.

For this we envisaged that the design specification of the required interfaces involves the specification of:
- Programmes and Publications Engine
- AXEPTool Publishing Engine
- AXEPTool Loading Engine
- AXEPTool P2P Active Selection Engine
- AXEPTool Objects Monitoring Tool
- AXEPTool Publication & Loading Editor/Viewer
- Programme & Publications Rule Editor/Viewer
- Composition Engine
- Compositional Commands & Reporting
- Formatting Engine
- Collector Engine
- Protection Engine
- AXOM Editor Plug-in
- Compositional Rule Editor/Viewer
- Protection Tool
- Protection Tool Editor/Viewer
- User Interface

Research has been carried out for the design and development of the above interfaces between the AXMEDIS Workflow (AXWFM) and the relevant core AXMEDIS components (tools/engine/Query

Support) to facilitate indexing, tracking, storing as well as optionally to support Authentication, Authorisation & Auditing (AAA) with single sign-on and all-in single-billing. These interfaces have already been implemented and are now tested for a real-life scenario in one of the Axmedis partners factory.

Accordingly Research has been carried out to finalise the Exchange Format and Metadata states required. An Exchange Format must satisfy the following specification criteria:

- **i.** must be able to identify and index itself (Self-referentiality Criterion)
- **ii.** must carry the user_credentials (including session_ID and any other data that may be required for AAA of the originating client-session-owner), (AAA Criterion)
- **iii.** must be able to bear an explicit link or implicit pointer to link it to other co-related session exchanges to which it refers or which might refer to it (Co-referentiality Criterion)
- **iv.** must be able to convey data sufficient for the goal of the exchange to be achievable such that each transactor in turn need only process elements of data sufficient and necessary for it to achieve the exchange sub-goals for which it is responsible within its own sub-system environment (Necessity and Sufficiency Criterion)

Each Exchange is an Exchange-instance and has to be indexable as an Exchange_Instance_ID. Any Exchange between the adopted AXMEDIS workflow system (AXWFM) and relevant AXMEDIS service providing components (Editors, Engines, Tools, Query Support) can thus be uniquely distinguished by means of an AXWF-Exchange-instance –ID or WF-Exchange_ID for short.

All these exchanges were developed in order to be compatible to the selected workflow environments and are a neutral exchange format for accessing any transaction between the AXWFM environment and any relevant AXMEDIS component whilst imposing minimal metadata requirement on the AXMEDIS Object Model. An efficient way of handling the transactions between the transactor (typically AXWFM) and other tools will be developed along with history information of such transactions over the period of any sessions so as to be able to provide the data structures to support a viewer-friendly re-call of all types of session exchanges as well as the session costs, rights granted and used, DRM and billing.

The AXMEDIS Object metadata and the internal workflow engine lifecycle status data when fully integrated provide all the information needed to enable the workflow management system to locate and track the progress status of any involved entity anywhere in any (sub) workspaces to enable the user to enquire about the lifecycles status of the three interacting types of entities (workflow, actors, Objects) involved in any project.

WF-Exchange_ID Protocols and Methods

An Exchange_ID has defined as a logical neutral format, that specify what must logically, minimally and necessarily be passed but it will rightly defines itself not in terms of some local dialect from any particular WFMS currently in the market but in terms of generic knowledge engineering-led ontology which is designed exactly to ensure efficient effective as well as complete, consistent, and coherent messaging.

So as long as all the necessary data from WF-Exchange_ID, to be specified, are made available appropriately to various contexts of transaction, i.e. to each transactor as required, so that:

- **a)** Those systems that need specific Exchange data elements in order to pass the right data back and forth can efficiently find, within the WF-Exchange_ID, the data for targeting, linking, binding and tracking their Requests/Responses and any related Objects

- **b)** Those systems that do not need to know/process certain elements of the Exchange data are not forced to process those elements

Then the WF-Exchange_ID design have satisfied the requirements.

In this way a generic model for AXWFM-AXMEDIS transactions is specified such that it can abstract from the idiosyncrasies of design/processing modes within individual adopted workflow engines whilst providing a vehicle for sufficient and necessary status data/parameters to be exchanged between any AXWFM and the relevant AXMEDIS components through a single composite string structure (i.e. WF-Exchange-ID) that is transparently and uniquely decodable and decomposable within the individual *participant sub-systems* to provide all the necessary lifecycles information of interest locally as well as to be able to index various sub-fields that need to be combined to provide multiple or particularised views.

The Neutral Exchange Messaging Format has to be specified for the plug-ins as the logical, necessary and sufficient transaction protocol to remain both AXMEDIS-compliant as well as technologically realisable and consistent with the capabilities of existing workflow systems given their available method invocations and transaction protocols. Thus the reason for adoption of a *neutral exchange messaging format* is to provide a uniform and generic transaction standard so as to allow portability of the interfaces to various workflow systems and facilitate the development of new plug-ins.

Research has also been done for efficient tracking of digital assets within AXMEDIS framework. For this the Object's metadata information is considered to be that which allows relevant access to the status and tracking information regarding the processing and progress of the work done on any Objects and the results of various actors/tools/engines acting upon the Objects through the lifecycle of various projects. In general, the chief requirements for tracking and control is *focused situation assessment* and the semantics types recommended for the metadata and status descriptors are intended to provide the situation assessment on current state and progress of any digital assets developed during any project.

This can be done by metadata partitioning and assigning a *10-P situation assessment criterion* (as defined in DE3.1.2G Framework and Tools Specifications. This 10-P status data type is meant to provide a sufficient basis for tracking the status of AXMEDIS Objects through their development life cycles whilst allowing purpose-specifically filtered metadata-view and reasoning to serve the current focus of the actors/user/tools concerned.

The workflow rules are defined based on the development process involved in the production and distribution of multimedia artefacts, which have been studied with the cooperation of the AXMEDIS partners involved in the respective fields (e.g. OD2, ILABS, XIM, AFI, ANSC, SEJER, etc)

Mapping the WF-Exchange_ID to WebServices XML Envelope or DCOM Call in .NET

The above data exchange requirements are protocol invariant but be deliverable by for example the XML-RPC protocol over HTTP as deployed by some workflow engines such as OpenFlow.

It must be noted therefore that the WF-Exchange-ID developed contains the *method invocation*, as well as INPUT/OUTPUT/NOTIFICATION parameters i.e. it is not just another parameter in the INPUT/OUTPUT/NOTIFICATION class invocation. So the WF-Exchange-ID format (which is basically a text string) should be compatible with any communication protocol as desired.

Where WebServices are deployed for AXWFM transactions with the AXMEDIS Service Provider Components such as tools/engines/editors etc, the WF-Exchange-ID are encoded as an XML string and accordingly its parameters (e.g. the invoked engine and method etc) finds an appropriate mapping or expression in the WebService XML envelope.

Openflow as the AXWFM Interfacing with the AXMEDIS Components

All the possible Workflow engines in the Open-Source arena that are suitable for AXMEDIS offer a Web application based User Interface.

The User Interface is then executed via a normal browser, while the User Interface logics reside on the Application Server of the WorkFlow Engine.

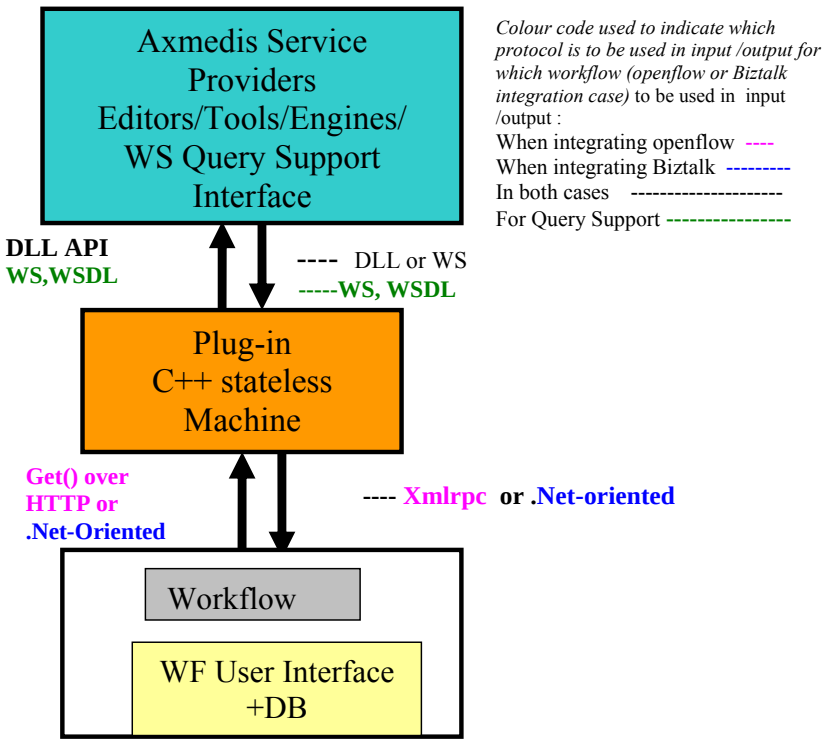
The WorkFlow Manager will be a customisation of one of the adopted Workflow Management System with some functions added, if not already present (e.g. search engine).

The Workflow User Interface will also require some customisation to serve the integration of the adopted WFMS with the AXMEDIS Framework. The User Interface will use the WorkFlow Manager class methods via the internal interface of the adopted WFMS.

Calls from external applications (AXMEDIS tools and engines) to Openflow are to be made via http protocol, using the classical GET method (parameters given in the URL).

In the case of most potential AXWFs, including Openflow, five parameters tend to be always required to be passed to an application e.g. to an AXMEDIS service provider: openflow_ID, process_ID, activity_ID, instance_ID and workitem_ID.

The Interface Protocols Diagram below illustrates the protocols to be deployed for the range of AXWFM-AXMEDIS plug-ins as listed above.



2.1.3 T4.3.4 Adaptation Support and Algorithms

The activity on content adaptation has been delayed to anticipate the work on content processing. The first activity on this topic will be on integrating content adaptation algorithms for video, text, image, document, metadata, license, DRM, etc. These algorithms have to be realised by exploiting external libraries connected by means of plug-in and directly implemented in terms of C++, or AXCSP java script code. In this manner, they become directly accessible for both AXMEDIS authoring tools and AXCP java script. In this area specific innovative algorithms for content adaptation and transcoding will be developed for mobile ringtones. The adaptation algorithm will be profiled in order to classify and access them directly without static linkage. Just to remark that all the algorithms for the adaptation has to be developed, according to the specification, to be used as plug in of the AXMEDIS plug in manager or directly linked to the players. In this view, in this task an user manual for the development of plug in for the AXMEDIS plug in manger will be produced.

For IRC, the development of multimedia systems has had a major influence in the area of image, audio and video coding. The problem of interactivity and integration of audio/video data with mobile devices is relatively new and subject to a great deal of research worldwide.
The key design goals of mobile transcoding will include 1) to maintain the perceived quality during the transcoding process and 2) to keep the transcoding process complexity as low as possible. In order to make available the media files (typically audio) to be used as a ringtone on various devices, the media should be converted from one file format to another using a transcoding algorithms. Even for the same file format different devices requires different characteristics (e.g. bit rate, sampling rate, etc) for the media. Media adaptation for mobile will cover transcoding, formatting algorithms for distribution on-demand to mobile devices and new generation of PDAs. Updating and extending profile resolution and transcoding algorithms as part of the evolutionary refinement of the AXMEDIS Framework for distribution to mobiles. This is to cover the new leading mobile devices profiles that will be introduced in the near future such as the Sony-Ericsson combined mobile phone and "walkman" that has just been announced or more efficient transcoding algorithms for distribution to mobile devices as applicable.

For FHGIGD, besides the libraries that are being integrated in the AXFW and in particular with the AXMEDIS plug in manager and AXCP java script, additional functionality will be provided to the AXMEDIS framework by integration further algorithms. As video adaptation (and digital item adaptation as well) is a currently emerging field, the research activities will be monitored and research organisations will be asked for the permission to integrate their algorithms within the AXFW. As identified in DE-4-3-1 the focus of the monitoring includes for example permissible and secure adaptation, semantic clues for adaptation, and modality conversion.

For DIPITA, a first prototype for text documents transcoding ha been developed by integrating the XPDF and HTMLDOC libraries. The demonstrator works as a command line interface tool. The next steps foresee the exploiting of the rest of the libraries in order to allow the transcoding between all the possible pairs of formats (rtf, ps, pdf, html, txt). The tool will be delivered as a dll library, however it will have to cope with the license of some of the exploited libraries (which is GPL). The designed solution to comply the AXMEDIS requirements foresees an executable program (licensed under GPL) which directly access the libraries, while the dll calls this executable using a communication mechanism to be defined (it will be provided by DSI).

For EPFL, a first prototype of audio adaptation algorithm has been developed by integrating the ffmpeg and libsndfile libraries into a single command line application allowing easy audio files transcoding, sampling rate conversion and channel mixing. As a second step in the following months, more functionality will be added including a time-stretching tool to facilitate synchronisation between audio and video streams and a number of transformations of the audio signal (including volume normalization, equalization, reverberation…). MPEG-21 Digital Item Adaptation Profiles will be supported so that files can be automatically adapted based on some specific user preferences or needs (MPEG21 AudioPresentationPreferences and AuditoryImpairment) and/or based on the capabilities of the output device (MPEG21 AudioPresentationPreferences and AuditoryImpairment). The tool will be accessible through the plug-in interface and through the AXCP java script.

Currently, adaptation algorithms are configured by means of parameters passed according to a format defined by AXMEDIS profiles (expressed in XML). In the next versions of these algorithms MPEG-21 DIA descriptors will be used as input, therefore the XML profiles shall be modified accordingly. A multimedia files adaptation tool will be developed using the functionalities of the GPAC 0.4.0 library. GPAC is a multimedia framework based on the MPEG-4 Systems standard developed from scratch in ANSI C. The tool will notably allow the creation of IsoMedia files (i.e. files based on the MPEG-4 Part 12 specification: MP4, 3GP and MJ2K files) based on some simpler files such as MPEG-4 audio and video, MPEG-1/2 audio and video, JPEG images, AVI files, SRT subtitles… File splitting by size or time will be supported as well as the extraction of media track from multimedia files. A number of conversions between complex formats will also be possible. A critical issue to be investigated and addressed is the implementation of a transcoder from SMIL to MPEG-4 BIFS and vice-versa. This activity will evaluate the possibility to use the XMT-O editing tools developed by IBM (http://www.research.ibm.com/mpeg4/Projects/XMTInterop.htm). XMT-O is a part of the XMT architecture developed by IBM conceived to provide interoperability between SMIL and MPEG-4. An evaluation of its adoption needs to be performed since the toolset is developed in Java and it is available only for 90 days in trial version and licenses costs range from 500$ (1000 toolkits for internal use only) to 5000$ (500 toolkits for internal and/or external use and distribution). The tool will be accessible through the plug-in interface and through the AXCP java script.

For UNIVLEEDS, the main metadata adaptation functionalities include tools for Dublin Core and AXInfo adaptation. The main required functionalities include the scaling of the metadata by filtering elements and adapting fields where names, format, sizes or others are changed during the adaptation of the object. This can be achieved by defining lookup tables to define valid and invalid elements and to adapt metadata from one standard to the other.

For FUPF, there are different rights expression languages that can be used to describe what users can do with content they have purchased. Moreover, users may want to use this content in different devices, with may have limited processing capabilities. For those reasons, DRM adaptation may be needed. In some cases, the adaptation will involve translation between languages. For instance, when a user has a license expressed in one language, but the system providing the content uses a different language, translation has to be performed before user can be authorised. In other cases, when a user wants to make use of content in a device with limited process capabilities, like current mobiles, adaptation may involve the simplification of rights expressions in order to allow DRM to be processed into those devices. The continuation in the study and implementation of the possible adaptations is foreseen for this period.

For DSI, it is very important to find a solution for managing in dynamic and uniform manner the software components related to the different algorithms involved in the content processing area and tools but also in the tools for authoring and playing content. They belong to different area of work such as: adaptation and transcoding including decoding and encoding, protection tools, descriptors extractor tools, fingerprint extractors, etc. On this regard, DSI has started to work on a middleware to exploit these components managing their composition, deployment, certification, remoting, etc. To this end, it is very important to monitor and contribute to the activities related to multimedia middleware that are running in the market and in the standardisation arenas such as MPEG M3W, Multimedia Media Center, etc. If they will address the needs of AXMEDIS, the AXMEDIS platform could become in a easier manner compatible with a large number of software components for content processing.

Research and development in adaptation (FHGIGD, DIPITA, EPFL, FUPF, IRC)

The following aspects and algorithms will be taken into account:

- Formalization of distribution channels, devices, tools, user profiles and the class hierarchy for their loading and saving, and algorithms for their processing
- Formalization of adaptation algorithm profiles and the class hierarchy for their loading and saving, and algorithms for their processing
- Definition and implementation of a common framework and library for the adaptation algorithms and its

usage from AXMEDIS Editor, Composition and Formatting Engine

- Definition and implementation of AXMEDIS Object Manager (AXOM) Content Processing interface for external plug-ins in conjunction with AXOM
- Digital Item adaptation for several media:
    - Documents and speech (DIPITA)
    - Video (FHGIGD, XIM)
    - Audio (EPFL, DSI, UNIVLEEDS)
    - Images: (DSI, EPFL)
    - Multimedia MPEG-4 (EPFL)
    - Multimedia IE (DSI, UNIVLEEDS, XIM)
    - PAR: Possible Available Rights (FUPF)
    - Metadata and AXINFO adaptation (UNIVLEEDS)
    - Etc.
- Integration of available algorithms and solutions for performance comparisons
- Definition of transcoding algorithms according to an extensible plug and play model standardized into AXMEDIS tool and framework
- Definition of transcoding algorithms according to portability issues that may concern PDAs, mobile devices, etc. (resolution, definition, illumination and number of supported colors, available memory storage, available computational capabilities, easiness of usage of the device and its GUI)
- Multilingual support for production and verification of multilingual metadata
- Usage and integration of commercial and open source text translation algorithms and tools: leading technology on translating text into several languages, and technology in vocal synthesis from text
- Collection and management of AXMEDIS Digital Item Adaptation Support
- Verification and production of several test cases

3 AXMEDIS Content Processing area (DSI)

In this section the status of work performed in defininig the model of AXCP area is reported. This report shows the main concepts, formalization and implementation of the model.

For technical details see document:

DE3.1.2.2.6	Specification of AXMEDIS Content Processing, first update of DE3.1.2 part C axmedis-de3-1-2-2-6-spec-of-ax-content-processing-upc-v1-5.pdf

The AXMEDIS Content Processing Area is mainly realized as a distributed tool for processing content on the basis of rules which are written as scripts. This solution allows considering such AXCP Rule Engine as a tool to be used for content processing according to the content designer and publisher needs.
The Rules and the Engine allow the definition of algorithms for composition, formatting, adaptation, transcoding, extraction of fingerprint and descriptors, protection, license manipulation, potential available rights manipulation, resource manipulation, load and save from databases and file system, mathematics, combinatorial, etc.
The AXCP Rules can be written by referring to a large set of common data types derived from AXMEDIS Framework, MPEG21, and general resource definition such as: images, documents, video, licenses, etc. It is a sort of engine the allow to script middleware algorithms for content processing.
Thus the AXCP Engine and Rule are a versatile and customizable way to produce and manage digital content respecting legal aspects, DRM, ownership, user and publishers requirements.
In this terms, a rule:

- describes what resources are involved in the processing (i.e. extracting digital resources from the AXMEDIS database by means of queries built on metadata and licensing information or from a composite AXMEDIS object);
- works on contenent by using distribution channel properties, user device features, user profile, etc…
- generates the final output using a specific integration format (MPEG-4, SMIL,…) or using DIP capabilities provided by MPEG-21 objects
- describes how to combine different digital resources and create relationships in terms of:
  - spatial relationships (for graphic layout, resource adaptation, …)
  - time relationships (for synchronisation, transitions effect, fitting (shrinking or stretching, cutting,)…);
- describes how to manage and combine DRM rules for the new formatted resource;
- describes operations or actions that have to be performed during the formatting process, for example:
  - which formatting algorithms have to be used (synchronisation, image scaling, resolution scaling, format conversion, etc…)
  - which external functionalities (by dynamic call to services provided by external tools) have to be used
  - Fingerprint estimation and application for the new composite item
  - Object ID assignment for the new composite item.
- describe how to protect resources

In addition, the AXCP area is capable to receive commands coming from the factory Workflow by means of a web service interface. Despite of several mentioned advantages, delegating the processing activity to a single AXCS Engine seems to be not the best solution since the amount of work in terms of elaboration time and the dimension of data that the engine has to manage can be very high in most of the content factories in which even millions of digital resources are managed per months. The main idea to solve this problem has been to design the AXCS Engine as a distributed environment of rule executors based on a **GRID infrastructure**. This solution maintains advantages of a unified solution and allows enhancing the capabilities of the AXMEDIS Content Processing area by running rules in parallel and rationally using the computational resources accessible in the content factory.

In the Figure, the relationships between the AXCP Rule Engine and the other parts of the AXMEDIS environment consisting of the *AXMEDIS Workflow Manager*, the *AXCP Rule Editor* and the *AXMEDIS Database,* are described. The figure shows also the main components and tools used by the AXCP Rule Engine.

AXMEDIS Workflow Manager to allow the reception of commands from connected workflow tools of the content factory and thus to control the AXCP Rule Engine and Editor activities.

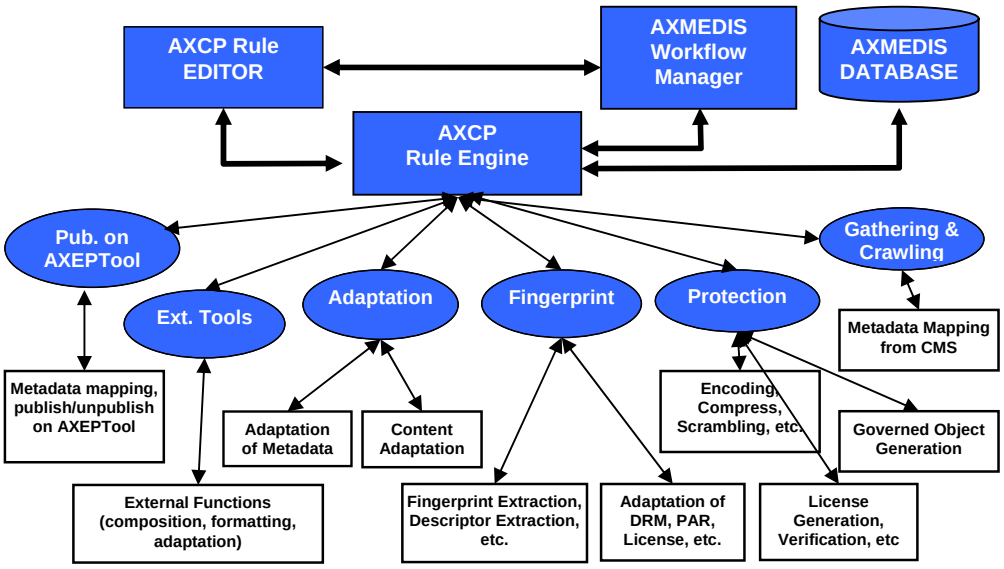
AXCP Rules Editor, which is front end tool to produce AXCP Rules for the user. It is supported by a repository of rules and interacts with the AXCP Rule Engine when a rule has to be executed.

AXCP Rule Engine to put in execution AXCP Rules. As a result of the rule execution, it may generate new AXMEDIS objects, licenses, etc. Such information may be stored in the AXMEDIS database to be further delivered via distribution channels. The Engine is supported by the set of plug-ins, called AXMEDIS plug-ins, that allows performing several functionalities and coping with:

1. **Protection** algorithms performing the protection of the AXMEDIS objects, license production and processing, etc.
2. **Fingerprint/Descriptor** algorithms for estimation of fingerprints and descriptors of AXMEDIS objects and contained digital resources.
3. **Adaptation** algorithms for content/resource adaptation, DRM and license adaptation, etc., that typically needed for different distribution channels and format paradigms.
4. **External tools** for the interaction with commercial tools and libraries for exploiting their formatting and processing functionalities.
5. **Publication** algorithms to perform metadata manipulation, gathering and mapping, publication of AXMEDIS objects on external channels and in particular on the B2B distribution.

This report describes the development activity performed in the AXMEDIS Content Processing Area and fix the current status of implemented features related to the following items:

- AXMEDIS Rule Model
- AXMEDIS AXCP Rule Editor
- AXMEDIS AXCP Rule Scheduler
- AXMEDIS AXCP Rule Executor
- Composition and Formatting tools, algorithms and formalizations
- Content Adaptation: Digital Resource Transcoding (audio, video, image, multimedia, etc….)



Content Processing Flow Diagram and Relationships

3.1 AXMEDIS Rule Editor (DSI)

Major Features accessible:

- Docking windows and toolbars
- Embedded the rule executor for running, compiling and debugging the rule (JavaScript code)
- Load & save of an AXCP rule
- Mozilla Browser based on wxMozilla contribution for browsing html help pages
- Dialogs/Tools for Edit Rule components
  - ♣ Header dialog for editing of rule header attributes
  - ♣ Schedule dialog for editing of schedule attributes
  - ♣ Parameter dialog for editing of attributes of a rule parameter
  - ♣ Dependency dialog for editing of attributes of a AXMEDIS PlugIn
  - ♣ XML Selection Editor (XML viewer/editor for the XML representation of selections) and JavaScript Editor based on Scintilla Editor for text/javascript code editing. It provides full editing capabilities (copy, cut, paste, redo, undo, syntax highlighting, etc…), print preview, page setup and print functionalities, syntax highlighting, brace highlighting, Folding/Hiding of lines, Breakpoint insertion/removal, Visualisation of line numbers
  - ♣ Selection editor for creating technical query on AXMEDIS Database and generating an XML for the selection parameter.
- Data interface to manage and synchronise tree items with rule items
- Drag and Drop of an XML AXCP Rule file
- Rule List: function for rule searching on the Rule Repository folder
- Debugging Rule functionalities: add/remove breakpoints, start debug, next breakpoint, step over, step into, stack calls monitoring, local variables visualization, Breakpoints list
- Print output window for visualising internal errors when script runs and monitoring the debug session
- A Library view (Info View) provide help on the set of functionalities provided by the Plugins installed and automatically detected by the editor. It was realised by using a Tree control that permit to show and browse plugins module and the functionalities that they provide according to their profile. The profile is used to build on the fly an html documentation page. The user can see the documentation associated with the selected function by interacting with the corrisponding item of the tree. To this end a contextual popup menu was designed. The required documentation is displayed in the integrated Mozilla browser**.**
- Activation and installation of Rule sends the current rule to the scheduler. A connection with the Rule Engine allow the migration of rules in the Scheduler repository.
- Checking rule tests the feasibility of the rule (like a compiler plus some tests on AXMEDIS objects)
- Importing and Exporting external JS script file (*.js)
- AXCP Rule Editor Configuration for managing gui properties and layout
- Command & reporting for communication with the AXMEDIS workflow
- Multi scripts management for a modular programming in Javascript
- Help on the JS class added to the Javascript

Work to be done

- Watching variables during debugging session
- Editing facilities for the script editor (intellisense, keywords suggestion, etc..)
- Rule profiling (estimation of some parameters such as the files complexity and required workload)
- Adding the access to the Metadata Mapper Editor
- Testing and solving possible bugs

AXMEDIS AXCP GRID Scheduler (DSI)

Major Features accessible:
The core scheduler is the manager of active rules. It is a multithread module. It detects, fires, launches, manages and monitors the execution of a rule. During its activity, the core scheduler:

1. preserves the scheduled work from interruption of service (crash of the application) giving the possibility to restore the last status of activity
2. manages and update the list of rules to be scheduled and their status
3. manages and update the list of available rule executors
4. notifies to the AXMEDIS Workflow Manager messages due to:
    - errors during the phase of rule association with an executor
    - errors due to the launching phase
    - errors during the rule execution on remote executor.
    - errors due to the time out deadline missing (the executor did not respond to request)

The following list reports the actually implemented functionalities:

- Selecting from the internal scheduled rules the rule that matches conditions for the execution. This is performed by:
    - checking the execution time and date
    - receiving an immediate run command from the AXMEDIS Workflow Manager
- Modifing and setting the time resolution for the control of rules execution
- Adding a new submitted rule in the list of jobs
    - Loading the corresponding rule xml file from the repository directory
    - Extracting the metadata for scheduling
    - Generating and assigning a Job Id to the rule
- Removing a rule from the list of jobs
- Running a rule on demand
- Rescheduling a rule (by overriding the schedule information)
- Overriding rule arguments (by replacing the current arguments)
- Checking expiration conditions of a rule
- Providing the list of jobs/rules
- Updating firing conditions of a periodic rule
- Browsing the list of jobs/rules
- Modifing the status of rules
- Removing an executor from the list of executors
- Providing the list of executors
- Browsing the list of executors
- Saving periodically on disk a backup copy of the list of jobs
- Restoring the last status by loading the backup copy of the list of jobs
- Tracing all activity by means logs
- Logs viewer
- GUI for the visualization of scheduled jobs. The current version is a simulator of a planned activity
- Interfacing with clients that require AXCP services by means workflow plugin
- Sending notification to workflow or other servers connected to the AXCP scheduler
- Run of multiple instance of a rule

Work to be done

- Realization of a planner based on Deadline monotonic algorithm and optimization by means Taboo Search
- GUI for the visualization of scheduled jobs will work with the planner and in real time.

- Definition of a internal communication model for providing the possibility to a rule to invoke other rules in the GRID
- Accessing to the executor for estimating the percentage of progress
- Defining a functional for estimating the execution cost and the percentage of progress
- Testing and solving possible bugs

3.2 AXMEDIS AXCP Rule Executor/Engine/GRID-Node (DSI)

The RuleExecutor is a console application embedding the GRID interface for network communication (peer) and the Spidermonkey JavaScript Engine extended with AxJSClasses that wrap AXMEDIS components and different Data Model. The following sections report the current status of the prototype under development and the list of functionalities that are available in the current version.

Major Features accessible:
The Rule executor consists of the following main components:

- **Grid Peer Interface** – the communication support with the AXCP rule scheduler
- **Rule Executor Manager** – the command interface to control the javascript engine
- **Script Executor/JSEngine** – The SpiderMonkey Javascript Engine (called JS Engine) extended with a set of JSClasses for AXMEDIS contents processing.

At now, the rule executor provides the following functionalities:

- Discovering: it responds when the Rule scheduler performs the discovering of peers on the network
- Sending profile: it generates the capabilities profile of peer machine and sends it to the rule scheduler
- Rule Receiving: it performs the download of a AXCP Rule from the rule scheduler
- Sending Messages: it sends different types of messages such as notification and errors,
- Commands parser for remote control messages

The **Rule Executor Manager** is the interface between the JSEngine and the Grid Peer Interface.The actual prototype of the Rule Executor Manager provides functionalities for:

- Routing messages produced by internal components to the scheduler via the Grid Peer Interface
- Receiving control messages and commands from the Scheduler via the Grid Peer Interface
- Parsing and executing commands coming from the scheduler such as:
  - Launching the execution of rule
  - Killing the execution of rule
  - Pausing the execution of rule
  - Resuming the execution of rule
  - Requesting profile
  - Requesting status
- File Transferring to:
  - send the profile of the Rule Executor
  - receive the rule to be executed
- Sending messages and notification to the Scheduler via the Grid Peer Interface
- Creating the profile of the executor according to the XML schema
- Managing the status of the Executor
- Starting the execution of the rule calling the internal rule launcher

The role of the Rule Launcher is to start the execution of the script. The main steps that the Launcher performs, are:

- Loading the rule XML file received by the Scheduler
- Extracting the script included in the Rule (all the information included in the *Definition* section of the XML file)
- Instantiating the Script Executo/JSEngine

- Calling the *Script Executor* for executing the script

The Script Executor receives the script code and arguments (Selections and parameters), then, it performs the necessary operations for:

- Invoking and initialising the JS Engine and variables.
- Sending the script to the JS Engine.
- Running and managing the communication with the JS Engine according to the capabilities and functionalities provided by the JS Engine.
- Routing errors coming from the JS Engine to the Rule Executor Manager.
- Sending Messages coming from the script in execution to the Rule Executor Manager.
- Generating output

In the initialization phase, the JS Engine is initialised according to the guideline of Spidermonkey for defineing the context, the runtime, the global variables, the definition of parameters of the rule.
The Engine Ouput for the JSEngine allows defining specialised display for output communication: GUI, network communication, etc.. It provides the way to:

- print internal error generated by the engine when compiling a script and at runtime during the execution/debugging
- print a message in the output interface
- visualize the line where the engine is stopped when a trap occurs
- clear the output display
- display the function name and the location (line of code) in the stack calls list
- reset the Stack call display
- reset the Local variables display
- print a variable in the Local variables display:
- print the properties of an object in the Local variables display

The Script Executor was developed to be used also in the AXCP Rule Editor in order to perform the debugging of rule.

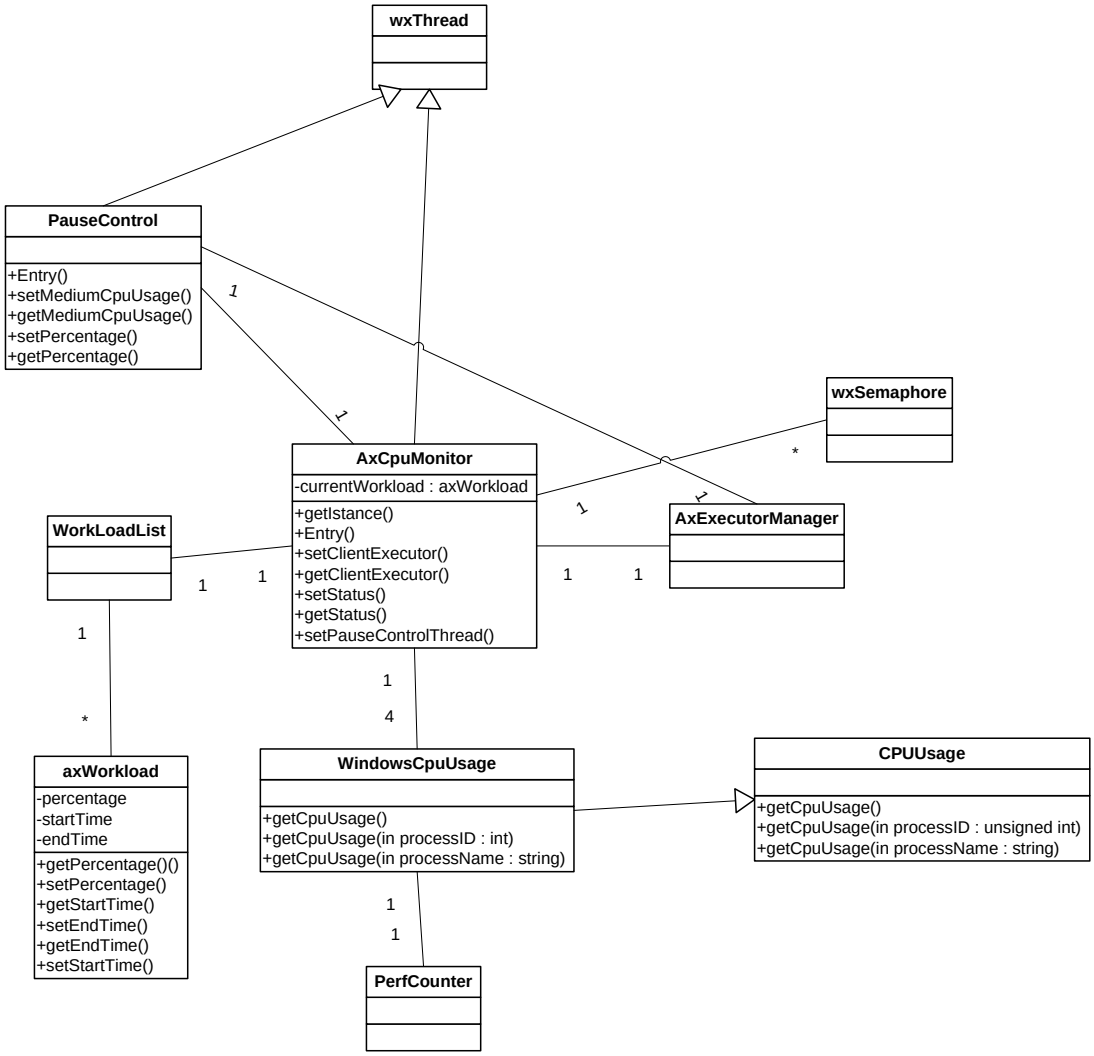
Debugging of rule: This modality was developed using the debug function API provided by JSDebug API of SpiderMonkey and to be controlled by the AXCP Rule Editor. The Spidermokey APIs permit to :

- put traps in the code corresponding to breakpoints (interrupting the execution)
- reporting of local variables
- manage the stack of functions
- realise the interface for debug functions and controls in the AXCP Rule Editor.

3.2.1 CPU monitor

CPU monitor is the module that allows controlling the workload of the executor and guarantee that the executor does not require all CPU time. The idea was to define a set of workload thresholds (the CPU percentage to use) associated with temporal intervals (Adaptive Threshold). The CPU monitor updates the workload threshold according the temporal profile. When the CPU percentage overcomes the current workload, the CPU monitor puts in pause the executor for the necessary time to reduce the load. After that the monitor resumes the executor until next overcoming.

In the following, the class diagram of the CPU monitor is reported.



AxCpuMonitor and *PauseControl* control di rilevare le condizioni nelle quali il thread di esecuzione (*AxExecutorManager*) deve essere messo in pausa e quelle in cui deve essere risvegliato. Si è scelta tale soluzione architetturale in quanto l'utilizzo dei performance counter come meccanismo per il controllo del carico di lavoro della cpu poteva portare a problemi di deadlock in occasione della sospensione dei thread. Con due diversi thread è stato invece possibile utilizzare i performance counter solo all'interno dell'*AxCpuMonitor* permettendo all'altro thread (*PauseControl*) di non subire deadlock e di effettuare il resume del thread messo in pausa. Si riporta ora una breve descrizione di ciascuna delle classi create.

AxCpuMonitor
It is a Singleton and contains methods and algorithms to estimate the current workload, the performance and to put in pause the *AxExecutorManager*

PauseControl
It monitors the *AxExecutorManager* and decides when it has to be paused or resumed.

AxWorkload e AxWorkloadList
These classes provide the data model for the workload information and temporal profile.

WindowsCpuUsage e CpuUsage

These classes are in charge to estimate the CPU percentage both total of the computer and relative to the Executor.

Control Algorithm
The control algorithm was implemented by estimating the error in a window of a configurable number of samples (N). The error was defined as

$$e = \Sigma_{i\ i}(Threshold - Current\ Workload)\ with\ i=-N,-N-1,\ldots 0$$

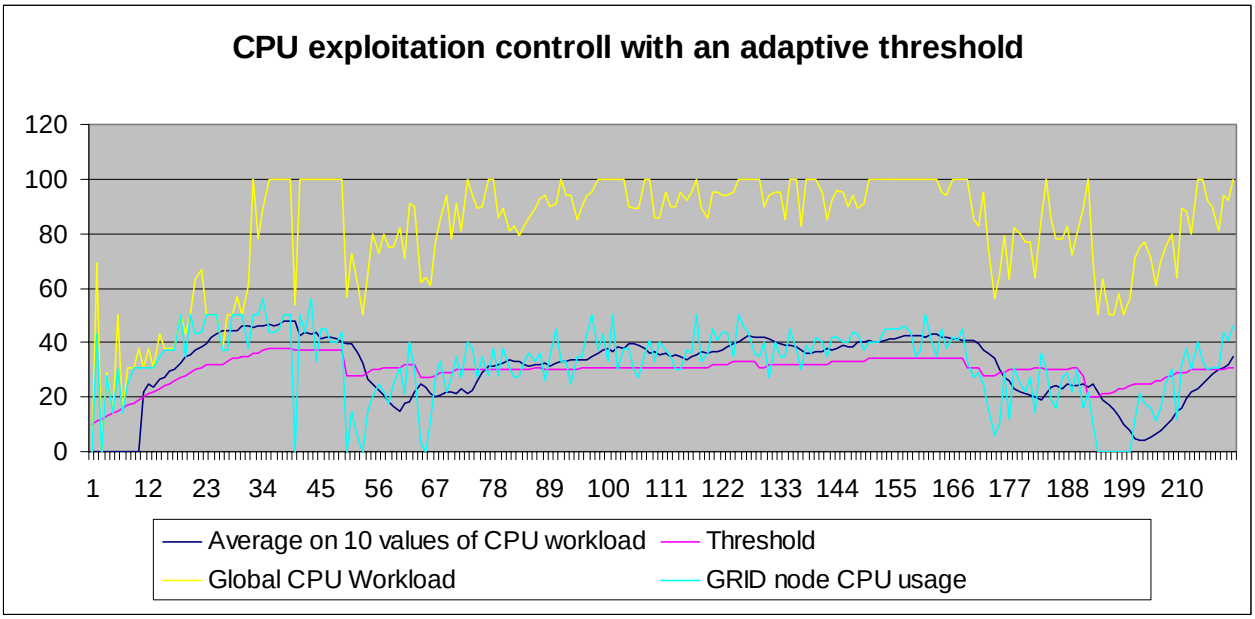
where:
Σ_i := sum of previous N samples

When N is equal to 1 the control is puntual.

List of configuration parameters

Parameter Name	Type	Description
ResumeTime_Granularity	Integer	Resuming period (in ms)
PauseTime_Granularity	Integer	Pausing period (in ms)
Cpu_Profile_File	string	Name of file with the report of CPU
Window_Lenght	Integer	Size of window
Percentage_Update	Integer	Percentage to use for incrementino threshold
Update_Percentage_Treshold	Integer	Threshold walue to use when comparing the current and desidered workload.
Mode	Mode	Define the modality of working: with or without threshold increment
Processors	Integer	Number of processor on the computer

In the following a graphic for CPU exploitation control with ad adaptive threshold is reported.



Workload description
The workload profile description allows defining a weekly profile for the workload. It is defined according to the configuration manager schema. An example is reported below:

```
<Module category="" id="WORKLOAD_SETTINGS">
```

```
            <Parameter name="MON"
type="string">0;0;0;0;0;0;30;50;30;30;20;30;45;50;20;30;45;50;30;30;0;0;0;0</Par
ameter>
            <Parameter name="TUE"
type="string">0;0;0;0;0;0;30;30;30;30;20;30;45;50;20;30;45;50;30;30;0;0;0;0</Par
ameter>
            <Parameter name="WED"
type="string">0;0;0;0;0;0;30;30;50;60;20;30;45;50;20;30;45;50;30;30;0;0;0;0</Par
ameter>
            <Parameter name="THU"
type="string">0;0;0;0;0;0;60;60;60;60;20;30;45;50;20;30;45;50;30;30;0;0;0;0</Par
ameter>
            <Parameter name="FRI"
type="string">0;0;0;0;0;0;50;50;50;50;20;30;45;50;20;30;45;50;30;30;0;0;0;0</Par
ameter>
            <Parameter name="SAT"
type="string">0;0;0;0;0;0;50;50;50;50;20;30;45;50;20;30;45;50;30;30;0;0;0;0</Par
ameter>
            <Parameter name="SUN"
type="string">0;0;0;0;0;0;50;50;30;70;20;30;45;50;20;30;45;50;30;30;0;0;0;0</Par
ameter>
      </Module>
```

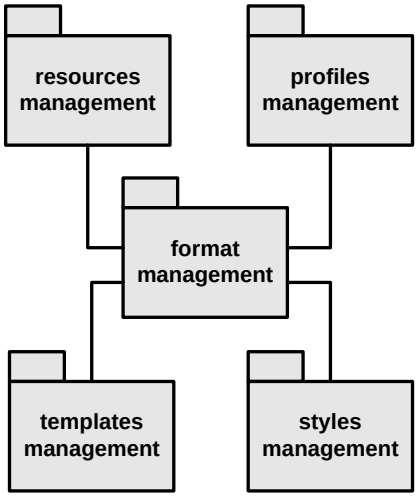
MON, TUE, WED, etc… are days of week and the associated string reports the profile of the day with the resolution of 1 hour.

Work to be done

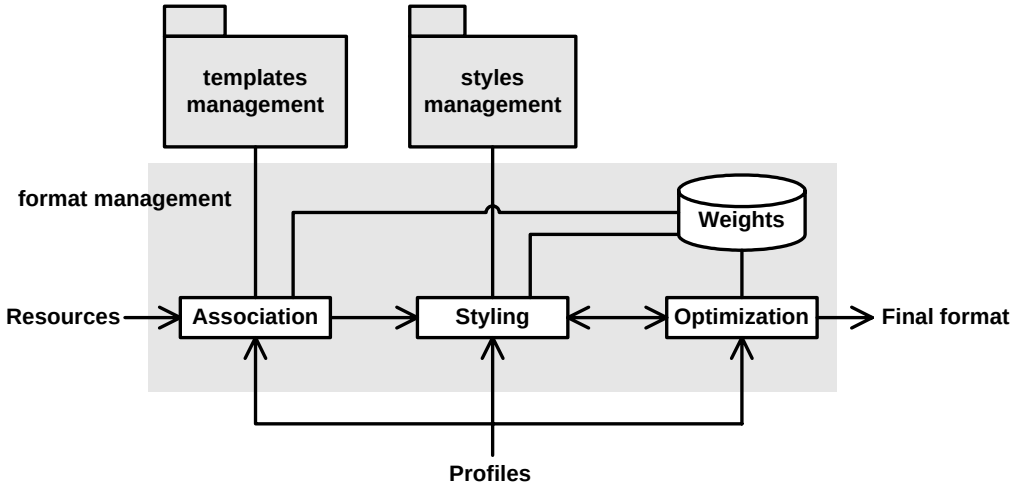
- Definition of a internal communication model for providing the possibility to a rule to invoke other rules in the GRID
- Test of the CPU monitor with complex rule and on different machine
- Definition of a model for estimating the progress of the execution in terms of percentage of wrok done
- Test of the communication protocols with several executors
- Check Mode - This modality will be mainly used by AXCP Rule Editor when it will be necessary to check the feasibility of a rule. In the check mode, the rule will be executed in order to:
    - verify the correctness of the rule before to send it to the AXCP Rule Engine
    - estimate some parameters related to the complexity of the rule. Such parameters will be identified and defined during the project life. They will be used to define a complete profile of the rule in terms of required computational resources.

3.3 AXMEDIS Content formatting (DSI: vaccari)

The Content Format Engine provides formatting functionalites that are used by the AXCP GRID and by format tools (*Template Editor and Selector*, *Style Editor and Selector*, *Style Optimizer*) integrated within the AXMEDIS SMIL Editor.
The goal of the Content Format Engine is integrating digital resources, contained within an AXMEDIS Object, in a multimedia presentation (based on the SMIL language) suitable for the final user. The Engine is based on templates and style-sheets that define respectively the basic structure and the look-and-feel of the presentation. To get best results in terms of adaptation to customer's needs, the Content Format Engine performs its choices according to profiles that describe personal preferences, user's device capabilities and delivery context.
The Content Format Engine Module is composed of five main subsystems:



Resources and profiles (user, device and context profiles) represent the input to the Format Engine; selected templates and styles are its output and they are used to create the SMIL presentation; the format management implements the filtering and optimization logic.
The format management subsystem is logically divided into three blocks:



The Association block manages the template library; templates are written using a subset of the SMIL language and integrate resources in a structured multimedia presentation.
The Styling block manages the style library; style-sheets are written in XSLT and tranform the SMIL template in a complete SMIL presentation including a precise look-and-feel description. Style-sheets may also operate complex transformations to adapt the output to specific needs.
The Optimization block tries to determine values for the parameters defined in the style-sheet to fit the layout of the presentation to the included resources and to the delivery context. This goal is achieved using Genetic Algorithms for exploring the space of solutions and evaluating the best ones.

Major Features accessible:
At this moment, the major features implemented by the C++ Content Format Engine are:

- management of templates, that is:
  - adding new items to the template library, thus allowing AXCP to use them in the future. To do this is in an efficient way, the template is processed and an XML descriptor containing its outstanding properties is created in a local directory;

    - browsing the template descriptors, getting information about each template;
- management of style-sheets, that is:
    - adding new items to the style-sheet library, thus allowing AXCP to use them in the future. To do this in an efficient way, the style-sheet is processed and an XML descriptor containing its outstanding properties is created in a local directory;
    - browsing the style-sheet descriptors, getting information about each style-sheet;
- templates and style-sheets may be created using an existing SMIL document (created with an external editor);
- template and style-sheet selection: on the basis of descriptors, the system selects the best template and style-sheet for the given resource set using some imposable criteria, that are mainly filtering and scores;
- SMIL creation: using template and style-sheet, the system creates a SMIL file that integrates the resources;
- all main functionalities are already available in JS scripts, that additionally allows the formatting of resources contained into AXMEDIS objects;
- Genetic Algorithms have been used to implement a prototype for estimating the optimized/compromised values for some parameters of the style-sheet (presently size and position of the spatial region defined in the SMIL presentation).

Work to be done:
The development of this module will be mainly focused on the following tasks:

- creating a GUI for selecting and previewing templates and style-sheets to give the user more control on the formatting process;
- adding in the AXMEDIS SMIL editor some new functionalities to create and edit templates and style-sheets;
- improving and testing the optimizer and integrating it in the Content Format Engine;
- improving the templates and style-sheets selection logic, adding new criteria for filtering and refining the existing ones;
- improving the use of profiles in JS, exploiting functionalities that will be offered by JSDeviceProfile, JSUserProfile and JSNetworkProfile.

3.4 JS Classes Status (DSI plus ALL)

The composition and formatting process by means javascript required the extension of the object data type inside the javascript language. For this reason the following set of JSClasses was defined:

- JS_AXOM
  - JSAxBaseObject Class base class for *JSAxObject*, *JSAxDublinCore, JSAxMetadata, JSAxInfo* and *JSAxResource* classes. It allows linking them to the current instance of the Axmedis Object Manager (*theAxom*) and accessing to its methods and commands. It allows also to associate a reference to *theIndex* (*AxIndex*) with the current instance of a child class when it is inserted into the Axmedis Object Data Model
  - JSAxResource Class models the AxResource type of the Axmedis Framework, it is used to store digital resource (audio, video, image, text, etc…) both raw data and Axmedis digital component.
  - JSAxmedisObject Class mapping of an AXMEDIS Object for JavaScript. According to the specification of the AXMEDIS OBJECT MANAGER and Axmedis Data Model
  - JSAxInfo Class maps and allows managing the metadata of the AXINFO in the JavaScript. This class manages the access to individual elements and fields in AXINFO metadata, this class map all the functionalities provided by AxInfo class exposing setter and getter methods for accessing to data
  - JSAxDublinCore Class models the AxDublinCore type of the Axmedis Framework, it is used to store metadata in Dublin Core format.
  - JSAxMetadata Class models the AxMetadata type of the Axmedis Framework, it is used to store metadata in XML format
- JSAXCPPlugin a metaclass that refers and wraps dynamically AXMEDIS Content Processing Plugins. It provides and manages the access to the set of functions exposed by DLLs discovered by the Plugin Manager and related to AXMEDIS content production, protection, adaptation algorithms and tools
- JSConnection
  - JShttpConnection The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
  - JSftpConnection The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
  - JSodbcConnection The class models the http connection by providing primitive methods for accessing and retrieving information by means the http protocol
  - JSwebServiceConnection The class models a meta class for managing web services by loading WSDL description and dynamically creating services (methods).
  - JSForm The class models a simple form of a http web page
- JSZipArchiver models Javascript class for creating archives of files based on the ZIP algorithm
- JSAxSelection class maps the Selection document in the JavaScript. This class allows using Selection objects to manage the access and to make queries to the AXMEDIS database, and to retrieve AXMEDIS objects ID (AXOID). It manages the array of AXOID provided by the MainQuerySupport when the selection is resolved to be actualized.
- AXSBJS - AXMEDIS searchbox javascript bridge JS_CRAWLERDB ACCESS (DSI WITH SUBCONTRACT) (see DE4.2.1)
  - AXSearchbox: The main object for accessing searchbox.
  - Document: Used to handle documents as opaque objects.
  - MetadataValue: Used to hold metadata.
  - QueryParser: Used to specify the query string parser to use.
  - QueryInfo: Used to specify the information returned as result of a query.
  - QueryView: Used to restrict the set of documents returned as result of a query.
  - QueryAtomType: Used to specify the type of a QueryAtom.
  - QueryAtom: Used to build a query in RPN notation.

  - QuerySliceWeight: Used to specify slice weights.
  - QuerySpec: Used to submit a query.
  - QueryResult: Used to return information on a query result
- JS_PROTECTION (FHGIGD, see DE4.5.1)
  - JS_ProtectionInfo The protection information holds the IPMP information as stated in MPEG-21 Part 4 IPMP standard and more
  - JS_ProtectionStamp To identify the correct ProtectionInfo for an AXMEDIS object also the so-called ProtectionStamp is needed which binds an object to the different protection models that can be applied to the object.
- JS_DRM (FHGIGD, see DE4.5.1)
  - JS_License – class that models a License
  - JS_PAR – class that models a PAR
- JS for P2P connection:
  - JSDownloader class wraps the class that controls the download of AXMEDIS objects form the P2P environment
  - The JSLoader Javascript class wraps the class for loading/trasnferring AXMEDIS objects from the AXDB In Database to the Factory AXMEDIS Database
  - JSPublisher Javascript class wraps the class for loading/trasnferring AXMEDIS objects from the Factory AXMEDIS Database to the AXDB Out Database and/or to the P2P internal DB.
- JSFunctions a set of auxiliary functions wrappred into Javascript. They were divided into the following categories:
  - File system functions – File exists, dir exists, create dir, etc…
  - Mime type: conversion from extension to mimetype and viceversa
  - Process: execute external application
  - I/O: print
- JSUserProfile captures and stores information related to the users, the usage environment and the user preferences. This profile should be managed at the personalization server and be able to store minimal details about the user to identify his personal preferences in respect to the services being offered. The elements in this profile are the minimal set of elements that is required to meet the AXMEDIS User's personalization needs. Each of these elements are based on the definitions provided by MPEG21 schema [ISO MPEG-21, Part 7 - Digital Item Adaptation, ISO/IEC JTC1/SC29/WG11/N5231, (Oct 2002)]. The schema for this profile is contained in axmedis-user-profile.xsd The JSUserProfile is a JS class to automatically create and manage user profiles as a part of rule scripts for AXCP engine
- JSDeviceProfile Device profile captures information related to user's device, that the service provider must take into account while delivering the contents. Generally the minimal set of elements required are those that affect the proper utilization of the contents on the device e.g. screen resolution for images. This profile represents the characteristic of the user's personal device and hence contains most vital information for the content providers that can be used to transcode the content to match the device characteristic and can be better utilized by the user. Each of these elements are based on the definitions provided by MPEG21 schema [ISO MPEG-21, Part 7 - Digital Item Adaptation, ISO/IEC JTC1/SC29/WG11/N5231, (Oct 2002)]. The schema for this profile is contained in axmedis-device-profile.xsd
- JSNetworkProfile captures information related to the carrier that is used for delivery of the content to the user's device. It is necessary for the distributors to know various characteristics of the Network so as to provide the promised QoS. The information contained in this profile can also be used to for transcoding of the contents to better utilize both network and device capabilities. The elements in this profile are the minimal set of elements that is required to meet the AXMEDIS User's personalization needs and to meet the distributor's needs in terms of the network capabilities. Each of these elements are based on the definitions provided by MPEG21 schema [ISO MPEG-21, Part 7 - Digital Item Adaptation, ISO/IEC JTC1/SC29/WG11/N5231, (Oct 2002)]. The schema for this profile is contained in axmedis-network-profile.xsd.
- JS_FORMATTING (Style and Optimisation)

  - JS_format wraps the functionalities offered by the C++ Content Format Engine, allowing the automatic selection of templates and style-sheets and the creation of a SMIL presentation that integrates a set of resources. Resources can also be contained in an AXMEDIS object and the resulting SMIL can be embedded into the object.
  - JS_template allows to add new templates to the library used by the Content Format Engine and to browse them.
  - JS_style allows to add new style-sheets to the library used by the Content Format Engine and to browse them.
- JS_MetadataMapper wrapper around the MetadataMapper C++ library. It provides the following functionalities: Instantiating a MetadataMapper object within a JavaScript, Loading an XSLT stylesheet, Transforming XML from source to destination language according to the stylesheet

Some of them are at prototype level, the other will be implemented during the life of the project since most of these depend from AXMEDIS components and modules.

3.5 Content Composition, examples (DSI)

The Compositional Process aims to discover and specify the relationships (semantic, logical or spatial) between the resources returned from the Selection Process, and to create new digital items (AXMEDIS objects) that include the resources and their metadata and relationships.
Different types of relationships are possible:

- spatial: resources belong to the same area of the layout (e.g.: header or footer elements);
- logical: resources are linked together (e.g.: the image associated with a menu item and the resourced related to the linked item);
- semantic: resources are connected by their "meaning" within the presentation (e.g.: a painting and its description).

Relationships could be defined through the hierarchical structure of the AXMEDIS object (which may include other AXMEDIS objects), or specified within metadata.
Relationships may be obtained from the author (who could define them explicitly) or from the Selection Process. In the latter case, the queries executed on the database could be used to get informations about the relationships between the resources that were returned.

In this section two examples of script for content processing are reported.

3.5.1 Composition process, example

In the following an example of composition script. A new object is created by adding an image resource and defining some metadata.

```
var obj = new AxmedisObject();

 var info = new AxInfo();
 resource = new AxResource();
 var obj_metadata =  obj.getMetadata();

 // definition of a resource, the resourcePath variable defines where the resource is located
 resource.mimetype="image/gif";
 resource.ref=resourcePath;
 resource.contentID="gif";
// mCon is the reference to the resource embedded in the Axmedis object "obj"
 mCon = obj.addContent(resource);


// Adding an empty AxInfo inside the Axmedis Object
 var mInfo = obj.addMetadata(info);

 mInfo.isProtected = true;
 mInfo.setOwnerID("abcdefghi", "Ivan");
 mInfo.ownerNationality = "Italy";

//Setting the Date of Creation
 var e = new Date();
 mInfo.objectCreationDate = e.getDate()+"."+(e.getMonth()+1)+"."+ e.getFullYear();

 obj.save("c:\\axobj.xml");
```

3.5.2 Content Processing, example

In this portion of javascript code the conversion/adaptation of digital resources for mobile is reported.

```
   // Loading Axmedis Object via URI
    var b = new AxmedisObject(URI);
//Retrieving the array of Resources inside the object
    var resource = b.getResource();

var obj = new AxmedisObject();

// For each resource the following lines perfom the coversion for the mobile specified in the "profile"
// parameter and put the newResource in the empty Axmedis object "obj"
for( res in resource)
{
          newResource = AdaptForMobile(resource[res], mobileProfile)
            mCon = obj.addContent(newResource);
  // mCon is the reference to the resource embedded in the Axmedis object "obj"
}

[…..]
```

3.6 Content formatting, examples (DSI)

3.6.1 Loading of resources contained in an AXMEDIS object (DSI)

This example shows how the resources contained in an AXMEDIS object may be loaded into a FormatManager. The loadAxFile() function adds to a ResourceList the resources contained into the Axmedis object stored in the axFile file.

```
function loadAxFile(axFile, rList)
{
   // opening file with axObject(s)
   var obj = new AxmedisObject(axFile);
   // loading resources contained in the root element
   // and starting iteration over the children
   loadResources(obj, rList);
   return true;
}

/*
 * Load in rList the resources contained into
 * the Axmedis object axObj and (recursively)
 * in its childs.
 */
function loadResources(axObj, rList)
{
   // exploring object's hierarchy
   var children = axObj.getContent();
   var childrenCount = axObj.childrenCount;
   var resources = new Array();
   var resCount = 0;
   var objects = new Array();
   var objCount = 0;
   var child;
```

```
   for (var i = 0; i < childrenCount; i++)
   {
      child = children[i];
      if (child instanceof AxResource)
      {
         resources[resCount] = child;
         resCount++;
      }
      else // child is instanceof AxmedisObject
      {
         objects[objCount] = child;
         objCount++;
      }
   }
   print(" - object id: " + axObj.AXOID);

   // adding resources
   for (i = 0; i < resCount; i++)
   {
      var type = resources[i].mimeType;
      var path = resources[i].localPath;
      var enc = resources[i].encoding;
      var id = resources[i].contentID;
      var ax = axObj.AXOID;
      rList.addResource(type, path, "", axObj.AXOID, "", "some_metadata");
   }
   // iterating over inner objects
   for (i = 0; i < objCount; i++)
   {
      loadResources(objects[i], rList);
   }

}
```

3.6.2 Formatting of an AXMEDIS object (DSI)

The following is a working example of formatting of an AXMEDIS object. It uses the functions defined above.

```
FormatManager.prototype.loadAxFile = loadAxFile;
var fm = new FormatManager();

var axFile = "axpresentation.axm";
var templPath = "../../../../Framework/doc/test/contentformatter/files/templateDescriptor";
var stylePath = "../../../../Framework/doc/test/contentformatter/files/styleDescriptor";
var devFile = "../../../../Framework/doc/test/contentformatter/files/profiles/device1.xml";
var outFile = "output.axm";

formatObject(fm, templPath, stylePath, devFile, axFile, outFile);

/*
 * formatObject()
 */
```

```
function formatObject(fm, templPath, stylePath, devFile, axFile, outFile)
{
   // load templates
   if (fm.loadTemplateDescriptors(templPath))
      print(" - templates loaded: " + fm.templateNumber());
   else
   {
      print("Error while loading templates.");
      return false;
   }

   // load style-sheets
   if (fm.loadStyleDescriptors(stylePath))
       print(" - styles loaded: " + fm.styleNumber());
   else
   {
      print("Error while loading styles.");
      return false;
   }

   // load profiles
   if (fm.loadDeviceInfo(devFile))
      print(" - device profile loaded");
   else
   {
      print("Error while loading profiles.");
      return false;
   }

   // load resources
   if (fm.loadAxFile(axFile, fm.getResourceList()))
      print(" - resources loaded: " + fm.resourceNumber());
   else
   {
      print("Error while loading resources.");
      return false;
   }

   // filtering
   var tplId = fm.getBestTemplateId();
   print (" - selected template is: " + tplId);
   var stlId = fm.getBestStyleId(tplId);
   print (" - selected style-sheet is: " + stlId);

   // create SMIL file
   if (!fm.saveSMIL(tplId, stlId, "output.smi"))
   {
      print ("Error while creating output file.");
      return false;
   }

   // include SMIL file into the object
   var obj = new AxmedisObject(axFile);
   res = new AxResource();
```

```
    res.load("output.smi");
    res.mimeType = "application/smil";
    res.contentID = "output.smi";
    res.localPath = "output.smi";
    obj.addContent(res);
    obj.save(outFile);
    return true;
}
```

3.7 Profiling, examples (IRC)

3.7.1 User profile

Sample Java Script code to get the value of a user-profile element and set it with user specified value.

```
var userprofile= new UserProfile();  //  Initialises the User Profile
userprofile.loadXmlFile("C:\axmedis\testProfile.xml"); // Loads an xml file into the profile class

// Get the volume preference of the user
var volume= userprofile.getValue("Preferences/AudioPreferences /VolumeControl");

//set the volume preference of the user
userprofile.setValue("Preferences/AudioPreferences /VolumeControl","0.5");

//Close the file
userprofile.close();
```

3.7.2 Device profile

Sample code to demonstrate copying one device profile into another file is given below. This example is mainly to demonstrate the methods used for retrieving the xml contents from a file and writing it to another file.

```
var deviceprofile1= new DeviceProfile();  //  Initialises the Device Profile for the source xml file
var deviceprofile2= new DeviceProfile();  //  Initialises the Device Profile for the destination  xml file

// Loads an source xml file
deviceprofile1.loadXmlFile("C:\axmedis\sourceProfile.xml");
// Retrieves the xml-string from the xml file and stores it into the variable xml
var xml=deviceprofile1.getXml();


//Creates the destination xml file
deviceprofile2.createXmlfile("C:\axmedis\destProfile.xml")
// Sets the contents for the destination xml file
deviceprofile2.setXml(xml)

// Closing both the xml files
deviceprofile1.close();
deviceprofile2.close();
```

3.7.3 Network profile

Sample Java Script code to get the value of a Network -profile attribute and set it with user specified value.

```
var networkprofile= new Network Profile();  //  Initialises the Network Profile
networkprofile.loadXmlFile("C:\axmedis\testProfile.xml"); // Loads an xml file into the profile class

// Get the yRadius
var yrad= networkprofile.getAttribute ("Mobility/UpdateInterval", "yRadius");

//set the yRadius
networkprofile.setAttribute("Mobility/UpdateInterval", "yRadius",0.5");

//Close the file
networkprofile.close();
```

4 Content Composition and Formatting (DSI)

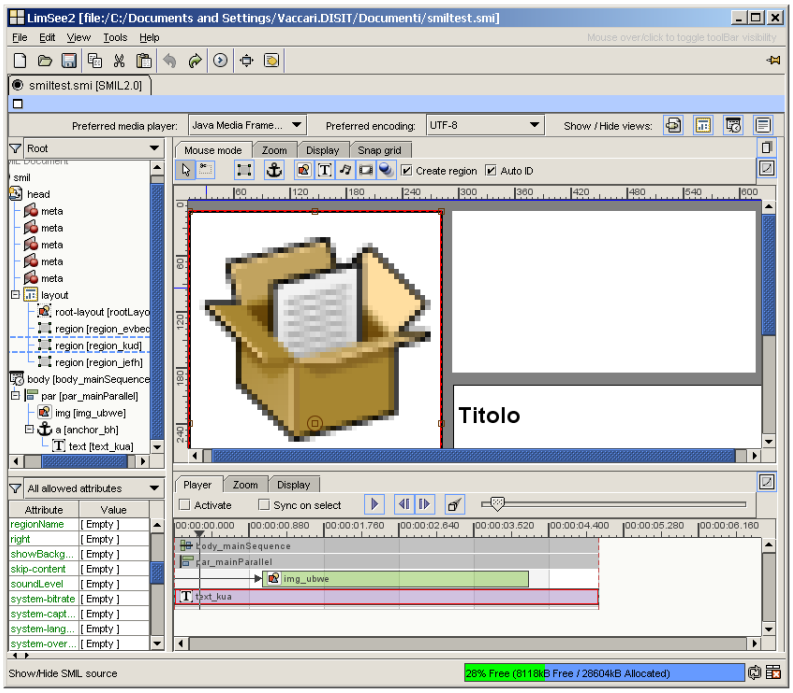
4.1 State of the art

4.1.1 Multimedia presentations

As the number of the devices which can exploit the potentialities of multimedia is more and more increasing, a great variety of approaches for the description and the adaptation of the multimedia presentations has been proposed in the last years.

Some of them are based on the graph grammars: the graph structure is governed by rules that guide the automatic adaptation of the presentation to satisfy input and output constraints. Furthermore, the graph gives the author a visual representation of the components in the presentation and their logical positioning. Relational Grammars [Wei94] and others grammatical specifications has been proposed to describe the structure of multimedia presentations; the graphs that express the document structure may also be represented graphically and managed with visual tools [Zha05]. Reserved Graph Grammars (RGG) have been used for the transformation of XML documents that define multimedia structures [Zha02]. Between the main lacks of these approaches we remember the absence of the time dimension.

Several document models have been developed specifically to describe multimedia formats. The SGML-based HyTime language has been associated with DSSSL (Document Style Semantics and Specification Language, a transformation language for SGML) to produce adaptable presentations [Rut98]. ZYX [Bol01] is a sophisticated model created to provide a great support for reuse and adaptation: it supports both static and dynamic adaptation, and introduces the concept of "augmentation", a semi-automatic selection of alternative contents which performs a "cross-media adaptation" [Bol99]. Madeus [Jou98] is a constraint-based authoring environment with temporal and spatial specifications: the timing of elements is calculated from the constraints specified among them, rather than as absolute position on the timeline. Madeus, unfortunately, doesn't offer any support for adaptation than constraints specification.



LimSee2

The models above have been used in the implementation of different framework for multimedia authoring. MM4U [Bol03], starting from the ZYX model, propose a generic framework for creating personalized multimedia presentations, staying independent from the underlying data model and the adopted compositional algorithm. With the same objective of realizing a generic system for the automatic cross-platform adaptation, Cuypers [Van01] exploits the semantic description of the document offered by the Rethorical Structures. LimSee and LimSee2 [Roi03] are SMIL editors that rely on the Madeus model and offer a multi-view solution to render the structure of the SMIL document at different levels during the authoring process; the timeline is the main tool for synchronization in LimSee2, and supports both direct and constraint-based temporal scheduling.

Recently, SMIL [W3C05] – an XML-based language for declarative definition of multimedia presentations – is considered the most significant standard in this field. The strength of SMIL relies on two main features: a strong definition of the temporal relations between the elements of the multimedia presentation [Har99], and an automatic user-level adaptation of the content [Bul98].

SMIL has become very popular for the description of structured presentations, and many extensions for its model have been proposed: for instance, a finer granularity in the references to the media and a more generic expression of duration properties may enrich the language [Thu02].
Moreover, the capabilities of adaptation offered by SMIL may be enhanced from the association with XML transformation languages: among them, the XSL has often been preferred. The use of XSL style-sheets may transfer a part of the adaptation work, normally requested to the SMIL clients, on the server-side [Pih03]; furthermore, a double transformation system has been proposed to adapt both the document source and its style-sheet [Lem03]. A refined adaptivity may be obtained introducing additional constraints that have to be solved after the style-sheet application [Bes01].

4.1.2 Template language

The choice of template language is the first step in the system definition. Templates have to describe correctly multimedia layout and synchronization (including user interaction), define media adaptation (if needed), be adaptable at different user profiles, be suitable for use of style-sheets; thus the choice of the description language is a delicate issue.
Many advanced systems for multimedia authoring (e.g.: MM4U [Bol03], GRiNS [Bul98], LimSee2 [Roi03]) use their own internal model to describe the structure of the document; this way they get the maximum degree of expressivity and flexibility. On the other hand, standard languages (e.g.: SMIL, HyTime) may have some lacks or limitations, but they allow using off-the-shelf software and they don't require long definition and verification activities. Last, the chosen language has to provide an easy way to convert the description into required output formats (MPEG, HTML, SMIL, etc.).
Foremost standard languages for describing multimedia are:

- HyTime (ISO/IEC 10744) is an SGML-based standard that seeks to provide ways of addressing and linking to any kind of information, anywhere in time and space. Very hard to learn, it has been substantially replaced by SMIL;
- MPEG-4 (ISO/IEC 14496) provides a standardized representation of interactive multimedia content in terms of media objects, hierarchically organized within a scene. The XMT (Extensible MPEG-4 Textual) format (and specifically the XMT-O) provides interoperability between MPEG-4 and SMIL;
- MPEG-7 is a standard (ISO/IEC 15938) for describing the multimedia content data that supports some degree of interpretation of the information meaning. It includes a Description Definition Language (DDL) based on XML Schema language (XSD);
- SMIL (Synchronized Multimedia Integration Language) is an open XML-based language proposed by the World Wide Web Consortium [W3C05]. SMIL allows to create a well structured document, with the focus at final result and not at rendering mechanism. As stated by its name, SMIL is an "integration language": media files (images, audio, video, animations, text) are only referenced, but

they stay independent from the SMIL file. SMIL has been designed to be used over Internet and may be integrated with others W3C standards: XML, XSL, CSS, XHTML, SVG, etc.

SMIL project has been started by W3C in 1995; its first version (SMIL 1.0) has been released in 1998, the second (SMIL 2.0) in January 2005. Currently (August 2005) the last version is the "Candidate Recommendation" for SMIL 2.1.
SMIL 2.0 is composed by 45 modules divided in 10 groups:

- timing (19 modules);
- time manipulations (1);
- animation (2);
- content control (4);
- layout (4);
- linking (3);
- media objects (7);
- metainformation (1);
- structure (1);
- transitions (3).

Modules are designed to be reusable as parts of other XML vocabularies, so vendors or other standards initiatives may decide to implement only parts of SMIL. On the other hand, it is possible enhance SMIL with proprietary extensions: for instance, Apple QuickTime provides extensions to play automatically the presentation, to show a time slider, to allow full screen playback, to get informations about the bandwidth a media object needs in order to play back in real time.

SMIL has some interesting features that make it our best choice as description language for multimedia documents:

- human-readable format;
- "open" standard (in opposition to proprietary standards);
- designed for the web (in opposition to media-oriented MPEG);
- separation between spatial relationships (layout) and time relationships (synchronization);
- CSS support;
- independent multiple windows support (via the <topLayout> element);
- switching between different layout types and media formats depending on the client or the user preferences (via the <switch> element);
- synchronization with absolute time, relative time or asynchronous events;
- support of complex transitions between media;
- metadata support;
- extensibility (proprietary extensions allow adding of specific attributes to SMIL presentations);
- interaction with the user.

For the reasons above, we decided to use SMIL language for definition of structure and synchronization. Moreover, it is important to remark that a SMIL editor will be provided within the AXMEDIS framework: this is a plus for the adoption of SMIL in the formatting system.
Templates will describe synchronization in detail, but layout properties may be very generic (i.e.: "auto" and "indefinite" values will be used often); spatial relationships will be defined with precision in the style-sheets. This way our template is "real SMIL" also if visual rendering may be inconsistent; more important, there's

no formal difference between a template and a styled presentation: thus style-sheets may be applied indifferently to templates and to already styled presentations, enhancing possibilities of reuse.
To make more general the document description, we could also define some extensions for the SMIL language used in templates. For instance, could be defined some tags to describe homogeneous groups of elements, without regard for their number, making templates more reusable.
Ideally, every graphical SMIL editor could be used as Templates Editor. Anyway, it must be pointed that, in the SMIL language, media may be synchronized using an absolute timing (the presentation time), a relative ordering, or a set of constraints based on the temporal properties of other media; therefore, to manage such temporal and behavioral properties, the editor has to be very refined. It would also be good if the editor could check the syntax against an arbitrary DTD, to allow modifications and extensions to SMIL.

4.1.3 Style-sheet language

The main goal of style-sheets systems is separation between document content and its visual representation. The most widely diffused style-sheets standards are CSS and XSL, both defined by the W3Consortium.
Cascading Style Sheets (CSS) is a simple mechanism for adding style to XML documents. A CSS engine visits each node of the XML document node hierarchy and tries to match this node to a CSS rule. Therefore, CSS cannot be used to modify the XML document structure and the rendered document is strictly dependent on the XML document structure. Nevertheless CSS is human readable, very easy to learn, efficient.
XSL (eXtensible Stylesheet Language) is the more advanced style-sheet language for XML. It is made of three parts:

1. XSL transformations (XSLT), used to transform XML documents;
2. XML Path Language (XPath), used by XSLT to refer to parts of an XML document;
3. XSL Formatting Objects (XSL FO), used to describe formatting semantics.

The main advantage of XSL over CSS is its capability of transform the XML structure. Price to pay is a more difficult syntax and a less immediate readability. Furthermore, for its tree-shaped nature and its dependence from the XML source, creating visual tools for XSL editing it's not a simple task: market offers several IDE for XSL editing, but rarely they offer sufficient flexibility, since they are targeted to a particular type of output (generally HTML).
Anyway, XSL is the more advanced, powerful and thus preferred language for styling XML (and so SMIL); and CSS, eventually, may still be used in association with it.
In our system, the XSL processor receives as inputs an XSLT style-sheet and a template written in SMIL; its output is a styled SMIL document. XSLT style-sheets may radically change the final aspect of the presentation, and a style-sheet may be applied to an already styled document with no problem.
The XSLT style-sheet will be divided in two parts: a generic redefinition of the SMIL format structure (valid for every style-sheet) and a style-sheet specific layout description. This way, style-sheet creation will be simpler.
The Style-sheets Editor will be a visual XSLT editor: it operates on the entities included in the template, managing their spatial properties and their temporal parameters.

4.1.4 Optimization algorithm

The optimization of the style-sheet is the process that aims to get the best looking layout for the document, according certain criteria and the range for values of parameters. The number of parameters is potentially very high, and their values may have a large range: this makes the optimization a critical step in the automated creation of the document.
The optimization problem may be defined as follows. Many parameters allow to tune the style-sheet, and each of them may assume a given range of values; every combination of parameters generates a possible layout that has to be evaluated according some general criteria, based on aesthetic or functional considerations. For example: a cost functional could evaluate results in terms of proportions, visibility and use of the screen offered by the considered layout, adopting a different weight for each criteria.

The optimization algorithm should provide the optimal solution, that is the best layout for the document. Actually, the complexity of the problem is very high (it may be considered NP-complete), thus meta-heuristic algorithms, as the Genetic Algorithms, could be used profitably.

GA [Gol89] are meta-heuristic algorithms based on the evolutionary theory. A population of candidate solutions evolves toward better solutions; the evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (mutated or recombined) to form a new population. The new population is then used in the next iteration of the algorithm.
GA have often been used for formatting problems or in other similar cases.

The two-dimensional bin-packing is an optimization problem that consists in arranging some shapes on a rectangular canvas [Pon05]; thus it can be considered similar to the problem of laying out the regions of a presentation into the player's window. It can be defined as follows [Goo94]:

given a finite set of rectangular boxes $E=\{e_1, \ldots, e_n\}$ *with associated sizes* $W=\{(x_1, y_1) \ldots (x_n, y_n)\}$ *such that* $0<x_i, y_i <L^*$, *place without overlapping all or some of the boxes from E into the rectangular bin with sizes* $X>L^*, Y>L^*$ *such that the relation of the sum of box areas to the bin area is the maximum.*

GA have also been used for determining the layout of a digital photo album [Gei03]. Given a set of images for an album, a page-layout algorithm distributes the images among a set of pages and lays out the images on each individual page, requiring minimal user input. If the desired layout is more similar to a scrapbook than to a simple table, GA are appropriate for such an "artistic" task because they doesn't attempt to mimic or model any particular process to create solutions. Instead, they generate solutions randomly and evaluate them after the fact.

Other examples of application of the GA are the optimization of the advertisement layout on a newspaper [Ric97] and the optimization of the layout of a web page [Oli99].

4.2 Work done

4.2.1 Template and style-sheet selection logic

This task is extensively described in section 3.3 and examples of its use through JS scripts are reported in section 3.6.

4.2.2 Strategy for document adaptation

A single multimedia document (described by its template) has to be formatted in different ways and its components have to be adapted to different users that access the document over different channels.
Complexity degree introduced by the number of different channels, users platforms and users preferences, may be very high. Therefore, the automatic formatting system has to offer an easy but powerful way to manage this complexity.
We provide these functionalities at three levels:

1. at template selection level: only templates that match with context (and input) have to be selected. For this reason each template should include a list of output options and context informations. Context informations will be stored in the Templates Database. For output options, could be used the SMIL <switch> statement: a single SMIL template may adapt its structure to cope with several targets. In this way we can change layout and synchronization and even disable some features to get the best results on each client. The <switch> statement may also drive the input adaptation: depending on the targeted channel, the template chooses the right format for the input media. The style-sheet will process only the appropriate <switch> cases and it will remove the whole <switch> statement.

   The <switch> statement, defined in the SMIL 2.0 Basic Content Control Module, expresses that a set of document parts are alternatives, and the first one fulfilling certain conditions should be chosen. One or

more test attributes may appear on media object references or timing structure elements; the system test attributes are:

- systemBitrate: allows a choice based on the user connection to the network;
- systemCaptions: specifies a redundant text equivalent of the audio portion of the presentation;
- systemLanguage: allows a choice based on the languages indicated by user preferences;
- systemRequired: provides an extension mechanism for new elements or attributes;
- systemScreenDepth: specifies the depth of the screen color palette in bits;
- systemScreenSize: specifies if the playback engine is capable of displaying a presentation of the given size;
- systemAudioDesc: specifies whether or not closed audio descriptions should be rendered;
- systemCPU: specifies the CPU on which a user agent may be running;
- systemComponent: identifies the components of the playback system (e.g.: user agent component/feature, number of audio channels, codec, HW MPEG decoder, etc.);
- systemOperatingSystem: specifies the operating system on which a user agent may be running;
- systemOverdubOrSubtitle: specifies whether subtitles or overdub is rendered;

2. at style-sheet selection level: several style-sheets may be provided for each template, to get the maximum flexibility. In the Style-sheets Database every style-sheet has a descriptor that specifies output and context informations. Moreover, every style-sheet may be parameterized, and the value of its parameters will be chosen according an optimization algorithm. Finally, templates may be applied in cascade: the application of a style-sheet to a previously adapted document is very useful in terms of adaptability and reuse;
3. at user level (eventually): if the output is in SMIL format, style-sheets may provide an additional <switch> statement that the SMIL player will process at runtime to fit specific user preferences.

The system we have designed allows to manage these different levels of adaptation:

- SMIL templates permit to specify, within the <switch> statement, alternatives for layout and media to enclose;
- the generic part of the XSL style-sheet process the <switch> statements, selecting only the appropriate <switch> cases;
- the specific part of the XSL style-sheet allows the fine-grained specification of parameters, whose values will be chosen by the optimization algorithm. Such parameters can also define values of attributes that fulfill the conditions of the <switch> statement: in doing so, editing the generic part of the style-sheet will not be necessary at all.

4.2.3 Optimization: proposed approach

The proposed approach for the optimization through Genetic Algorithms is based on GAlib.
GAlib is an open source library that makes easier adopting GA in many optimization problems. It offers different representations for chromosomes and default genetic operators, different types of evolution, and allows the definition of custom operators for initialization, crossover and mutation.
The generic solution (the "individual", or "phenotype") is represented as a list of rectangles, each one with properties that define its dimension and position (currently these are the only parameters supported by the prototype).
Initially many individual solutions are randomly generated to form an initial population: in doing this, constraints on position and dimension of each rectangle are respected.
In each generation, the fitness of every individual is evaluated and, on this basis, multiple individuals are stochastically selected from the current population. The fitness of each individual is function of the parameters ($f_1, f_2, \ldots, f_n$), and each term may have a different relevance:

$$\text{fitness} = k_1 f_1 + k_2 f_2 + \ldots\ldots + k_n f_n$$

Presently, the terms used in the prototype are:

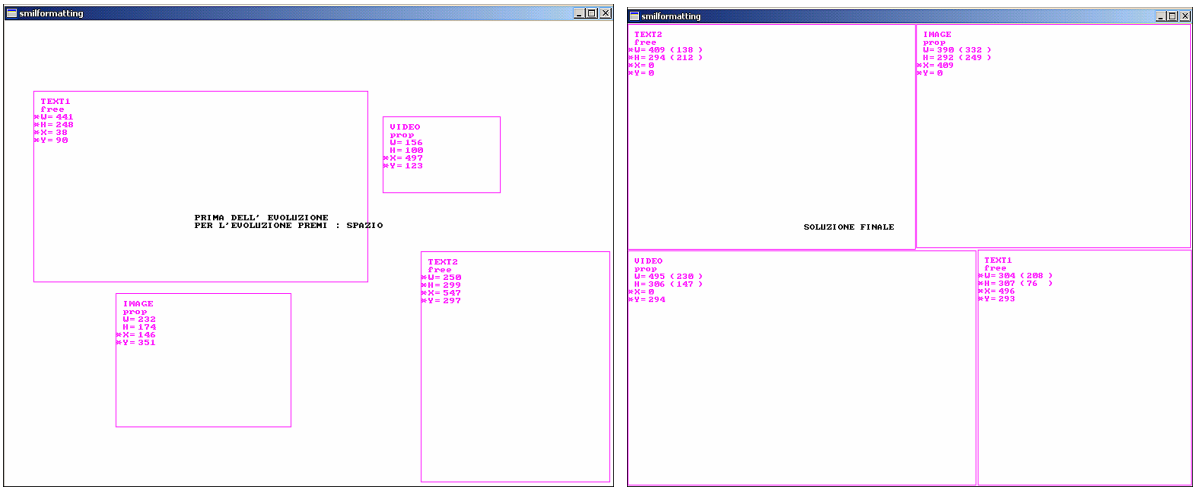
- overlap: the area overlapped between rectangles;
- misalignment: this terms evaluates the alignment between the rectangles;
- blank space: the amount of unused space.

Selected individuals are then modified to form a new population. Individuals may be recombined or mutated:

- for each pair of solutions, the implemented crossover method swaps some attributes of the n-th rectangles;
- some attributes of some rectangles are randomly selected and changed to a new random value.

Finally, the new population is used in the next iteration of the algorithm.

As example, following screenshots illustrate the optimization of a document structure composed of four regions: on the left is depicted the best individual of the initial population, on the right the best individual generated during the evolution.



Applying this approach to a SMIL presentation implies a parsing of the code to detect all the regions that compose the document and their spatial properties. Furthermore, other SMIL related problems are the management of the different layers that can be defined nesting the regions or using the *z-index* attribute.
Currently the prototype manages correctly the documents with nested regions, executing a recursive optimization of each layer.

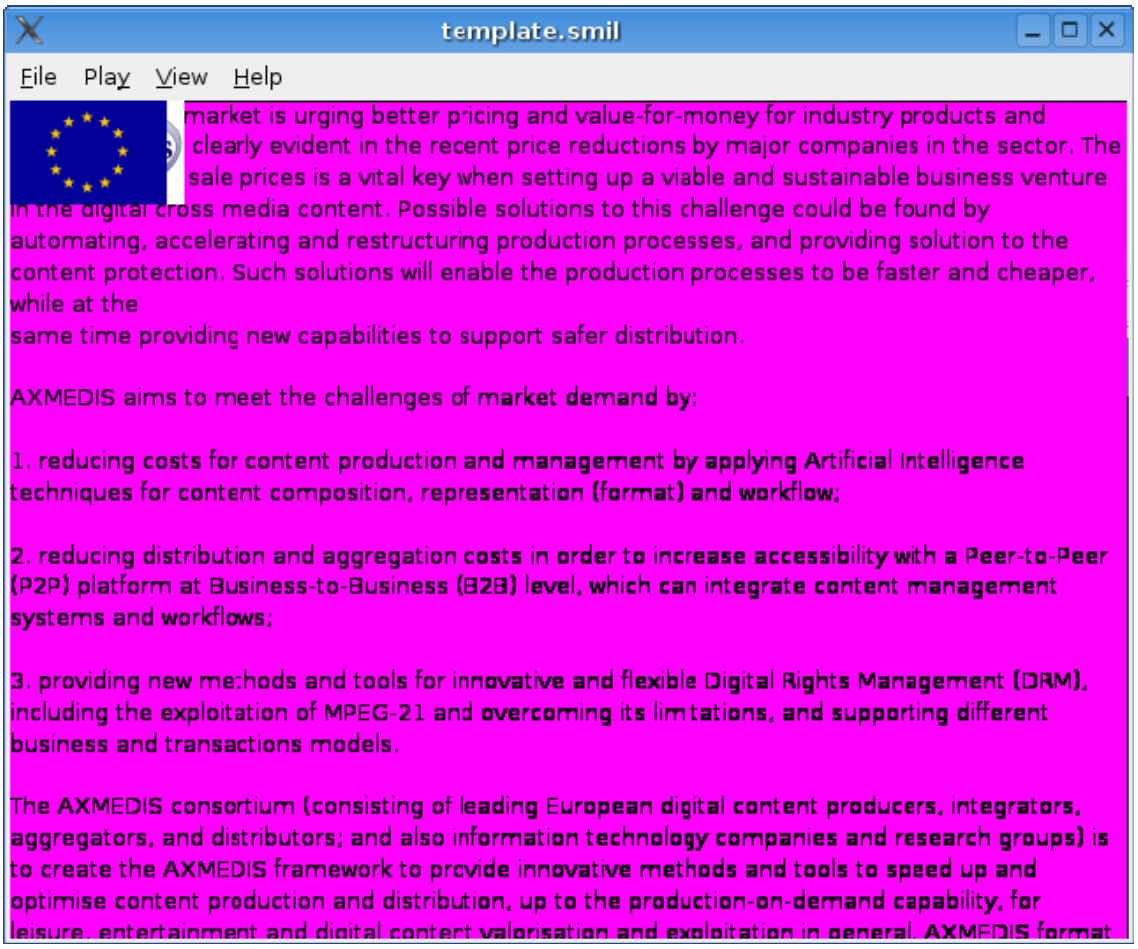
4.2.4 Format example

A simple SMIL template looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/2005/SMIL21/">
 <head>
  <!-- metadata -->
  <meta name="title" content="Axmedis Formatting Test" />
  <meta name="author" content="P. V." />
  <meta name="copyright" content="nocopyright" />
  <!-- regions -->
  <layout type="text/smil-basic-layout">
   <root-layout
       width="auto" height="auto"
       title="Smil test"
       />
   <region id="text_region"
         width="auto" height="auto"
         top="auto" left="auto"
```

```
        />
   <region id="axm_img_region"
        width="auto" height="auto"
        top="auto" left="auto"
        />
   <region id="eu_img_region"
        width="auto" height="auto"
        top="auto" left="auto"
        />
  </layout>
 </head>
 <body>
  <!-- layout -->
  <par>
   <switch>
    <text id="text1_it"
        systemLanguage="it"
        src="media/axm_it.txt"
        type="text/plain"
        region="text_region"
        />
    <text id="text1_en"
        systemLanguage="en"
        src="media/axm_en.txt"
        type="text/plain"
        region="text_region"
        />
   </switch>
   <img id="axm_img"
       src="media/logo_axm.png"
       type="image/png"
       dur="indefinite"
       region="axm_img_region"
       />
   <img id="eu_img"
       src="media/flag.png"
       type="image/png"
       dur="indefinite"
       region="eu_img_region"
       />
  </par>
 </body>
</smil>
```

The layout section describes all the regions that will be displayed in the "root-layout" window; the body section associates a media file to each region. The <switch> element in the body section is the most interesting thing in this simple template: two different text files are associated to the same region and the system is supposed to choose the right one on the basis of the systemLanguage attribute.
A SMIL player (AmbulantPlayer) shows the template in this way:



A SMIL template

The document is correctly opened by the player, since it is correct SMIL, but its structure it's not consistent and its aspect it's really poor (two images are overlapped and both overlap the text).
A basic style-sheet is the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <!-- parameters -->
 <xsl:param name="screenWidth"/>
 <xsl:param name="screenHeight"/>
 <xsl:param name="backGround"/>
 <xsl:param name="foreGround"/>
 <xsl:param name="menuWidth"/>
 <xsl:param name="textWidth"/>


 <!-- SMIL structure -->
 <xsl:include href="smilstruct.xsl" />

 <!-- attributes for root-layout -->
 <xsl:template match="root-layout">
  <root-layout
     width="{$screenWidth}" height="{$screenHeight}"
     backgroundColor="{$foreGround}"
     title="Smil+XSL test"
```

```
    />
 </xsl:template>

 <!-- attributes for regions -->
 <xsl:template match="region">
  <xsl:choose>
   <xsl:when test="@id='axm_img_region'">
    <region
        id="{@id}"
        backgroundColor="{$backGround}"
        fit="hidden"
        regPoint="center"
        regAlign="center"
        width="110" height="70"
        top="{$screenHeight * 0.05}"
        left="0"
        z-index="1"
        />
   </xsl:when>
   <xsl:when test="@id='eu_img_region'">
    <region
        id="{@id}"
        backgroundColor="{$backGround}"
        fit="hidden"
        regPoint="center"
        regAlign="center"
        width="100" height="70"
        bottom="{$screenHeight * 0.05}"
        left="0"
        z-index="1"
        />
   </xsl:when>
   <xsl:when test="@id='text_region'">
    <region
        id="{@id}"
        backgroundColor="{$backGround}"
        fit="scroll"
        width="{$textWidth}"
        left="{$menuWidth}"
        height="{$screenHeight * 0.9}"
        top="{$screenHeight * 0.05}"
        z-index="2"
        />
   </xsl:when>
   <xsl:otherwise>
    <!-- include the region as is -->
    <xsl:copy-of select="."/>
   </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
</xsl:stylesheet>
```

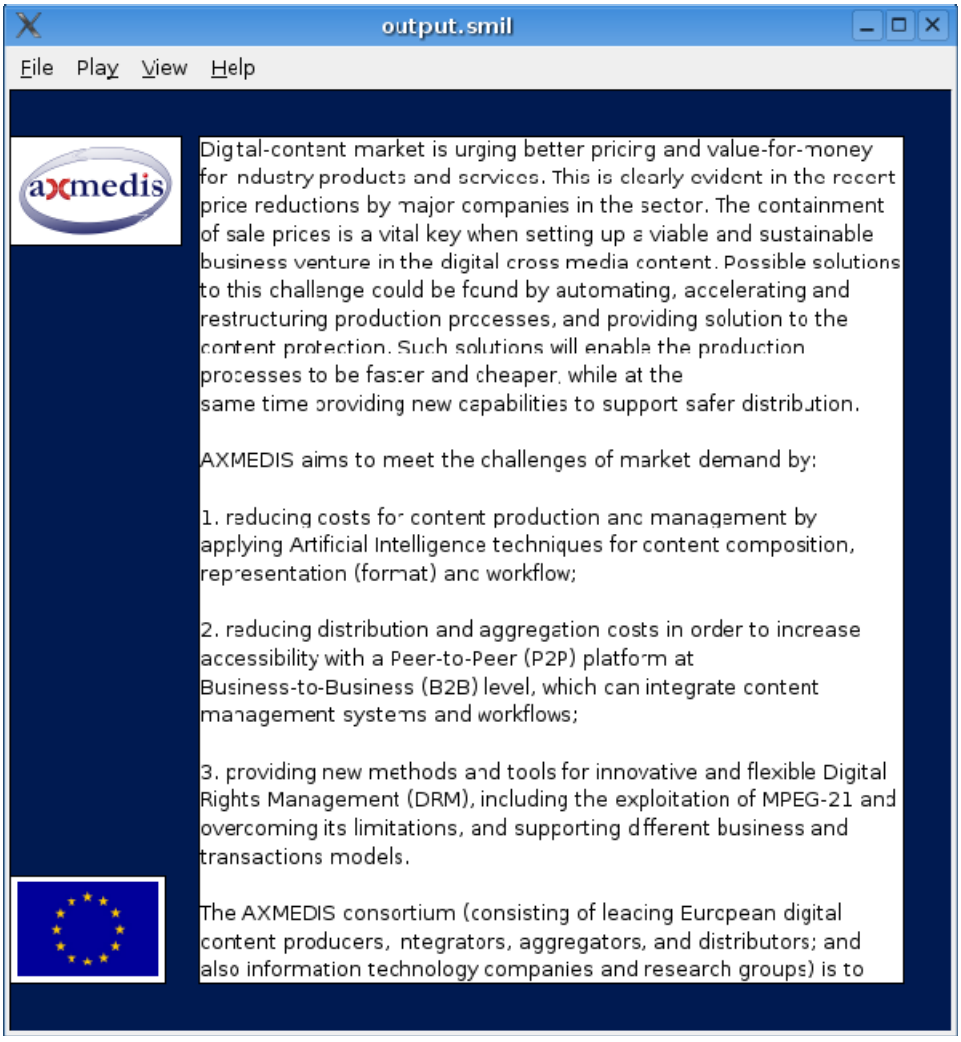
The style-sheet starts with the declaration of the parameters: their values will be determined by the optimization algorithm. The main part of the style-sheet is for region attributes: their values are redefined to customize the presentation; most of them are based on the parameters. The style-sheet include an external section (smilstruct.xsl):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <!-- build the basic SMIL structure -->
 <xsl:template match="smil">
```

```
 <smil>
  <head>
   <xsl:apply-templates select="head"/>
  </head>
  <body>
   <xsl:apply-templates select="body"/>
  </body>
 </smil>
</xsl:template>
<!-- apply templates for head -->
<xsl:template match="head">
 <xsl:apply-templates/>
</xsl:template>
<!-- apply templates for body -->
<xsl:template match="body">
 <xsl:apply-templates/>
</xsl:template>
<!-- apply templates for layout -->
<xsl:template match="layout">
 <layout type="text/smil-basic-layout">
  <xsl:apply-templates/>
 </layout>
</xsl:template>
<!-- apply templates for switch, using my preferences -->
<xsl:template match="switch">
 <xsl:apply-templates select="*[@systemLanguage='en']|
                       *[@systemScreenDepth='16']|
                       *[@systemCaptions='on']|
                       *[@systemOperatingSystem='linux']|
                       *[@systemScreenSize='800x600']
                       "/>
</xsl:template>
<!-- apply templates for par -->
<xsl:template match="par">
 <par>
  <xsl:if test="@id">
   <xsl:attribute name="id">
    <xsl:value-of select="@id"/>
   </xsl:attribute>
  </xsl:if>
  <xsl:apply-templates/>
 </par>
</xsl:template>
<!-- apply templates for seq -->
<xsl:template match="seq">
 <seq>
  <xsl:if test="@id">
   <xsl:attribute name="id">
    <xsl:value-of select="@id"/>
   </xsl:attribute>
  </xsl:if>
  <xsl:apply-templates/>
 </seq>
</xsl:template>
<!-- apply templates for excl -->
<xsl:template match="excl">
 <seq>
  <xsl:if test="@id">
   <xsl:attribute name="id">
    <xsl:value-of select="@id"/>
   </xsl:attribute>
  </xsl:if>
```

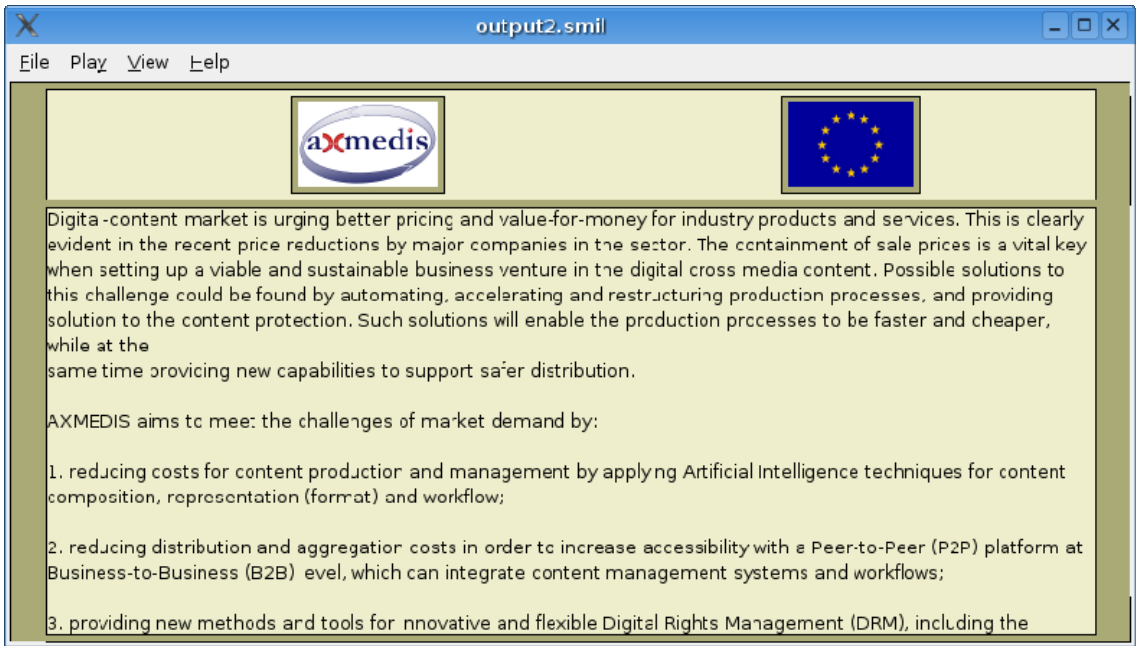
```
    <xsl:apply-templates/>
   </seq>
  </xsl:template>
  <!-- media -->
  <!-- apply templates for img -->
  <xsl:template match="img">
   <img
     id="{@id}"
     src="{@src}"
     type="{@type}"
     region="{@region}"
     >
    <xsl:if test="@dur">
     <xsl:attribute name="dur">
      <xsl:value-of select="@dur"/>
     </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
   </img>
  </xsl:template>
  <!-- apply templates for text -->
  <xsl:template match="text">
   <text
     id="{@id}"
     src="{@src}"
     type="{@type}"
     region="{@region}"
     />
  </xsl:template>
  <!-- apply templates for audio -->
  <xsl:template match="audio">
   <audio
     id="{@id}"
     src="{@src}"
     type="{@type}"
     >
    <xsl:if test="@dur">
     <xsl:attribute name="dur">
      <xsl:value-of select="@dur"/>
     </xsl:attribute>
    </xsl:if>
   </audio>
  </xsl:template>
  <!-- apply templates for area -->
  <xsl:template match="area">
   <area
     href="{@href}"
     />
  </xsl:template>
</xsl:stylesheet>
```

This section is standard for every style-sheet: its goal is simply recreating the basic structure of a SMIL document, based on the elements that appear in the template. The most interesting part is the <switch> processing: here we are default values for selecting options in the <switch> statement.
The result of the style-sheet application to the template, depending on values assigned to parameters, could be the following:



A styled presentation

Other style-sheets may be applied at the document we have just produced, modifying its aspect; for instance:



Another style-sheet

5 Content Formatting Tools (DSI)

These tools offer a GUI for manual or automatic formatting of multimedia presentations. The "presentation" is intended here as the most generic type of multimedia content that may be diffused through a network, which includes all kinds of single and composed media and the interactions among them and with the user. The format of the presentation has to be automatically selected between the previously created models and has to be adaptable to a wide range of platforms and user preferences.
The authoring of such a multimedia presentation is a complex process that requires the specification of several types of information: the media items to use, their spatial layout, links, interactions, and temporal relationships between them. Moreover, the increased number of platforms and network connections that are used to obtain multimedia contents, requires an additional step for the final optimization.
Content Formatting Tools are designed to be integrated into the AXMEDIS SMIL Editor.

5.1 Template Editor

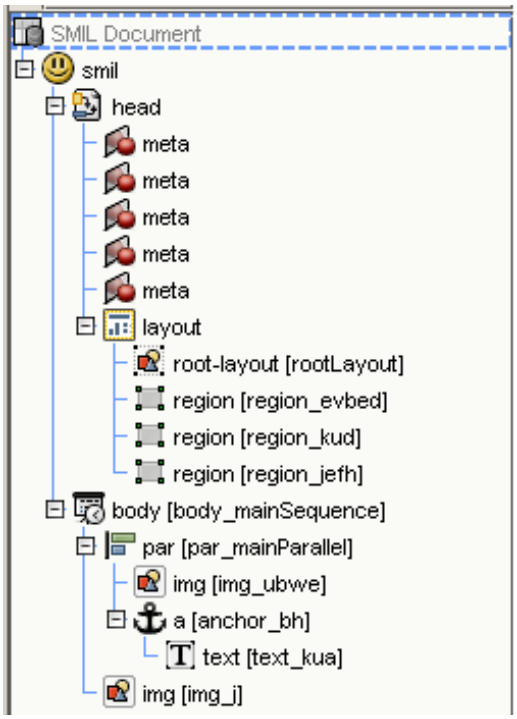
The Template Editor allows to create the basic structure for a new type of document. In the Formatting System, templates are written in SMIL: they have to define the spatial regions of the presentation, their associations with the media, their temporal relationships and their interactions. Although the spatial properties have not to be defined in the template, a good SMIL editor is required to accomplish the task of defining timing and synchronization.
The Template Editor GUI allows the user to:

- create and edit (with mouse operations) rectangular regions on the screen;
- associate the rectangles with the <region> elements of the SMIL language;
- associate media file to the rectangular regions;
- access a context menu to set regions attributes (e.g.: identifier, position, dimensions, colors) that will be immediately reflected in the drawing;

Attribute	Value
background-color	[Empty]
backgroundColor	white
bottom	[Empty]
fit	scroll
height	269
id	region_evbed
left	294
right	[Empty]
showBackground	[Empty]
top	198
width	329
z-index	[Empty]

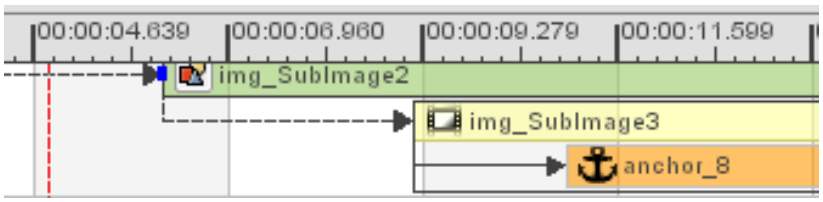
Example of attributes menu

- switch by one touch tabs among different views such as text view (XML text view), composite media scene view (WYSIWYG), tree view (XML elements), single media view (video, audio, images, etc.);



Example of tree view

- edit timing and synchronization properties with a timeline view.



Example of timeline view

- syntax coloring and automatic indentation in XML text view;
- code completion in XML text view.

Examples of SMIL editors with these capabilities are:

- LimSee2 (http://wam.inrialpes.fr/software/limsee2/) a free (for not commercial purposes) and open-source Java application that features a powerful WYSIWYG graphical user interface designed to ease the manipulation of time-based multimedia;
- GRiNS (http://www.oratrix.com/Products/G2E), a commercial product (with a free trial version) that features a high level of control over presentation making, multiple exports, complete control for the all SMIL 2.0 Content Control constructs.

We planned to use as Template Editor the SMIL Editor included within the AXMEDIS Framework, which will provide such functionalities. Additionally, the editor should check the syntax against an arbitrary DTD, to allow extensions to the SMIL language.

5.2 Automatic Template Selector

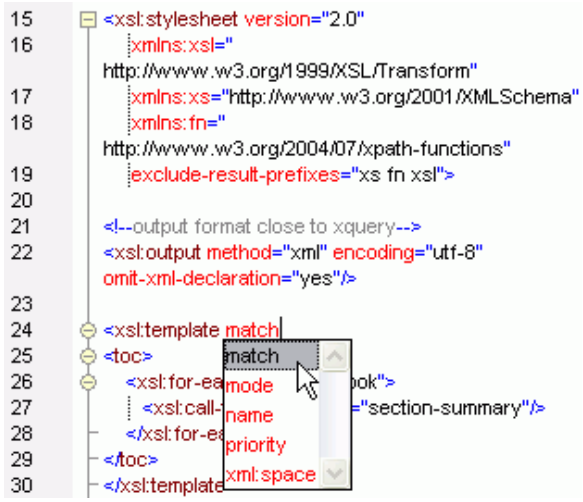
The Automatic Template Selector performs an input mapping, based on selected media and other informations provided by the author.

The input media set, that varies for type and number of elements, may be considered as composed of subsets to get more informations about content destination: these subsets should represent the "multimedia primitives" (scenes, clips, slides, etc.). Such subsets may eventually derive from querying process or be specified by the author and they are defined in the Compositional Process (see section 3.5). The author also specifies if interactions with the user are needed or not, and other general preferences.

5.3 Style-sheet Editor

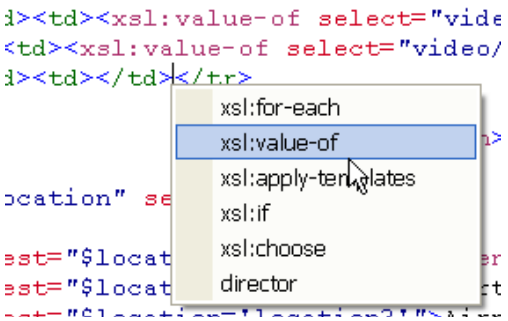
The Style-sheet Editor allows to create and modify the XSL style-sheets for the templates. XSLT is a powerful general-purpose transformation language and purely visual editors can not fully exploit its expressivity. Thus the Style-sheet Editor has to offer both visual and text tools.
Text tools provide:

- syntax coloring and automatic indentation;
- code completion;



Example of code completion and syntax highlighting

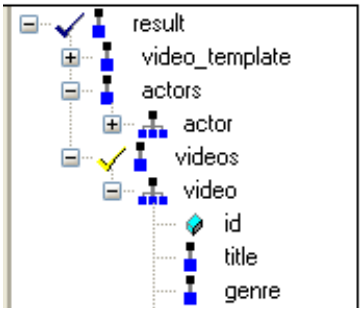
- available functions listing and their prototypes.



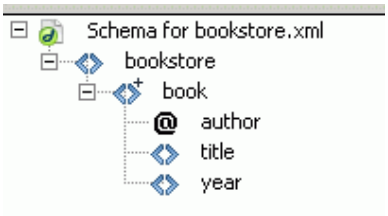
Example of functions listing

Visual tools provide:

- a source tree view for the input XML (SMIL) document, featuring the canonized (simplified and symbolic) representation of the tree;



Example of symbolic XML tree view



Example of canonized XML tree view

- an xsl:template view;
- association between xsl:template and source tree elements with mouse operations;
- result preview.

A good examples of XSL editor is the Stylus Studio's suite (http://www.stylusstudio.com/xslt.html), that offers an XSLT Editor, an XSLT Mapper and WYSIWYG XSLT Designer.
Text tools may be almost the same used in the Template Editor. Probably, visual tools have to be created specifically for the Style-sheet Editor.

6 Workflow Management and database (IRC)

The workflow editor and viewer is the gateway interface for creating and changing new project workspaces referred to as NPDs in the terminology adopted for the AXMEDIS Workflow and object life cycle analysis elsewhere in our document.

Naturally the functionality of this editor/viewer at the level of NPD editing will be a subset of the use cases already set-out for the AXMEDIS workflow management system particularly focusing on the global management requirements of the NPD workspace including Actors, Objects, Processes, etc.

It is possible for the AXMEDIS workflow management system to support inter-factory workflow. An example of this would be collaborating content producers who work jointly on common objects. Content Factory A would create an object, then Factory B would perform some activities to add value to the object, then returning it to Factory A for completion. Conceptually, this process is identical to the normal, intra-factory scenario where activities are carried out in one content factory. In the inter-factory scenario, the collaborating factories needs to establish an agreed workflow in order to manage their division of work productively and efficiently. This workflow agreement can be modelled in the same manner as a conventional intra-factory workflow.

We have not proposed for centralised single server architecture; rather each partner will have their workflow running with their part of the project workflow definition. The transitions resulting into change of partner is defined in the workflow to reflect the collaborative workflow logic as agreed between the collaborators. The waiting period for the factories can be defined as "Idle/wait" activities within the workflow which are completed upon receiving the workitem from the external factory. For example when the workitem is handed to Factory B from Factory A, as defined in the workflow, Factory A will then start an "Idle/wait" activity which will end upon receiving the workitem back from Factory B.

It is important that collaborating factories therefore share common WFMS tools in order to manage and track the progress of an NPD across their combined activities. This enables dynamic planning and scheduling of resources across the factories, much in the way that automotive companies operating just-in-time policies use integrated logistics systems to track components through their value chain.

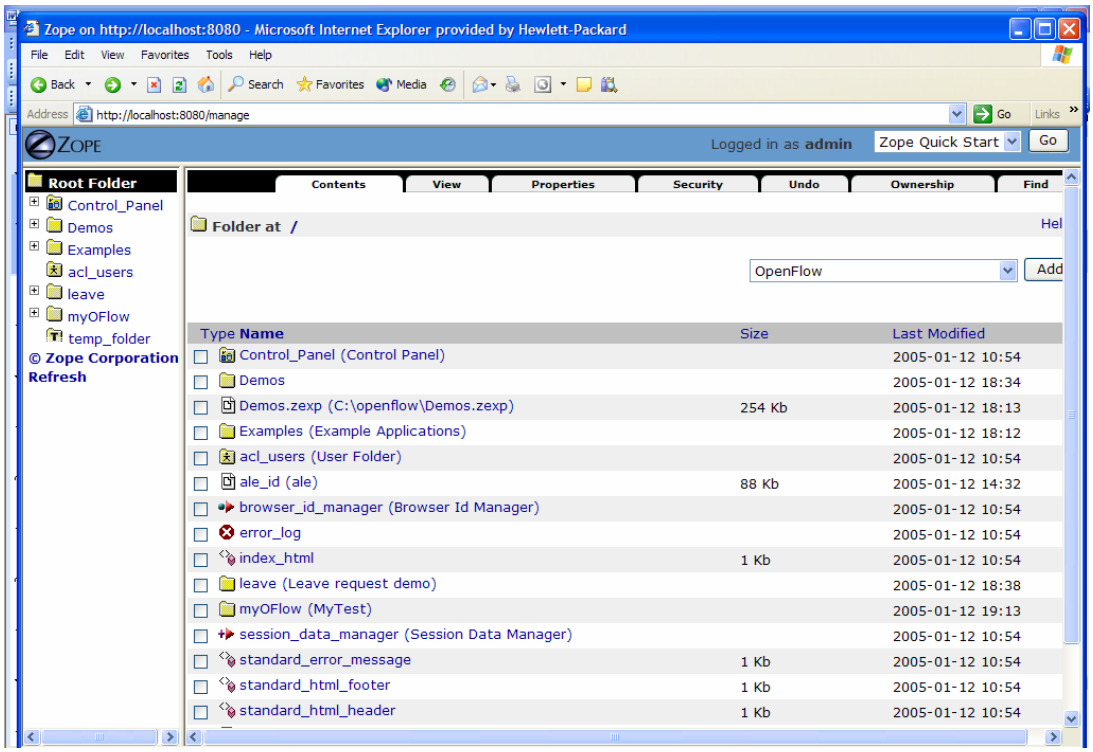
This dynamic visibility would not be possible if separate WFMS tools were employed in each factory and the only communication available were some embedded historic metadata within objects passed between factories.

For this reason, a common web-based editor is used for the AXMEDIS Workflow user interface, which is capable of being accessed from multiple collaborating content producers, integrators and distributors sharing a common inter-factory workflow.

6.1 Writing and describing workflow, harmonising AXMEDIS tools

Openflow runs on the Zope platform which is managed through the "Zope Management Interface" using industry standard browsers, typically by logging on as the administrator (admin) at URL http://localhost:8080/manage. The screen shot below shows an example of this management interface.

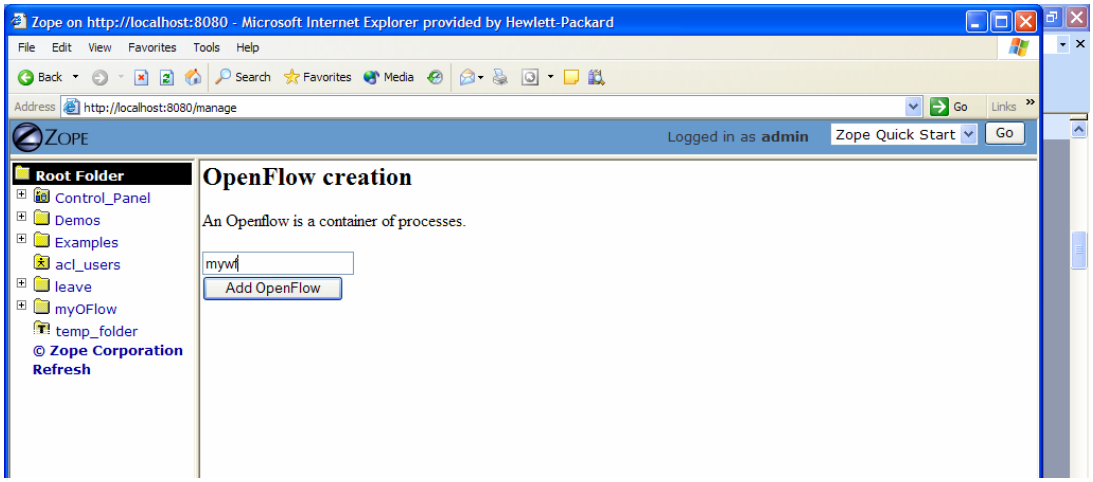
Creating a new process in openflow is a multi-step process which begins with adding an OpenFlow container using the Zope management interface as shown below (delineated by a an ellipse in red).



Adding an OpenFlow container through the Zope Management Interface

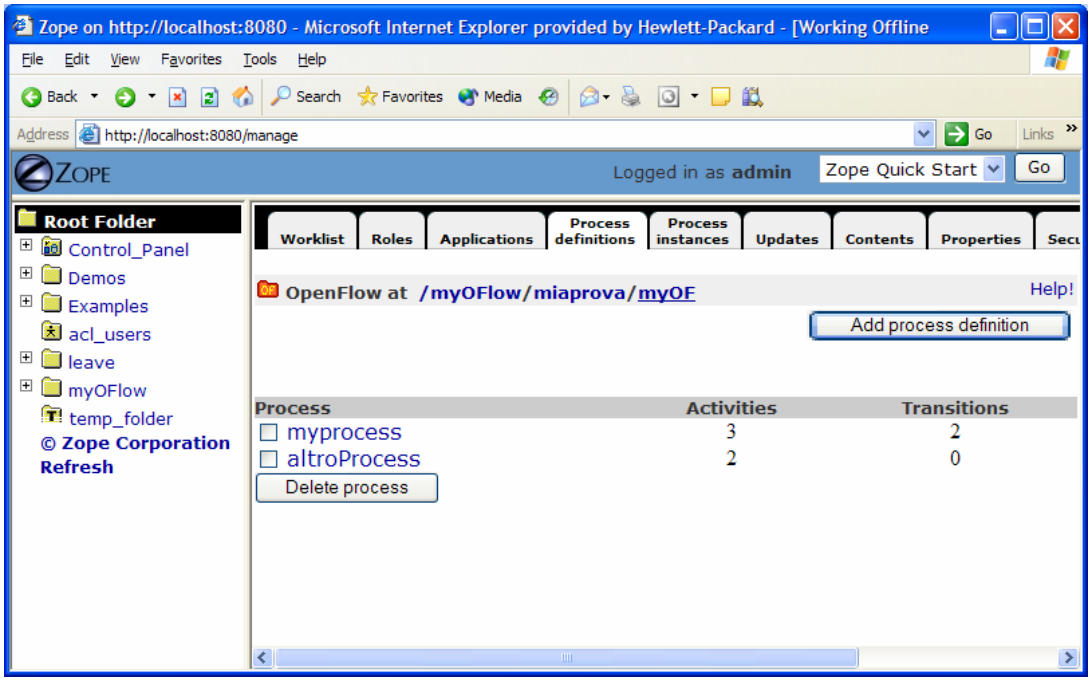
During the creation of the OpenFlow container, the name of the container must be specified as shown in the next screen-shot.

.

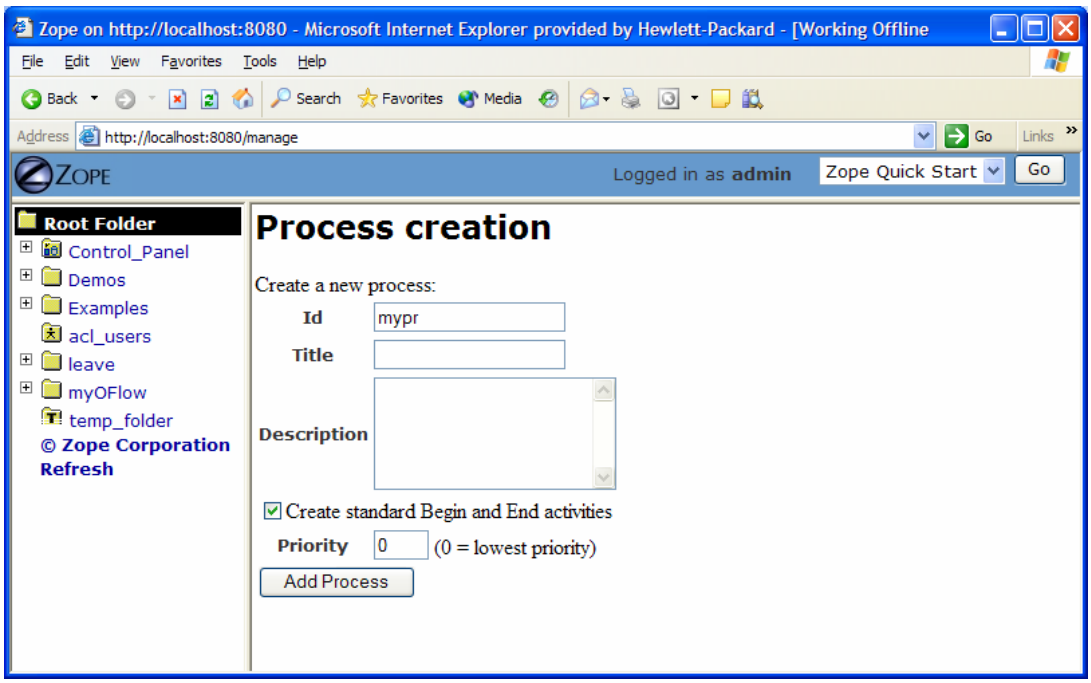


Creating the OpenFlow container

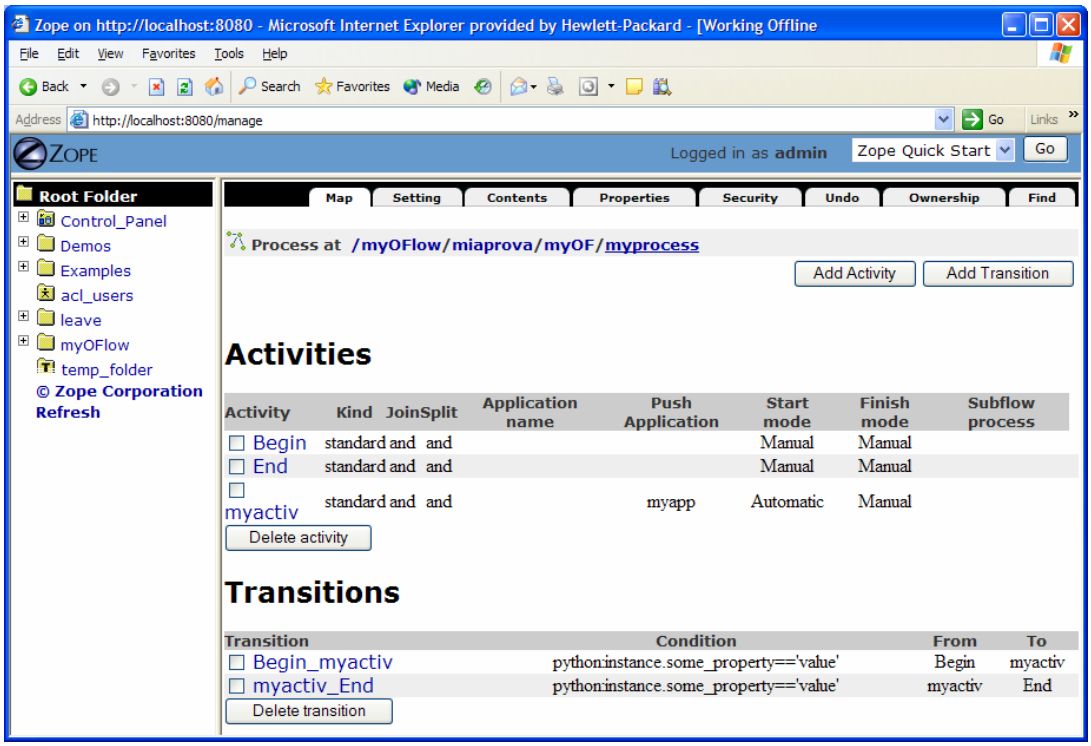
Next it is necessary to define the process and the activities pertaining to the process, together with their transitions (From Activity and To Activity). These operations are performed by accessing the tabs in the Openflow container as shown in the following screen-shots:



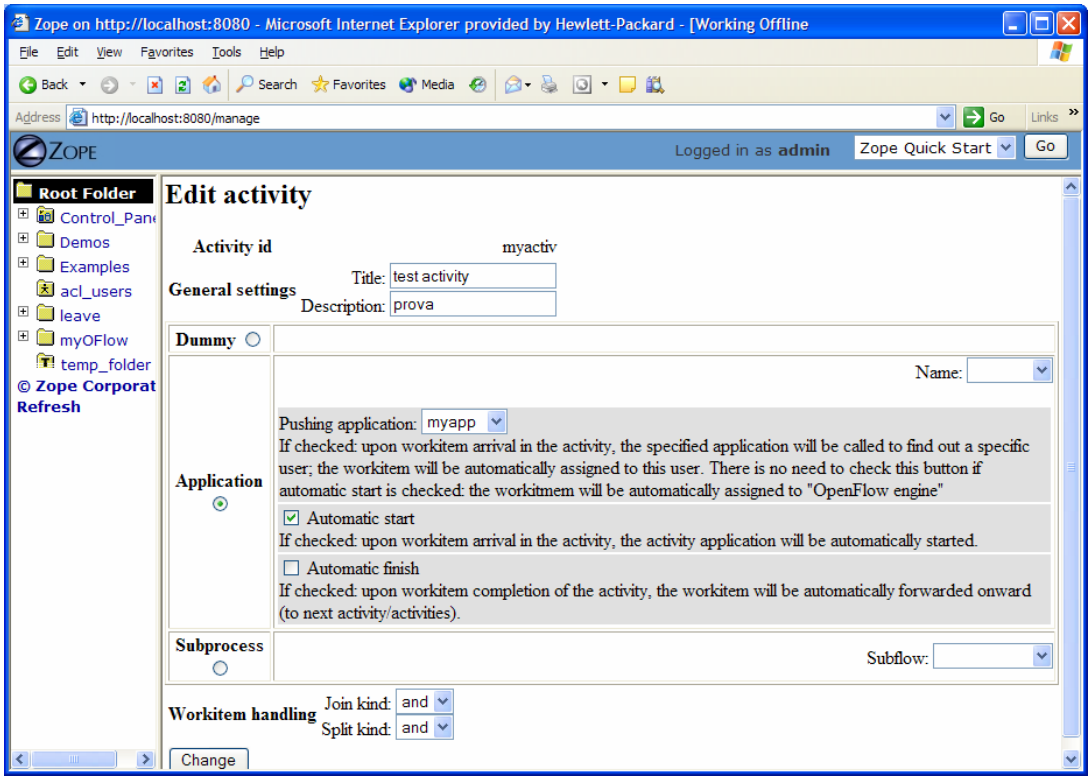
The process definition tab



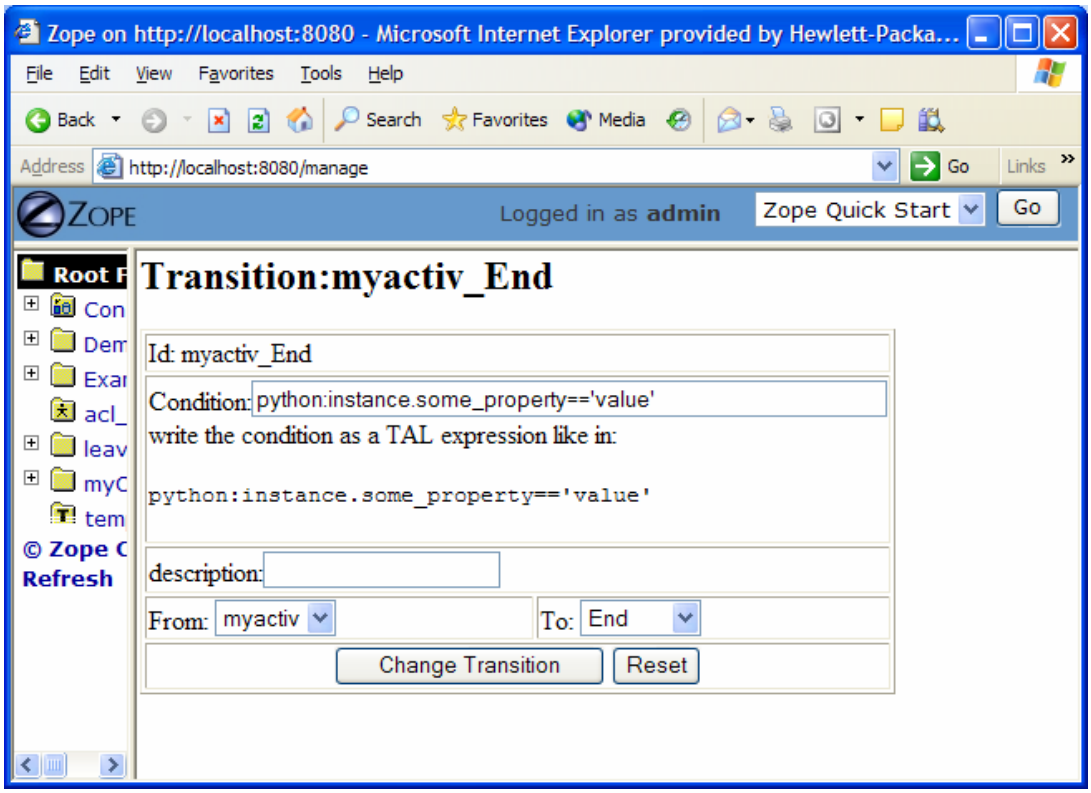
Creating a new Process definition



Management of activity and transitions of a process

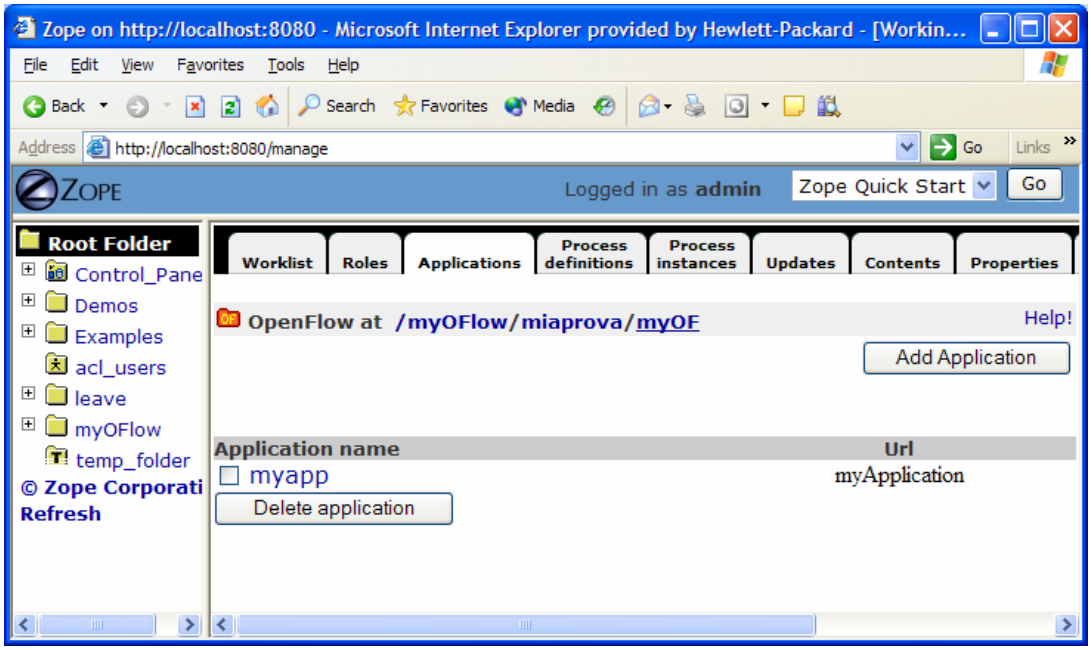


Editing a process activity



Defining process transition and related conditions

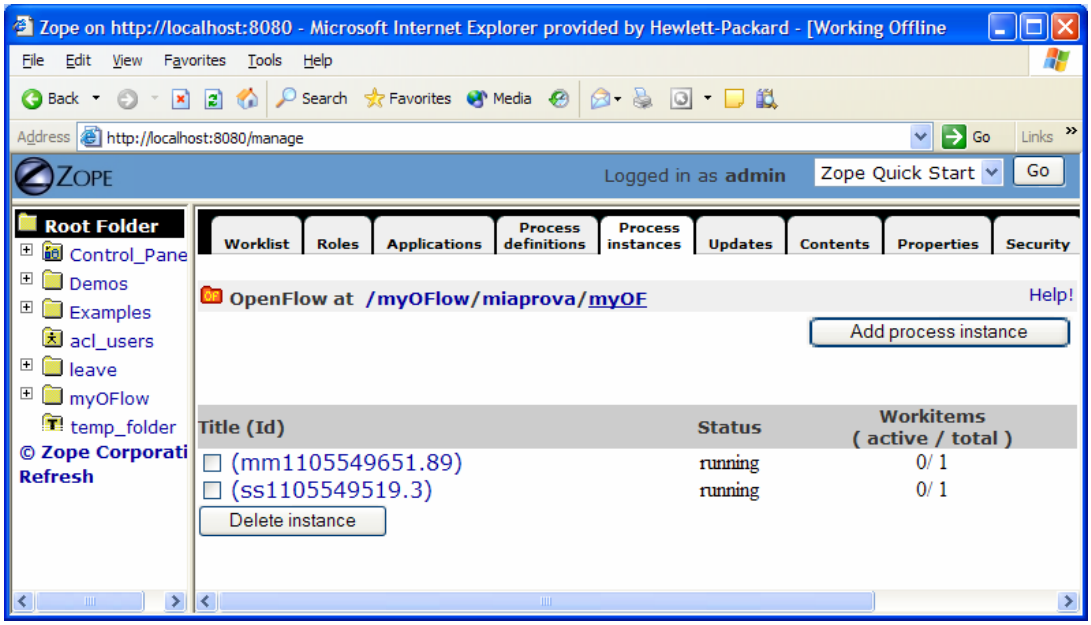
Applications associated to the activities are then specified selecting the Applications Tab.



: Defining process applications

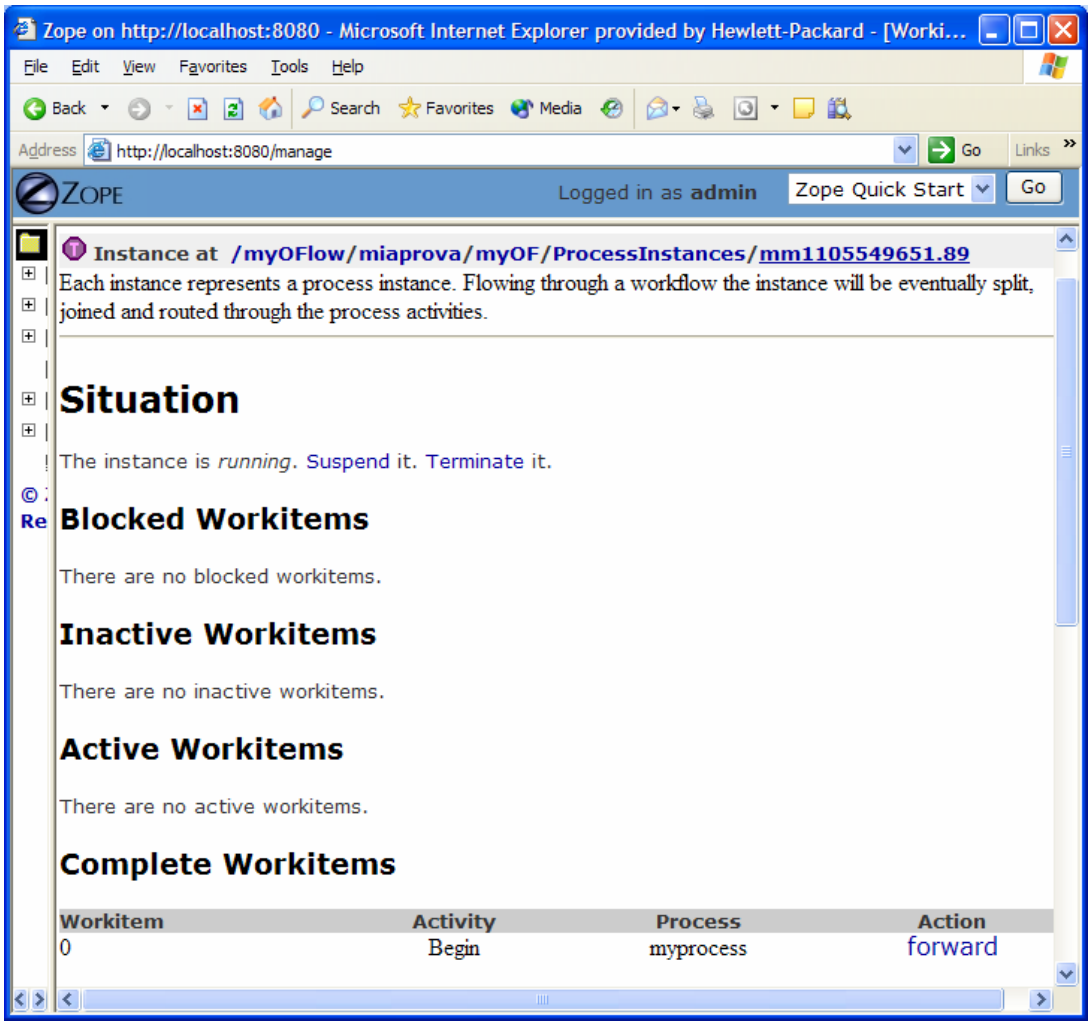
The users and roles are configured as Zope users and roles as access control list (acl_users).

Once a process has been defined it can be tested. An instance of the process can be created and executed directly in the processflow-instance management tab shown below.



Process instance management tab

The following Figure shows the of the workitems involved in the process instance that has been created.



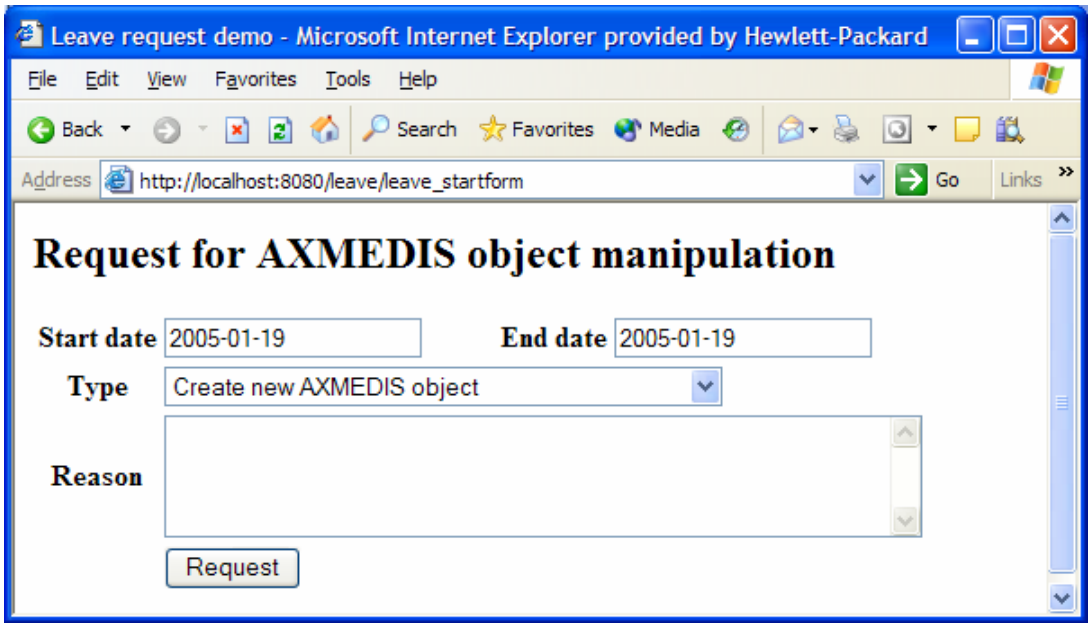
Monitoring and management of a specific process instance

Process Example:

The following simple example illustrates a process to request a AXMEDIS object manipulation (*a mock-up process*). This is an example of explicit forwarding to different actors having different roles. The first actor requests the creation of a new AXMEDIS object by filling out a form. The request goes to the second actor (called Socius) who checks that the request is acceptable. The request is then forwarded to the third actor (called Prefectus) for approval.

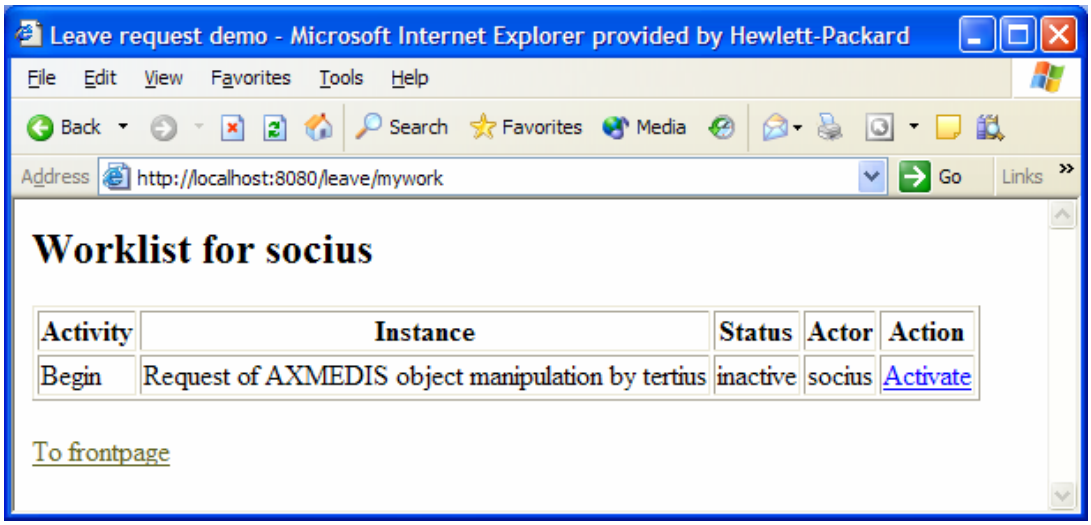
The following steps are necessary for the above example process to be enacted:

The first actor (called Tertius) enters an AXMEDIS object manipulation request by filling out the following form as shown in the screen-shot below:



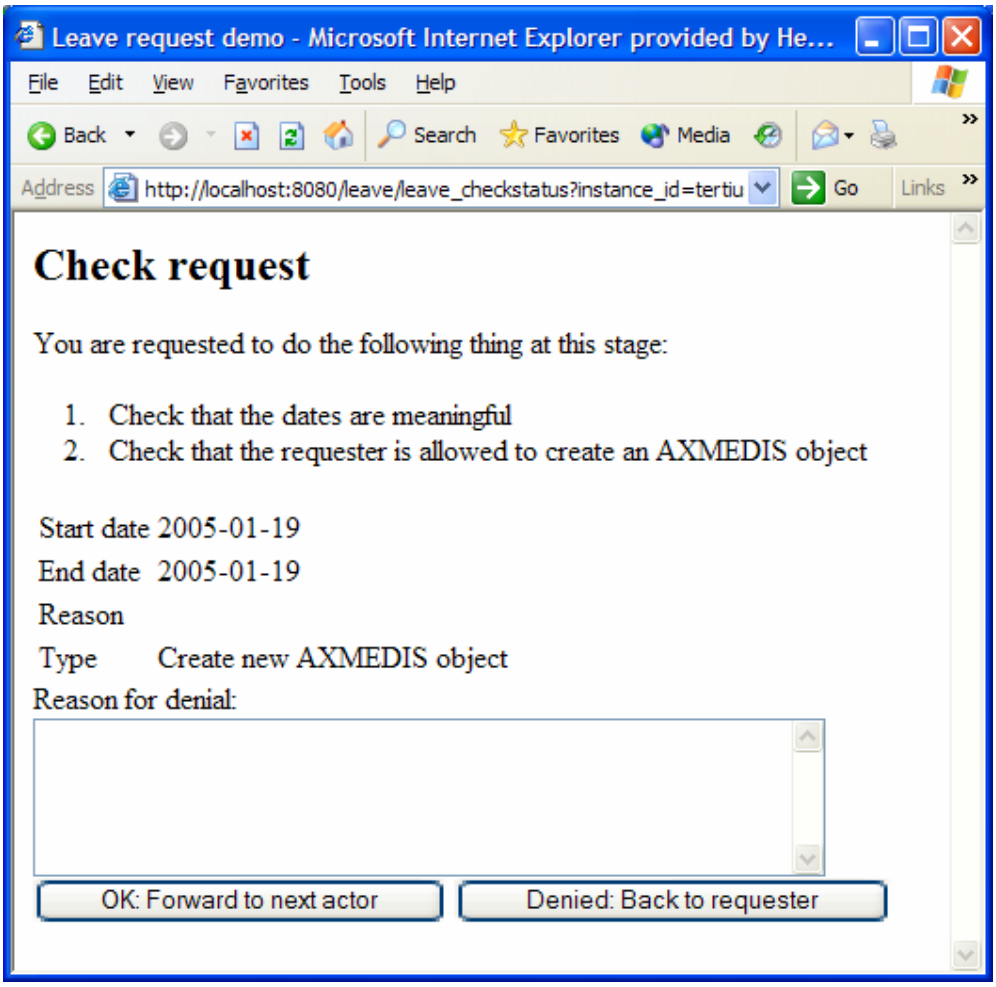
Tertius' AXMEDIS object manipulation form

According to the processflow, the request goes to the next actor (called Socius). When Socius logs in, his work list shows that there is a workitem in his worklist as shown in the screen-shot below:



Socius' worklist and workitem activation

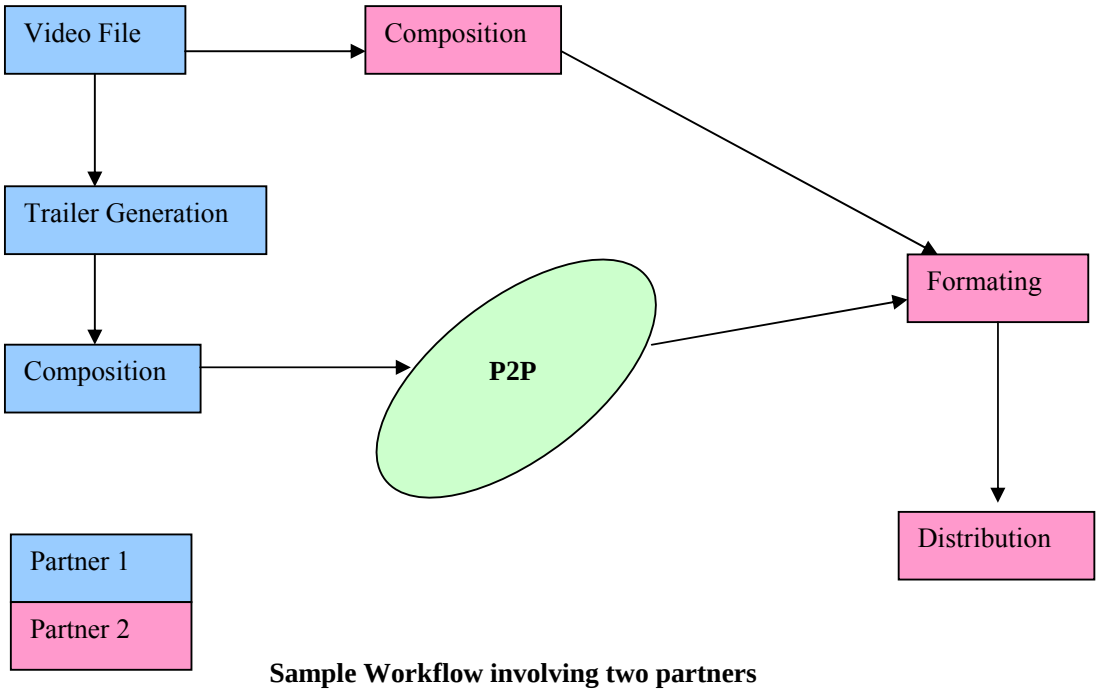
To execute the workitem, the actor (Socius) has to activate the workitem (Begin) and perform the related activities. Next this actor either forwards the workitem to the next actor, which in this case is the supervisor (called Prefectus), or rejects the request; as illustrated by the screen-shot below:



Socius' workitem execution and forwarding

Then the activity is forwarded to the last actor and the process ends.

The sample workflow that is being defined for demonstration purpose, will be a inter factory workflow utilising the important Axmedis tools like AXEPTool, PnP Engine, Axmedis Editor, etc. The following diagram is the schematic of the sample workflow to be deployed for the demonstration. It includes two factories sharing some content using AXEPTool. This workflow will be further refined to map to the actual low level atomic activities that are available within the Axmedis Framework.



Sample Workflow involving two partners

The above workflow involves two partners. The partner 1 is responsible for generating the trailer from a selected video file and then publishes over P2P network. The partner 2 downloads this trailer video from the P2P network along with the original video file from the database. These two files are then formated to be distributed to the user.

7 Workflow Integration of tools (IRC)

The Axmedis Workflow integration involes developing of following modules as per the specifications:

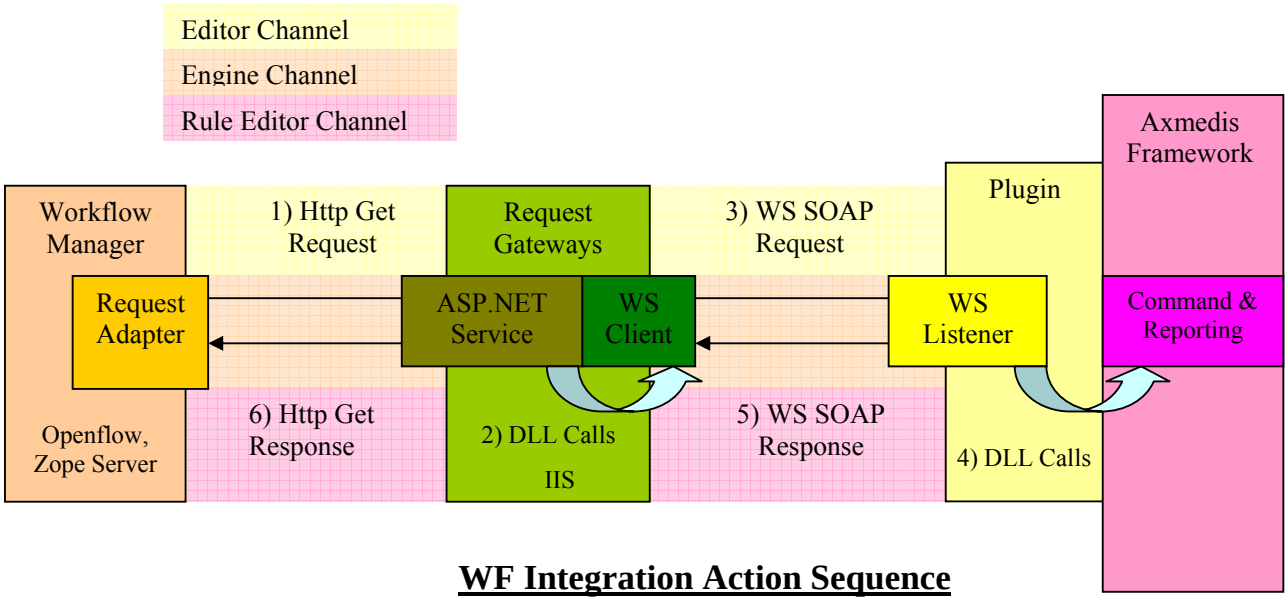
The AXMEDIS WorkFLow Area includes:

- WorkFLow Management User Interface and Tool
- WorkFlow Engine
- WorkFlow DataBase
- WF AXOM Request Adapter
- WF AXOM Input Queue Adapter
- WF Engine Request Adapter
- WF Engine Input Queue Adapter
- WF Rule Editor Request Adapter
- WF Rule Editor Input Queue Adapter
- WF DB Request Adapter
- WF DB Input Queue Adapter
- AXOM WorkFlow Gateway
- E ngine WorkFlow Gateway
- Rule Editors WorkFlow Gateway
- DB WorkFlow Gateway

The overall integration is divided into four channels in order to group the common functionalities together as follows:

- Workflow Editors Interfaces
- Workflow Engines Interfaces
- Workflow Rule Editors/Viewers Interfaces
- Workflow Query Support Interfaces

As part of the first prototype development, we have developed the communication path which is the backbone for the overall integration. The following diagram shows the modules that were developed for the first prototype and the protocols used for laying the foundation.



WF Integration Action Sequence

The following modules were delivered for the first prototype integration of Axmedis workflow:

1. Editor Channel
- i) Request Adaptors
  - a) Source Code
  - b) Documentation
- ii) Request Gateways
  - a) Source Code
  - b) Documentation
- iii) Plugin
  - a) Source Code
  - b) Documentation

2. Rule Editor Channel
- i) Request Adaptors
  - a) Source Code
  - b) Documentation
- ii) Request Gateways
  - a) Source Code
  - b) Documentation
- iii) Plugin
  - c) Source Code
  - d) Documentation

3. Engine Channel
- i) Request Adaptors
  - a) Source Code
  - b) Documentation
- ii) Request Gateways
  - a) Source Code
  - b) Documentation

iii) Plugin
   a) Source Code
   b) Documentation

4. Openflow

i) User Interface
   a) Source Code
   b) Documentation

For the next phase of project, we have planned to complete following task:

- Writing workflow for content production in one of the partners factory.
- Black-box testing for the overall integration using the workflow defined.
- White-box testing to eliminate any inaccurate results.

7.1 Integration Support with content processing tools (AXCP processing tools: engine and scheduler)

The workflow engine interacts with the following content processing tools:

- Axmedis Compositional/Formatting Engine through AXCP schedular
- Axmedis Program and Publication Engine

The integration is realised as per the specification document. The integration involves two distinct channels for workflow communication: Request Channel and Response Channel.

The Request Channel consists of Workflow Request Adaptors, Workflow Request Gateways and Webservice Listeners, while the Response channel consists of Response Gateways and Input Queue Adaptors.

The Axmedis Engine WorkFlow channel passes through the WF Engine Request Gateway where the Engine Command and Reporting exposes the following methods, via WebServices:

- Install_and_activate for installing a XML rule in the scheduler and activate it. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Run_rule for immediately run a rule already loaded inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Activate_rule for activating a rule already loaded inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- deactivate_rule for disabling a not-running rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Suspend_rule for suspending a rule for a specified time interval. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Pause_rule, for suspending a rule until it will be restarted. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.

- Kill_rule for stopping the execution of a rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Remove_rule for removing a rule from the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Resume_rule for resuming a paused rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_rule_status for getting the status of a rule inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_rule_logs for getting history log of a rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_list_of_rules for getting the list of the rules of a certain user inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_rule for getting the XML definition of a rule inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Status_request_to_PnP for getting the status of a Program of the Program and Publication Engine
- Suspend_PnP_Program for suspending a Program of the Program and Publication Engine
- Abort_PnP_Program for aborting a Program of the Program and Publication Engine
- Resume_PnP_Program for resuming a suspended Program of the Program and Publication Engine
- Activate_PnP_Program for acrivating a Program of the Program and Publication Engine
- WorFlow_Notification is used to return back to the requesting engine (basically PnP) the status about the requested execution of a WorFlow process

The method invocation is performed via a WebService request where the parameters are sent (to Engine Command and Reporting) and received back (in WebService result) from the Engine Command and Reporting

The methods invoked and the parameters invoked by WF Engine Request Adapter to the WF Engine Request Gateway are the same described in the preceeding Paragraph. Their encoding, however, is different. The request in in fact sent through an http GET call where the paramteres are invoked as follows:

As an example to understand the response, consider the compositional/formatting engine response to a request.

GET/Control_Panel/Products/OpenFlow/AXWF/comp_format_request_status?Credentials=”Credential string”&AXRQID=”Request ID string” HTTP/1.1" 200 368 "http://15.156.120.195:8080/Control_Panel/Products/OpenFlow/leave/leave_startform" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)"

Where “Credential string” is a string containing the credentials and “Request ID string” is a string containing the Request ID.

The response to the invoked method has the same contents listed in the preceeding Paragraph and is sent via an http GET response. The response is XML coded, following the schema:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="Engine_Response">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="result" type="xs:boolean"/>
				<xs:element name="errormsg" type="xs:string" nillable="true"
minOccurs="0"/>
				<xs:element name="errorcode" type="xs:int"/>
				<xs:element name="ruleid" type="xs:string" minOccurs="0"
maxOccurs="20"/>
				<xs:element name="status" type="xs:string" minOccurs="0"/>
				<xs:element name="xml_rule_schema" type="xs:string" minOccurs="0"/>
				<xs:element name="rulelog" type="xs:string" minOccurs="0"
maxOccurs="100"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

The Engine Command and Reporting send its notifications to the WorkFlow Engine by calling a WebServices exposed by the WF Engine Response Gateway.

The URI of the WebService is indicated in the previous request made by the WF Request Gateway in the EngineListenerService parameter.

The Notification also contain the original Request ID issued in the request (AXRQID).

7.2 Integration Support with Editors (AXCP rule editor and AXMEDIS editor)

The workflow engine interacts with the following Editor:

- Axmedis Editors (Object, DRM, Metadata, …)
- Axmedis Rule Editors (PnP, Content Processing, AXEPTool, etc).

The integration is realised as per the specification document. The integration involves two distinct channels for workflow communication: Request Channel and Response Channel.

The Request Channel consists of Workflow Request Adaptors, Workflow Request Gateways and Webservice Listeners, while the Response channel consists of Response Gateways and Input Queue Adaptors.

The Axmedis Workflow Manager communicate to AXOM's Command and Reporting through WF Editor Request Adapter. The WF Request Adapter sends the requests via an http GET call. This http GET call is received by a Web Server running Microsoft IIS and directed to an ASP process called WF Request Gateway. This ASP process decodes the GET requests and formats a WebService request towards the proper Axmedis module, Axmedis Editor in this case. As AXOM, along with Command and Reporting, is a static library, a listener service is required to listen to incoming Request from Workflow and invoke Axmedis Editor accordingly. We call this listener service as AXOM_WebServices_Listener which will be resident on client's machine. AXOM WebServices Listener is a multithreading process written in C++ which exposes methods through WebServices, listens to them and forwards the requests to the AXOM Command and Reporting module which is a C++ library. So the interface between AXOM WebServices Listener and AXOM Command and Reporting are C++ library calls.

As the AXOM and Commands and Reporting are a set of libraries, we define a new function Workflow_Editor_Launcher within the AXOM_WebService_Listener Module to launch the Editor. Upon receipt of a request to launch the Axmedis Editor, the Workflow_Editor_Launcher will launch the editor using an ActiveX Call invoking the appropriate editor. This method will not return until the Editor is terminated and the ActiveX control comes back, after which it can notify the workflow manager for the completion of the activity.

As described before, the Axmedis Editor WorkFlow channel passes through the WF Editor Request Gateway where the AXOM_WebService_Listener exposes the following methods, via WebServices:

- Edit_Object, for launching the Axmedis Object Editor and all its Plug-ins used for editing and viewing Axmedis Objects, Object Behaviours, DRMs, Hierarchies and Metadata
- View_Object_Attribute to request the Axmedis Object Manager to allow the viewing of the object attributes
- Add_History_Info to request the Axmedis Object Manager to have a history information added to an object AXINFO
- Edit_Composition_Formatting_Rule for launching the Composition/Formatting Rule Editor
- Program_Publication_User_Interface for launching the Program/Publication User Interface

The method invocation is performed using a WebService request where the following parameters are sent (to AXOM Command and Reporting) and received back (in WebService result) from the AXOM Command and Reporting via AXOM_Web_Service_Listener:

2.2 Interface between the WF AXOM Request Adapter and the WF Editor Request Gateway

The methods invoked and the parameters sent by WF AXOM Request Adapter to the WF Editor Request Gateway are the same described in the preceeding Paragraph. There encoding, however, is different. The request in in fact sent through an http GET call where the paramteres are invoked as follows:

GET/Control_Panel/Products/OpenFlow/AXWF/**editor_name**_request_editor?AXOID=”Object ID string”&Credentials=”Credential string”&AXRQID=”Request ID string”&execution_parameters=”execution parameter string”&attribute_values=”attribute_name_1:attribute_value1,attribute_name2:attribute_value2:,etc ”&log_info=”log_info string” HTTP/1.1" 200 368 "http://15.156.120.195:8080/Control_Panel/Products/OpenFlow/leave/leave_startform" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)"

Where **editor_name** is to be replaced by appropriate editor identifier “Object ID string” is a string containing the AXOID, “Credential string” is a string containing the credentials and “Request ID string” is a string containing the Request ID.

The response to the invoked method has the same contents listed in the preceeding Paragraph and is sent via an http GET response. The response is XML coded, following the schema:

```
<xs:element name="Editor_Response">
	<xs:complexType>
		<xs:sequence>
			<xs:element name="result" type="xs:boolean"/>
			<xs:element name="errormsg" type="xs:string" nillable="true" minOccurs="0"/>
			<xs:element name="errorcode" type="xs:int"/>
			<xs:element name="AXOID" type="xs:string" minOccurs="0"/>
			<xs:element name="historylog" type="xs:string" minOccurs="0"
maxOccurs="100"/>
			<xs:element name="attributes" minOccurs="0" maxOccurs="20">
				<xs:complexType>
					<xs:sequence>
						<xs:element name="attributeid" type="xs:string"/>
						<xs:element name="attributevalue" type="xs:string"/>
					</xs:sequence>
				</xs:complexType>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
</xs:element>
```

The AXOM Command and Reporting will send its notifications to the WorkFlow Engine via AXOM_WebService_Listener which will call a WebServices exposed by the WF Editor Response Gateway .

The URI of the WebService is indicated in the previous request made by the WF Request Gateway in the EditorListenerService parameter.

The Notification also contain the original Request ID issued in the request (AXRQID).

7.3 Integration Support with AXEPTools

The workflow engine interacts with the following engines of AXEPTool:

- AXEPTool Loading Tool Engine
- AXEPTool Publication Tool Engine

As part of refinements, all the Axmedis engines were unified to form a single interface towards Axmedis Engines, the integration with AXEPTool is exactly same as that of content processing engines.

The integration is realised as per the specification document. The integration involves two distinct channels for workflow communication: Request Channel and Response Channel.

The Request Channel consists of Workflow Request Adaptors, Workflow Request Gateways and Webservice Listeners, while the Response channel consists of Response Gateways and Input Queue Adaptors.

The Axmedis Engine WorkFlow channel passes through the WF Engine Request Gateway where the Engine Command and Reporting exposes the following methods, via WebServices:

- Install_and_activate for installing a XML rule in the scheduler and activate it. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Run_rule for immediately run a rule already loaded inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Activate_rule for activating a rule already loaded inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- deactivate_rule for disabling a not-running rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Suspend_rule for suspending a rule for a specified time interval. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Pause_rule, for suspending a rule until it will be restarted. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Kill_rule for stopping the execution of a rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Remove_rule for removing a rule from the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Resume_rule for resuming a paused rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_rule_status for getting the status of a rule inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_rule_logs for getting history log of a rule. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_list_of_rules for getting the list of the rules of a certain user inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Get_rule for getting the XML definition of a rule inside the scheduler. This method is valid for the Compositional/Formatting engine, for the AxepTool Loading and Publication Engine and for the Protection Engine.
- Status_request_to_PnP for getting the status of a Program of the Program and Publication Engine
- Suspend_PnP_Program for suspending a Program of the Program and Publication Engine
- Abort_PnP_Program for aborting a Program of the Program and Publication Engine
- Resume_PnP_Program for resuming a suspended Program of the Program and Publication Engine
- Activate_PnP_Program for acrivating a Program of the Program and Publication Engine
- WorFlow_Notification is used to return back to the requesting engine (basically PnP) the status about the requested execution of a WorFlow process

The method invocation is performed via a WebService request where the parameters are sent (to Engine Command and Reporting) and received back (in WebService result) from the Engine Command and Reporting

The methods invoked and the parameters invoked by WF Engine Request Adapter to the WF Engine Request Gateway are the same described in the preceeding Paragraph. Their encoding, however, is different. The request in in fact sent through an http GET call where the paramteres are invoked as follows:

As an example to understand the response, consider the compositional/formatting engine response to a request.

GET/Control_Panel/Products/OpenFlow/AXWF/comp_format_request_status?Credentials=”Credential string”&AXRQID=”Request ID string” HTTP/1.1" 200 368 "http://15.156.120.195:8080/Control_Panel/Products/OpenFlow/leave/leave_startform" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)"

Where “Credential string” is a string containing the credentials and “Request ID string” is a string containing the Request ID.

The response to the invoked method has the same contents listed in the preceeding Paragraph and is sent via an http GET response. The response is XML coded, following the schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:element name="Engine_Response">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="result" type="xs:boolean"/>
				<xs:element name="errormsg" type="xs:string" nillable="true" minOccurs="0"/>
				<xs:element name="errorcode" type="xs:int"/>
				<xs:element name="ruleid" type="xs:string" minOccurs="0" maxOccurs="20"/>
				<xs:element name="status" type="xs:string" minOccurs="0"/>
				<xs:element name="xml_rule_schema" type="xs:string" minOccurs="0"/>
				<xs:element name="rulelog" type="xs:string" minOccurs="0" maxOccurs="100"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

As desdcribed before, the Engine Command and Reporting will send its notifications to the WorkFlow Engine by calling a WebServices exposed by the WF Engine Response Gateway.

The URI of the WebService is indicated in the previous request made by the WF Request Gateway in the EngineListenerService parameter.

The Notification also contain the original Request ID issued in the request (AXRQID).

7.4 Integration Support with Query Support

The integration is realised as per the specification document. The integration involves two distinct channels for workflow communication: Request Channel and Response Channel.

The Request Channel consists of Workflow Request Adaptors, Workflow Request Gateways and Webservice Listeners, while the Response channel consists of Response Gateways and Input Queue Adaptors.

The Axmedis DB WorkFlow channel passes through the WF DB Request Gateway where the Loader/Saver and the Query Support WebServices Interface modules will expose the following methods, via WebServices:

- Delete_selection for removing a selection from selection DB
- Load_selection for getting a selection from selection DB
- Save_selection for storing a selection in selection DB
- List_user_selection for listing the current selections in DB associated to the user
- List_entitled_selections for listing the current selections in DB that the user is entitled to execute
- Activate_selection_sync for activating a selection and waiting its completion
- Activate_selection_async for activating a selection and getting completion notification later

- Check_out_sync for checking-out an Object (MPEG-21 file) from Axmedis DB and waiting the completion of the operation
- Check_out_async for checking-out an Object (MPEG-21 file) from Axmedis DB and getting completion notification later
- commit_sync for checking-in an Object (MPEG-21 file) to Axmedis DB
- commit_async for checking-in an Object (MPEG-21 file) to Axmedis DB and getting completion notification later
- lockObject for locking an object for a user in the AXDB
- unlockObject for unlocking of an object for a user in the AXDB

The method invocation is encoded in an http GET request that contains both the method and the INPUT parameters. The related GET response will encode the OUTPUT parameters:

The methods invoked and the parameters invoked by WF DB Request Adapter to the WF DB Request Gateway are the same described in the preceeding Paragraph. Their encoding, however, is different. The request in in fact sent through an http GET call where the paramteres are invoked as follows:

As an example to understand the response, consider the compositional/formatting engine response to a request.

GET/Control_Panel/Products/OpenFlow/AXWF/WFDB_request_status?Selection_ID=”Selection ID string”&Credentials=”Credential string”&AXRQID=”Request ID string”&Path=”path string” HTTP/1.1" 200 368 "http://15.156.120.195:8080/Control_Panel/Products/OpenFlow/leave/leave_startform" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)"

Where WFDB is the workflow database, “Selection ID string” is a string containing the AXOID, “Credential string” is a string containing the credentials and “Request ID string” is a string containing the Request ID and “path string” contains the designated pathname to get the MPEG-21 file.

The response is XML coded, following the schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="Database_Response">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="result" type="xs:boolean"/>
				<xs:element name="errormsg" type="xs:string" nillable="true"
minOccurs="0"/>
				<xs:element name="errorcode" type="xs:int"/>
				<xs:element name="AXOID" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
				<xs:element name="XML_Selection" minOccurs="0">
					<xs:complexType>
						<xs:simpleContent>
							<xs:extension base="xs:string"/>
						</xs:simpleContent>
					</xs:complexType>
				</xs:element>
				<xs:element name="Selection_ID" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
				<xs:element name="Version" type="xs:string" minOccurs="0"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

The Notification is used, as mentioned, to return to the WorkFlow the results of the requested operation. Specifically, it contains:

- Edit_Query for notifying the termination of the editor:
  - NOTIFICATION: AXRQID, Completion_Result (OK, EXCEPTION)
- Activate_Selection_async for notifying the completion of the search:
  - NOTIFICATION: AXRQID, Completion Result (list of selected objects, EXCEPTION)

  List_of_selected_objects is the list of AXOIDs selected in the Query
- Check_out_async:
  - NOTIFICATION: AXRQID, Completion_Result (OK, EXCEPTION)
- commit_async
  - NOTIFICATION: AXRQID, Completion_Result (version, EXCEPTION)

  Version is the version of the loaded object

Where the AXRQID is the AXRQID in the original request from the WorkFlow and Completion_result it can be either positive (OK or returned parameters) or negative (EXCEPTION is an error code returned for diagnostic purposes and useful for troubleshooting).

The NOTIFICATION is sent via an XMLRPC call whose XML encoding is specified:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Database_Notification">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="result" type="xs:boolean"/>
                <xs:element name="errormsg" type="xs:string" nillable="true"
minOccurs="0"/>
                <xs:element name="errorcode" type="xs:int"/>
                <xs:element name="AXRQID" type="xs:string"/>
                <xs:element name="AXOID" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Version" type="xs:string" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Where WFDB is the workflow database and AXRQID is a string containing the the original request sent to the Engine.

Result is a positive integer when the request was successfully completed, or negative integer otherwise.

Status is a string containing the newly created AXOID if result is positive, otherwise a string containing the returned error.

7.5 Integration Support with AXMEDIS P&P Editor

The integration is realised as per the specification document. The integration involves two distinct channels for workflow communication: Request Channel and Response Channel. This integration is exactly same as for the Axmedis Editors as mentioned above.

The Request Channel consists of Workflow Request Adaptors, Workflow Request Gateways and Webservice Listeners, while the Response channel consists of Response Gateways and Input Queue Adaptors.

The following functions are available for P&P Editor.

- Axmedis Program and Publication User Interface

The Axmedis Rule Editor WorkFlow channel passes through the WF Rule Editor Request Gateway where the User Command and Reporting will expose the following methods, via WebServices:

- Program_Publication_User_Interface for launching the Program/Publication User Interface

The method invocation is performed via a WebService request where the following parameters are sent (to User Command and Reporting) and received back (in WebService result) from the User Command and Reporting.

The methods invoked and the parameters invoked by WF Rule Editor Request Adapter to the WF Rule Editor Request Gateway are the same described in the preceding Paragraph. Thier encoding, however, is different. The request is in fact sent through an http GET call where the parameters are invoked as follows:

GET/Control_Panel/Products/OpenFlow/AXWF/**rule_editor_name**_request_status?AXRID=”Rule ID string”&Credentials=”Credential string”&AXRQID=”Request ID string” HTTP/1.1" 200 368 "http://15.156.120.195:8080/Control_Panel/Products/OpenFlow/leave/leave_startform" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)"

Where **rule_ditor_name** is to be replaced by appropriate editor identifier “Rule ID string” is a string containing the AXRID, “Credential string” is a string containing the credentials and “Request ID string” is a string containing the Request ID.

The response to the invoked method has the same contents listed in the preceeding Paragraph and is sent via an http GET response. The response is XML coded, following the schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:element name="Rule_editor_Response">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="result" type="xs:boolean"/>
				<xs:element name="errormsg" type="xs:string" nillable="true" minOccurs="0"/>
				<xs:element name="errorcode" type="xs:int"/>
				<xs:element name="programid" type="xs:string" minOccurs="0" maxOccurs="20"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

As desdcribed before, the User Command and Reporting sends its notifications to the WorkFlow Engine by calling a WebServices exposed by the WF Rule Editor Response Gateway.

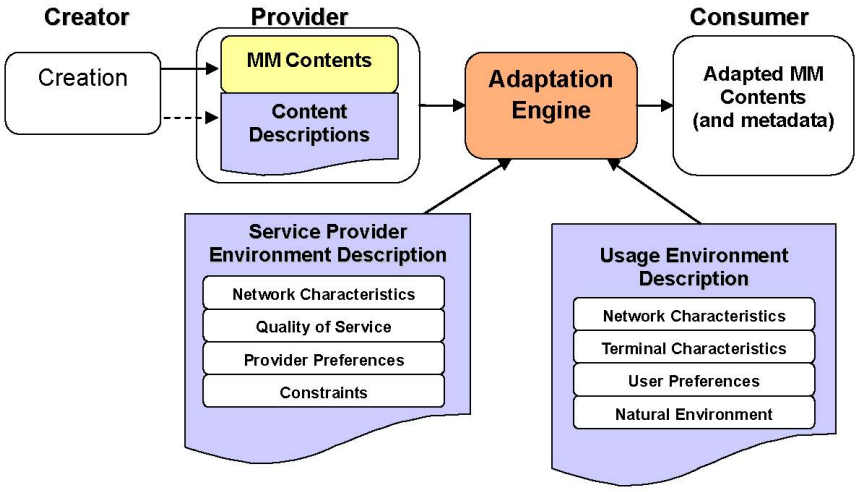
The URI of the WebService is indicated in the previous request made by the WF Request Gateway in the UserListenerService parameter.

The Notification will also contain the original Request ID issued in the request (AXRQID).

8 Transcoding and Adaptation (FHGIGD) (completed)

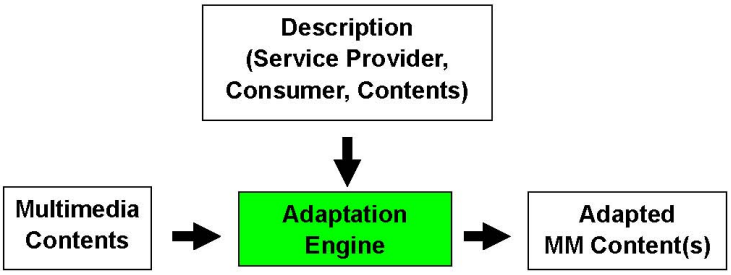
Nowadays, the users demand is developing into one direction: users want to access any multimedia content over any network type with any device from anywhere at anytime. This requirement is considered by the Universal Multimedia Access (UMA, cf. [1], [2] and [3]). As described in [1], "UMA is a key framework for multimedia content delivery service using metadata."

In [1] the adaptation between provider and consumers are identified as a major problem: for the end users, the quality of service as well as user experience has to be maximized. The task of the adaptation engine is to "bridge the gap between media format and terminal, network, user and provider characteristics" [1]. The central role of the adaptation engine in the UMA framework is shown in the next figure.



Central role of the Adaptation Engine within UMA (source: [1]).

As shown in the next figure, the adaptation engine according to UMA adapts content depending on the input content and the available description of the service providers, of the consumers and of the content.



Content adaptation framework in UMA systems. (source [1])

Examples for the different parameters for content adaptation are:

Media formats, Bitrate	Quality of Service	Access Network (Bandwidth)
Spatial resolution	Available Bandwidth	Display resolution / color
Temporal resolution	Error rate	Memory / CPU / Decoders
Number of Colors	Constraints	User preference
Limitations, rights	Delay	Access location, time

According to [1], the adaptation engine can have different functionalities:

- Transformation engine
- Variation selection engine
- Content selection engine
- Content and variation selection engine

This approach is considered within AXMEDIS as seen in the previous sections. MPEG-21 Part 7 Digital Item Adaptation (DIA), tools in MPEG-7 Part 5 Multimedia Description Scheme (MDS) are relevant for adaptation. These standards address a major part of the multimedia content adaptation process. Details of the AXMEDIS architecture are described in chapter 3 ("AXMEDIS Architecture for Content Processing").

In this section, the focus is on transcoding and adaptation, which is the "transformation engine" according to the UMA terminology.

General issues which are relevant for the transcoding and adaptation of content include (see [1]):

- Adaptation system architecture: adaptation on server-side vs. adaptation on client-side vs. proxy-based adaptation.
- Storage location of usage environment descriptions: storage on server side, storage on client side, storage on intermediate server
- Privacy of personal data, of personal communication, of the person and of personal behaviour.

[1] E, K. & T, E. New Frontiers in Universal Multimedia Access EPFL, 2004
[2] Sanjuan, D.M.; Steiger, O. & Ebrahimi, T. Design and Implementation of a Universal Multimedia Access Environment 2002
[3] Steiger, O.; Sanjuán, D.M. & Ebrahimi, T. MPEG-based personalized content delivery. 2003, 45-48

9 Transcoding Audio (EPFL)

9.1 Audio: State of the art

Content is generally created in a single format and can thus only be consumed by a limited set of users. Content adaptation provides mechanisms that allow formats to be interchanged such that the content may be delivered in other formats than that in which it was originally created. For example, it is common practice for radio stations to adapt the audio signal to be transmitted according to the channel characteristics.

With the explosion in availability of digital audio content and the advent of audio transmission over the Internet, adaptation, and particularly techniques that maximize quality output given a set of constraints, has gained a new dimension. Multimedia applications are increasingly used in mobile networks as well as in cable and fixed networks. Small devices with low performance have to be supported, as well as home entertainment equipment with high quality and multi channel audio.
To allow for transparent audio distribution, an adaptation tool needs to be able to perform the following three transformations:

1. Format and codec conversion
2. Sampling rate conversion
3. Channel mix-down

These transformations, though they may alter the quality of the signal, do not aim at modifying the content explicitly. On the contrary, digital audio effects aim at transforming the audio signal more radically. Transformations such as equalization or reverberation are widely used by music producers but also by music listeners since many hi-fis propose such settings. More complex effects such as time-stretching are used to synchronise easily a soundtrack or dialogues with a movie. Fading effects are used when one needs to shorten a sound excerpts (to build a small sample clip from a complete song for example).

9.2 Audio: The problems

The main problems encountered concern the integration of libraries coming from different sources so that special care has to be taken to allow producing a multi-platform tool for transcoding. Other libraries will be integrated to allow manipulating more file formats (libfaac and libfaad to use AAC files and libvorbis and libogg to use ogg files),

9.3 Audio: Work performed

The audio adaptation functionalities have been integrated with the axeditor as plug-ins through the AXCP interface. The plug-in simply consists of a DLL and an XML file describing the functionalities of the DLL. Both the DLL and the XML description are installed in the plug-in directory of the AXCP compliant tool using the plug-in. The FFMPEG library and the LIBSNDFILE library have been used for the audio transcoding functionalities.

9.3.1 FFmpeg Audio Transcoding

The FFmpeg Audio Transcoding function are used to convert an audio file into a different format and/or codec. Apart from the bit rate reduction depending on the selected codec, one can further reduce the size of the resulting audio file by changing its sample rate and its number of audio channels. Moreover one can select only a specific portion of the input file to produce the resulting output file by specifying starting and ending points in the input file.

9.3.1.1 Formal description of FFmpeg transcoding algorithm

Description: encode an audio file in another format or another codec and change its sample rate and number of audio channels if needed.

The generic syntax is:

string Trancoding(*AxResource* InputResource, *string* MimeType, *AxResource* OutputResource, *UINT32* OutputSamplingRate, *UINT16*, OuputNumChannels, *UINT16* OutputBitRate, *float* ReadStartingTime, *float* ReadEndingTime, *string* OutputCodec)

Parameter List:

Name: InputResource
Description: the resource to be converted
Parameter Type: *AxResource*
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg (.mp3), x.aiff (.aif, .aiff), x-wav (.wav), basic (.au, .snd), x-ms-wma (.wma), x-vorbis (.ogg), x-pn-realaudio (.ra, .ram)
Ranges:

Name: MimeType
Description: MimeType for the output resource
Parameter Type: *string*
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg, x-aiff, x-wav, basic, x-vorbis, x-ac3
Ranges:

Name: OutputResource
Description: Where the output resource will be stored
Parameter Type: *AxResource*
Default Value:
Constraints:
Range:

Name: OutputSamplingRate
Description: The sampling rate of the output resource in Hertz
Parameter Type: *uint32*
Default Value: by default, the sampling rate of the input resource is used
Constraints:
Range:

Name: OutputNumChannels
Description: The number of channels of the audio resource after transcoding
Parameter Type: *uint16*
Default Value: by default, the number of channels of the input resource is used
Constraints:

Range:

Name: OutputBitRate
Description: The bit rate of the audio resource after transcoding in kilo-Bytes (this parameter is used when transcoding towards a compressed audio format such as MP3)
Parameter Type: *uint16*
Default Value: by default, the bit rate is set to 64 kB
Constraints:
Range:

Name: ReadStartingTime
Description: set the beginning of the output resource to *ReadStartingTime* seconds from the beginning of the input resource
Parameter Type: *float*
Default Value: by default, the read starting time is set to 0 seconds which means that the input resource is considered from the beginning
Constraints:
Range:

Name: ReadEndingTime
Description: set the end of the output resource at *ReadEndingTime* seconds from the beginning of the input resource
Parameter Type: *float*
Default Value: by default, the read ending time is set to the end of the input resource
Constraints:
Range:

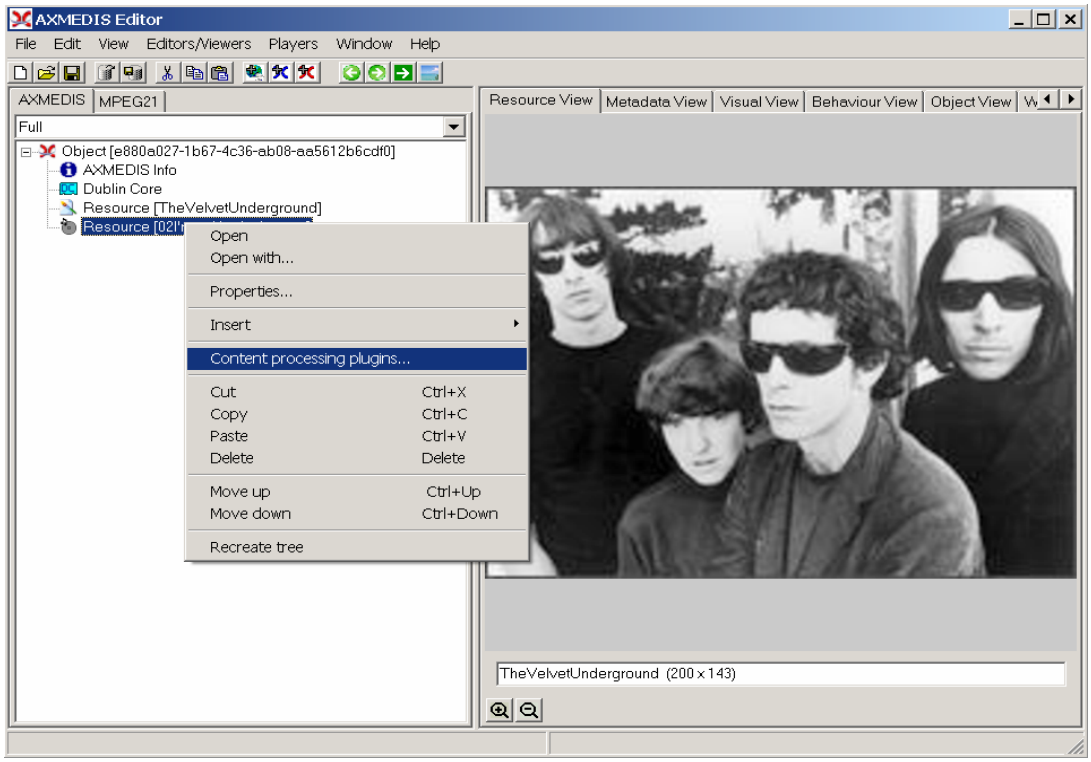
Name: OutputCodec
Description: set the codec of the output resource; depending on the mime type selected for the output resource, only a certain subset of codec will be supported (the following table shows the possible codecs according to the possible mime types)
Parameter Type: *string*
Default Value: the default codec depend on the mime type selected for the output resource (the following table shows the default codec according to the possible mime types)
Constraints:
Range:

Result: Result
Result Type: string
Result Description: the result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

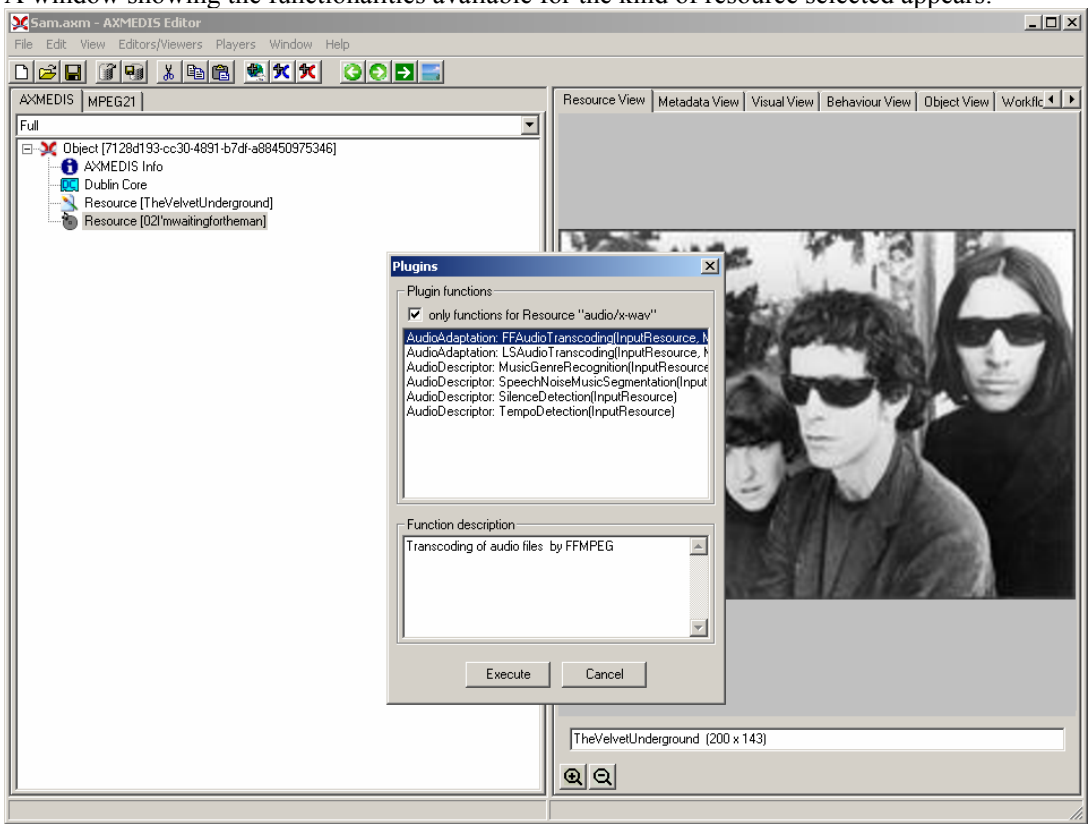
9.3.1.2 FFmpeg Audio Transcoding Plug-in

The following demonstrates the FFmpeg audio adaptation transcoding function as a plug-in integrated with for the AXMEDIS editor.

The plug-in must be applied on an audio resource of an AXMEDIS object. The adaptation plug-in is called by right-clicking on the interesting resource and selecting the 'Plugin…' command:



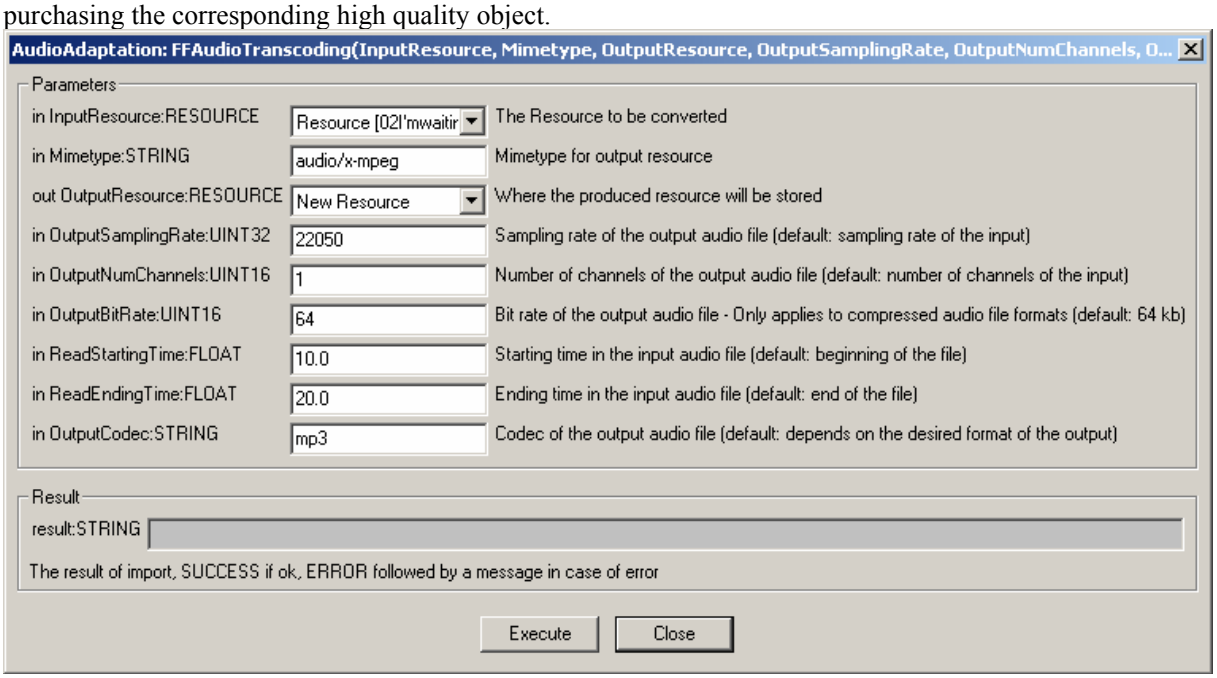
A window showing the functionalities available for the kind of resource selected appears:



The first audio adaptation function available is the FFmpeg transcoding function which is selected by clicking on **FFAudioAdaptation: FFAudioTranscoding**. A new window appears showing the interface to the audio transcoding function. In the example of the following figure, the transcoding function is used to create a 10 second snapshot with reduced bit rate of the input audio file:

- Mp3 compression is selected with a bit rate of 64 kB (which corresponds to a low quality)
- Further bit rate reduction is achieved by using a lower sampling rate for the output (22050 Hz) and mixing audio channels into a single mono channel
- Only a portion of 10 seconds of the input resource is selected (starting at time 10 seconds and ending at time 20 seconds)

A snapshot with reduced bit rate is particularly useful to allow a customer to pre-view an item before purchasing the corresponding high quality object.



9.3.2 Libsndfile

Libsndfile Audio Transcoding

The libsndfile Audio Transcoding function are used to convert an audio file into a different format and/or codec. Apart from the bit rate reduction depending on the selected codec, one can further reduce the size of the resulting audio file by changing its sample rate and its number of audio channels. Moreover one can select only a specific portion of the input file to produce the resulting output file by specifying starting and ending points in the input file.

9.3.2.1 Formal description of libsndfile transcoding algorithm

Description: encode an audio file in another format or another codec and change its sample rate and number of audio channels if needed.

Signature:

string Trancoding(*AxResource* InputResource, *string* MimeType, *AxResource* OutputResource, *float* ReadStartingTime, *float* ReadEndingTime, *string* OutputCodec)

Parameter List:

Name: InputResource
Description: the resource to be converted
Parameter Type: *AxResource*
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg (.mp3), x.aiff (.aif, .aiff), x-wav (.wav), basic (.au, .snd), x-ms-wma (.wma), x-vorbis (.ogg), x-pn-realaudio (.ra, .ram)
Ranges:

Name: MimeType
Description: MimeType for the output resource
Parameter Type: *string*
Default Value:
Constraints:
Resource Type: audio
Resource Format: x-mpeg, x-aiff, x-wav, basic, x-vorbis, x-ac3
Ranges:

Name: OutputResource
Description: Where the output resource will be stored
Parameter Type: *AxResource*
Default Value:
Constraints:
Range:

Name: ReadStartingTime
Description: set the beginning of the output resource to *ReadStartingTime* seconds from the beginning of the input resource
Parameter Type: *float*
Default Value: by default, the read starting time is set to 0 seconds which means that the input resource is considered from the beginning
Constraints:
Range:

Name: ReadEndingTime
Description: set the end of the output resource at *ReadEndingTime* seconds from the beginning of the input resource
Parameter Type: *float*
Default Value: by default, the read ending time is set to the end of the input resource
Constraints:
Range:

Name: OutputCodec
Description: set the codec of the output resource; depending on the mime type selected for the output resource, only a certain subset of codec will be supported (the following table shows the possible codecs according to the possible mime types)

Parameter Type: *string*
Default Value: the default codec depend on the mime type selected for the output resource (the following table shows the default codec according to the possible mime types)
Constraints:
Range:

Result: Result
Result Type: string
Result Description: the result of conversion, SUCCESS if ok, ERROR followed by a message in case of error

Libsndfile supported types and codecs:

For a list of codecs and formats supported by the Libsndfile library, please refer to section 34.5.

Mime Type accepted :

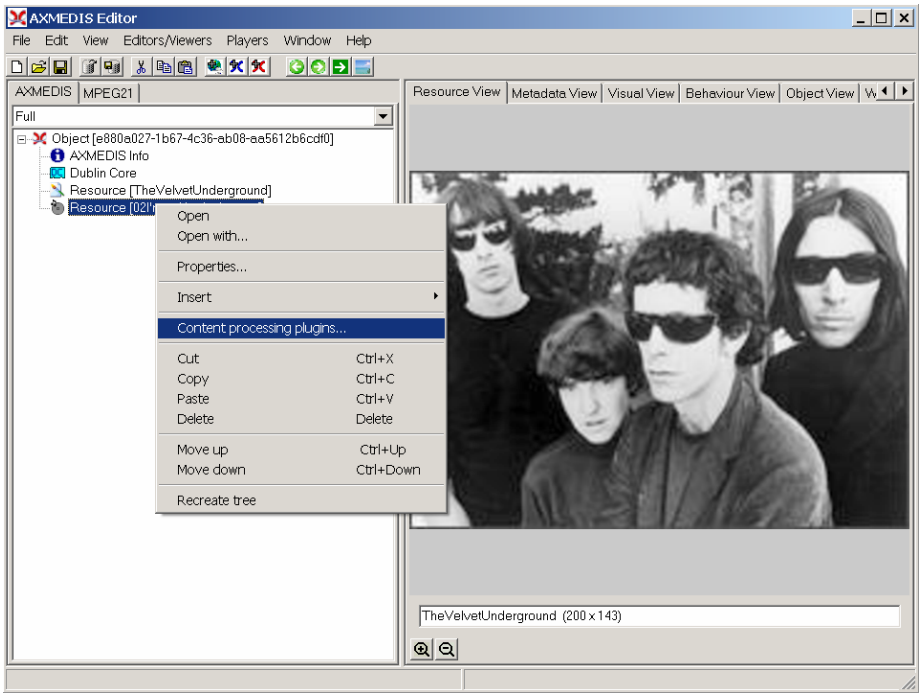
```
audio/x-wav
audio/x-basic
audio/x-paris
audio/x-svx
audio/x-nist
audio/x-voc
audio/x-ircam
audio/x-w64
audio/x-sd2
audio/x-flac
application/x-pcm
application/x-pagerecall
```

Here's an example on how to use the libsndfile audio adaptation transcoding function as a plug-in with for the AXMEDIS editor.

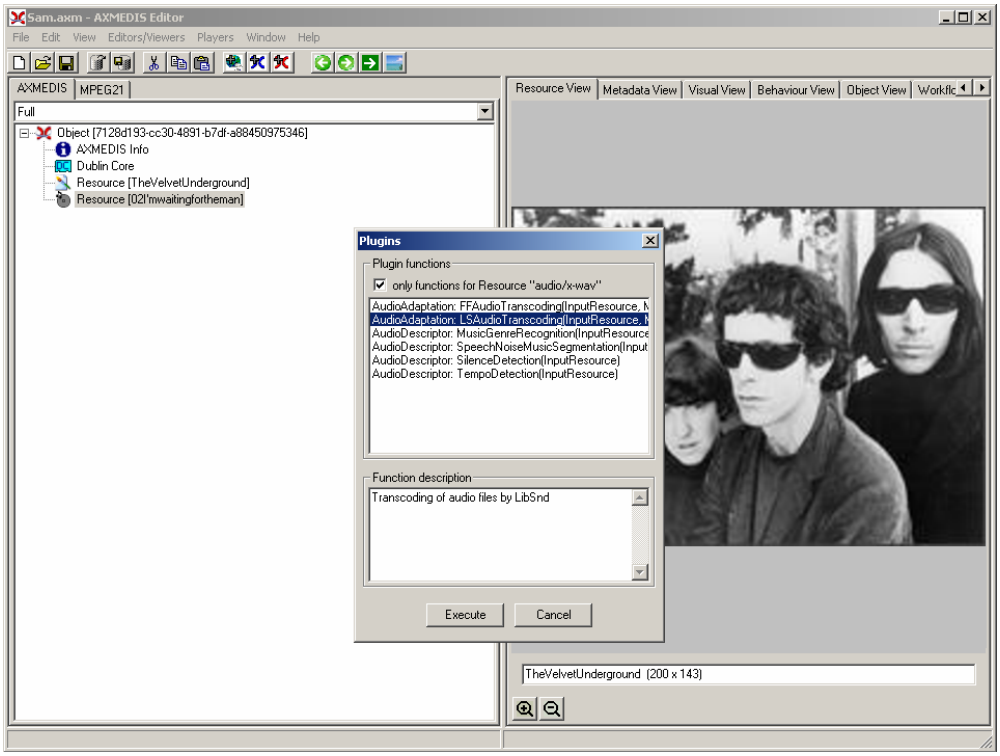
9.3.2.2 Libsndfile AudioTranscoding Plug-in

The following demonstrates the libsndfile audio adaptation transcoding function as a plug-in integrated with for the AXMEDIS editor.

The plug-in must be applied on an audio resource of an AXMEDIS object. The adaptation plug-in is called by right-clicking on the interesting resource and selecting the 'Plugin…' command:



A window showing the functionalities available for the kind of resource selected appears:

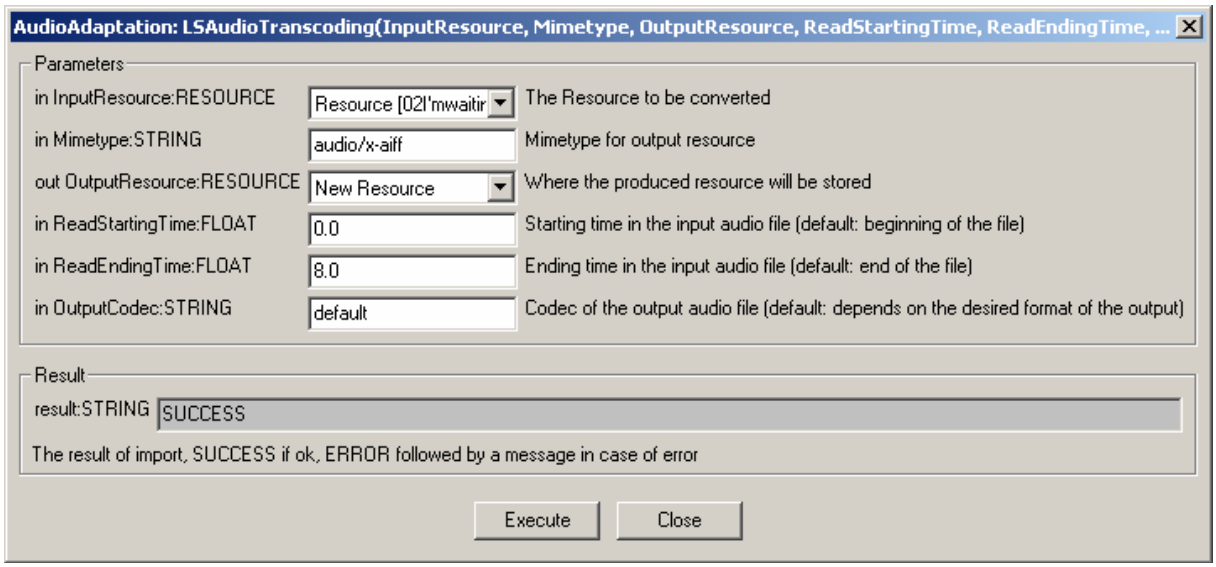


The first audio adaptation function available is the libsndfile transcoding function which is selected by clicking on **LSAudioAdaptation: LSAudioTranscoding**. A new window appears showing the interface to

the audio transcoding function. In the example of the following figure, the transcoding function is used to create a 10 second snapshot with reduced bit rate of the input audio file:

- AIFF format
- Only a portion of 8 seconds of the input resource is selected (just the beginning of the sound track)

Such a snapshot could be useful for small audio sampling.



Major Features accessible:

- Audio transcoding with FFMEPG library
- Audio transcoding with Libsndfile library

Work to be done:

closed

10 Transcoding Video (FHGIGD)

10.1 Video: State of the art

Chang et al. [Chang05] identified active research areas of video adaptation. These include:

- Semantic event-based adaptation: Semantic events in the video are the basis for this kind of adaptation. Depending on the video content and the target audience and the usage of the video content the adaptation varies. For example, in a soccer game only the key (scoring) scenes might be extracted in a video. The basic required semantic information can also be extracted from other source than from the video, e.g. from the corresponding audio [Chang01, Zhang02, Zhong01].
- Structural-Level Adaptation is a simpler approach than semantic event-based adaptation, which is based on the structure of video. An example is the extraction of key-frames that summarize the video content best [Irani98].
- Transcoding is on a signal level. For example, spatial resolution, precision or temporal properties (e.g. frame rate) are adapted according to the possibilities of the transmission channel and the receiving device [Vetro03].
- Rapid Fast-Forward can be compared with the fast-forward feature of a VCR player. The content is temporally "condensed" [Smith95, Smith97, Lou04].

10.2 Video: The problems

Different open issues have been identified in the literature, e.g. in [Vetro05] and [Chang05]. Among these open issues are:

- Permissible and secure adaptation: The two aspects which are comprised here are the permission for specific adaptations and the secure adaptation of encrypted objects.
- Adaptation in Constrained and Streaming Environment: For some devices or scenarios limited hardware capabilities have to be considered. This influences the adaptation, e.g. the conversion of adaptation parameters or the adapted content.
- Transport, Negotiation and Exchange of Descriptors: The adaptation parameters have to be transported, negotiated or exchanged. This has to be handled in a standardized manner.
- Semantic Clues for Adaptation: Semantic information of the content that should be adapted can improve the result of the adaptation.
- Modality Conversion: Sometimes a modality conversion (e.g. from a video format to a image sequence) might be better suited. E.g. for certain devices/transmission channel there might be strong hardware limitations.
- Maximum User Experience: The previous aspects address technical issues. The ultimate criteria, however, is the user experience. This user experience has to be maximised.

These general aspects are also relevant for video adaptation. In addition to the previously described active research areas, which can also be considered as open issues, in [Chang05] specific open issues for video adaptation are also described:

- Definition of utility measure and user preferences: There is a general lack of a definition of adequate measures or methods for estimating the quality or the user's satisfaction of the video content ("utility").
- Resolving ambiguities in specifying adaptation operation: A verbal description of an adaptation operation is often ambiguous.

- Relations among adaptation, utility, and resource. The relationship between adaptation, utility, and resource is often complex (high dimensional space). The representation of this high dimensional space strongly depends on the application.
- Optimal solutions in large spaces are required to address the previously described problem of high-dimensional spaces. From a mathematical point of view, this is a constraint optimization problem.
- Design end-to-end integrated systems

10.3 FFMPEG executable

The work performed focused on the (integration of) transcoding algorithms. To show AXMEDIS' capabilities in terms of content adaptation as well as extensibility the FFMPEG library has been selected for the implementation/integration of the firstly available video adaptation functionality.

The via command line version of FFMPEG available video options are:

-b bitrate	set video bitrate (in kbit/s)
-r rate	set frame rate (Hz value, fraction or abbreviation)
-s size	set frame size (WxH or abbreviation)
-aspect aspect	set aspect ratio (4:3, 16:9 or 1.3333, 1.7777)
-croptop size	set top crop band size (in pixels)
-cropbottom size	set bottom crop band size (in pixels)
-cropleft size	set left crop band size (in pixels)
-cropright size	set right crop band size (in pixels)
-vn	disable video
-bt tolerance	set video bitrate tolerance (in kbit/s)
-maxrate bitrate	set max video bitrate tolerance (in kbit/s)
-minrate bitrate	set min video bitrate tolerance (in kbit/s)
-bufsize size	set ratecontrol buffere size (in kByte)
-vcodec codec	force video codec ('copy' to copy stream)
-sameq	use same video quality as source (implies VBR)
-pass n	select the pass number (1 or 2)
-passlogfile file	select two pass log file name

Besides the basic functionality so called "advanced options" are available:

-pix_fmt format	set pixel format
-g gop_size	set the group of picture size
-intra	use only intra frames
-qscale q	use fixed video quantiser scale (VBR)
-qmin q	min video quantiser scale (VBR)
-qmax q	max video quantiser scale (VBR)
-mbqmin q	min macroblock quantiser scale (VBR)
-mbqmax q	max macroblock quantiser scale (VBR)
-qdiff q	max difference between the quantiser scale (VBR)
-qblur blur	video quantiser scale blur (VBR)
-qcomp compression	video quantiser scale compression (VBR)
-rc_init_cplx	complexity initial complexity for 1-pass encoding
-b_qfactor factor	qp factor between p and b frames
-i_qfactor factor	qp factor between p and i frames
-b_qoffset offset	qp offset between p and b frames
-i_qoffset offset	qp offset between p and i frames
-rc_eq equation	set rate control equation

-rc_override	override rate control override for specific intervals
-me method	set motion estimation method
-dct_algo algo	set dct algo
-idct_algo algo	set idct algo
-er n	set error resilience
-ec bit_mask	set error concealment
-bf frames	use 'frames' B frames
-mbd mode	macroblock decision
-mbcmp	cmp function macroblock compare function
-ildctcmp	cmp function ildct compare function
-subcmp	cmp function subpel compare function
-cmp	cmp function fullpel compare function
-4mv	use four motion vector by macroblock (MPEG4)
-obmc	use overlapped block motion compensation (h263+)
-part	use data partitioning (MPEG4)
-bug param	workaround not auto detected encoder bugs
-strict strictness	how strictly to follow the standarts
-deinterlace	deinterlace pictures
-ildct	force interlaced dct support in encoder (MPEG2/MPEG4)
-ilme	force interlacied me support in encoder MPEG2
-psnr	calculate PSNR of compressed frames
-vstats	dump video coding statistics to file
-vhook module	insert video processing module
-aic	enable Advanced intra coding (h263+)
-aiv	enable Alternative inter vlc (h263+)
-umv	enable Unlimited Motion Vector (h263+)
-alt	enable alternate scantable (mpeg2)
-trell	enable trellis quantization
-scan_offset	enable SVCD Scan Offset placeholder
-intra_matrix	matrix specify intra matrix coeffs
-inter_matrix	matrix specify inter matrix coeffs
-top	top=1/bottom=0/auto=-1 field first
-nr	noise reduction

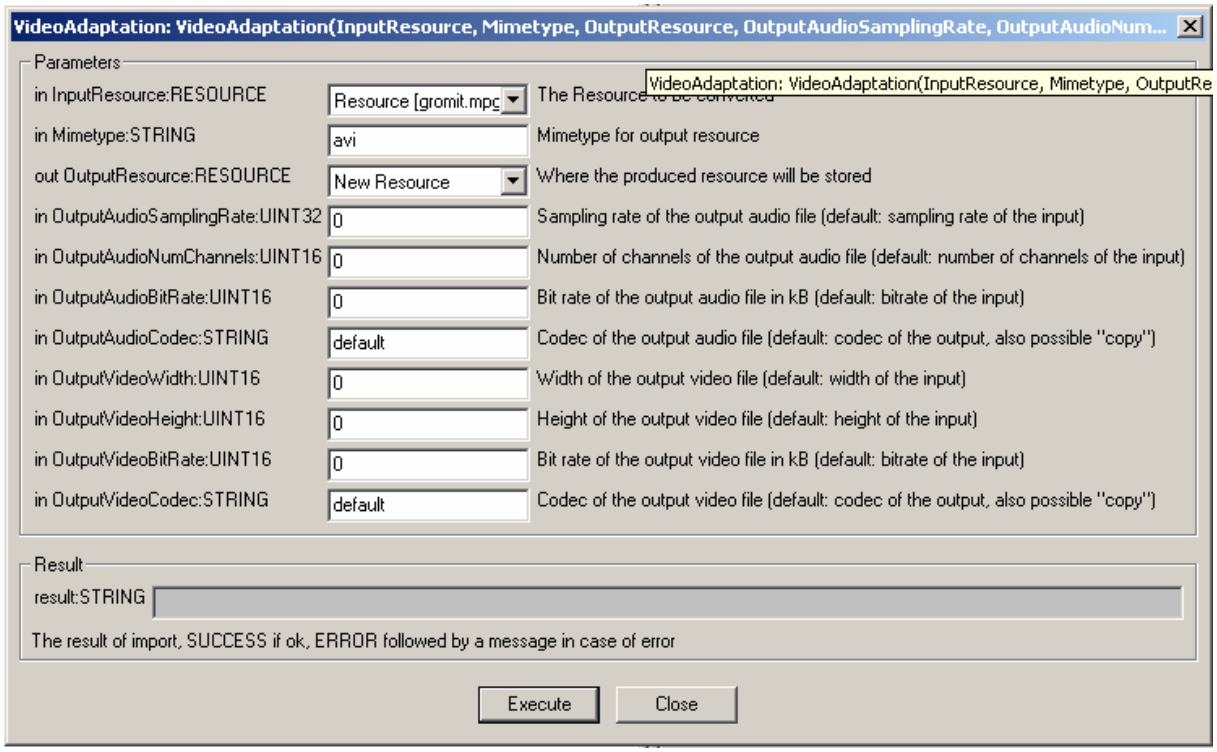
The functionality available within FFMPEG addresses two categories of video adaptation:

- Considering the (fixed) needs of the content creator or service providers: The content can be adapted according to the requirements of the content creator or service providers. The simplest case is the adaptation of the spatial resolution as supported by FFMPEG to a fixed resolution without changing the aspect ratio.

- Dynamic content creation: FFMPEG also allows changing the aspect ratio of the video content. Thus, a basic functionality required to dynamically adapt content according to the receiving device is available.

10.4 Video-Adaptation plug-in

Current version of the video adaptation plug-in implements a sub-set of the potential parameters as shown in the next diagram respectively as described below:



- AudioSamplingRate: Sampling rate of the output audio.
- AudioNumChannels: Number of Channels of the output audio file.
- AudioBitRate: Bit rate of the output audio file in kB.
- AudioCodec: Codec of the output audio file.
- VideoWidth; Width of the output video file.
- VideoHeight: Height of the output video file.
- VideoBitRate: Bit rate of the output video file in kB.
- VideoCodec: Codec of the output video file.

10.5 References

[Chang01] Chang, S.; Zhong, D. & Kumar, R. Real-Time Content-Based Adaptive Streaming of Sports Videos IEEE Computer Society, 2001, 139

[Chang05] Shih-Fu Chang Vetro, A. Video adaptation: concepts, technologies, and open issues 2005, 93, 148 – 158

[Irani98] Irani, M. & Anandan, P. Video indexing based on mosaic representation 1998

[Lou04] Luo, H. & Fan, J. Concept-oriented video skimming via semantic video classification ACM Press, 2004, 760-761

[Smith95] Smith, M. & Kanade, T. Video Skimming for Quick Browsing based on Audio and Image Characterization Computer Science Department, Carnegie Mellon University, 1995

[Smith97] Smith, M.A. Video Skimming and Characterization through the Combination of Image and Language Understanding Techniques IEEE Computer Society, 1997, 775

[Zhang02] Zhang, D. & Chang, S. Event detection in baseball video using superimposed caption recognition ACM Press, 2002, 315-318

[Zhong01] Zhong, D.; Kumar, R. & Chang, S. Real-time personalized sports video filtering and summarization ACM Press, 2001, 623-625

[Vetro03] Vetro, A.; Haga, T.; Sumi, K. & Sun, H. Object-based coding for long-term archive of surveillance video 2003, 2

[Vetro05] Vetro, A. & Timmer, C., Digital Item Adaptation: Overview of Standardization and Research Activities IEEE Transaction On Multimedia, 2005, 7, 418-426

11 Transcoding Documents and text (DIPITA)

The prototype for textual format conversion tool has to gather and integrate the different standard libraries for the conversion of the required formats. According to the AXMEDIS FW specifications (DE 3.1.2d), the formats treated within the conversion procedures are: plain text (TXT), Rich Text Format (RTF), Hyper Text Mark-up Language (HMTL), Portable Document Format (PDF), Post Script (PS).
Textual conversion is needed for:
- format adaptation tasks;
- perform keyword extraction on different format input files (see Humphreys 2002);
- perform content plagiarism evaluation (independently from the specific format of input file);
- guessing of the language of a document.

11.1 Documents and text: State of the art

Since textual conversion is a useful task for a lot of text processing applications, a huge variety of conversion libraries and software in order to get a plain text file from the most common formats has been developed (see Anjewierden, 2003). As a matter of fact, the conversion of RTF and HTML formats into plain text is a task that is completely satisfied by the modern technologies, and it represents a quite trivial issue. More problems emerge from the conversion procedures starting from PDF or PS format (see the next paragraphs for further details). Format adaptation is needed to pre-process document in order to deal with the main procedures of the AXMEDIS linguistic interface (keyword extraction and language guessing) and of the content protection and fingerprint. In the following section (7.1.3), the libraries to be exploited are presented.

Particular issue: conversion from PDF/PS document to plain text format
Most of the textual files associated to multimedia content production and documentation are in PDF format, because it is a platform-independent format, and because PDF files are relatively small in file size, allowing space saving on servers and repositories.
In different environment dealing with text pre-processing for information extraction issues, PDF format is used for storage and start point of the further conversion procedures. It is the case of CERN's solution for the automatic extraction of references from High Energy Physics documents and papers (to the aim of make them easy to access to physics researchers). The extraction procedure involve the whole repository of up to 170,000 electronic documents (available via the CERN Document Server; see Claivaz et al. 2001). For this task, PDF was deliberately chosen as a format from which extract the plain text document, due to its stability over other file formats such as PostScript or MSWord.
In Robinson (2001) there is a comparison study of these different tools with their advantages and drawbacks (pdftotext, ps2ascii.ps, pdftohtml, pdftohtml(.bin), pstotext, prescript). A lot of conversion tools were tested to find the most suitable one.
The primary flaw discovered was that none of the tools were able to convert PostScript documents that were produced by MSWord. Due to the increasing number of PostScript files submitted to the document archives and repositories that were actually produced from MSWord files, this is a major problem, as it means that none of these could be converted to text. This means that it is really necessary to perform the conversion from a PDF document, as it seems to be more stable. Tests were conducted for the conversion of PDF documents to text that were produced from PostScripts that were in turn produced from MSWord files. Problems were not encountered with these PDF file conversions.
Therefore, the best strategy for conversion of PS format into plain text seems to be a two stage conversion:
- first, convert the PS to PDF;
- second, covert the PDF obtained to a plain text.
As a result from the comparison among pdf2txt converters, the *pdftotext* tool from Foolabs (within XPDF tool-suite) was stated as the more reliable among the others (Robinson 2001).

11.2 Documents and text: The problems

Libraries to be exploited: general public license.
Because some of the used libraries are licensed under GPL, the conversion tool (pool of document converters) will be developed under GPL too. So the conversion tool will be accessed by the plug-in as a separated executable program, not as a library. The communication mechanism between the plug-in and the converter has to be defined.

Different output of format conversions
It is necessary to specify the particular features of the txt output files, depending on the different tasks in which textual format conversion are involved. The output textual files have to deal with:

1. Content fingerprint adaptation (within WP 4.2.2)
The main requirement to be satisfied by a fingerprint extractor is the independence from the file format. To solve this problem, the file of the original document is converted to a plain text file. As totally reliable format-conversion procedures are not available, to obtain the most similar fingerprint values from different format files it is needed, for example, to reduce the text file to a more stable one, by removing white space and punctuation, removing unusual characters (which can be easily mismatched by the converters), converting all letters to lower case, replacing variable names (in script files) by one string.

2. Language guessing adaptation (within WP 4.2.2)
Language guessing relies on statistics of character n-grams, extracted from documents of the languages to be treated. For this purpose, sequences of spaces and alphabetic character have to be exactly the same, even after document format conversion. Adaptation strategies such as the ones adopted in the fingerprint adaptation of text (which point to erase the uncertain converted characters) are unsuitable for this task.

3. Keyword extraction adaptation (within WP 4.2.2)
The most important unit for this task is not the text character (as in the previous ones), but the word. The conversion of text format for keyword extraction purposes has to respect exactly the word combinations in the context within they occurs. For this task, it is important to have a certain degree of reliability even in the treatment of punctuation.

11.3 Documents and text: Work performed

The following conversion libraries are the ones that will be used for text document conversion, as they obtain (according to the literature) the best performances compared to the other similar ones:

1. DOCFRAC (http://docfrac.sourceforge.net/), that allows conversions from RTF to HTML, from RTF to TXT, from HTML to RTF, from HTML to TXT, from TXT to RTF and from TXT to HTML.
The exploitation of the DOCFRAC features library will be particularly useful for converting active web pages, converting many documents at a time and converting output from Microsoft's Internet Explorer RTF control to HTML.
The supported platforms are Windows, Linux (command line) and programming kit (ActiveX and DLL). DocFrac is free. It is released under the LGPL.

2. GNU Ghostscript (http://www.cs.wisc.edu/~ghost/)
Ghostscript is the name of a set of software that contains an interpreter for the PostScript (TM) language and the Adobe Portable Document Format, and a set of C procedures (the Ghostscript library) that implement the graphics and filtering (data compression / decompression / conversion) capabilities that appear as primitive operations in the PostScript language and in PDF. Versions entitled "GNU Ghostscript" are distributed with the GNU General Public License.

3. XPDF (http://www.foolabs.com/xpdf/)
Xpdf is an open source viewer for Portable Document Format (PDF) files. The Xpdf project also includes a PDF text extractor, PDF-to-PostScript converter, and various other utilities.
Xpdf runs under the X Window System on UNIX, VMS, and OS/2. The non-X components (pdftops, pdftotext, etc.) also run on Win32 systems and should run on pretty much any system with a decent C++ compiler.
Xpdf is designed to be small and efficient. It can use Type 1, TrueType, or standard X fonts.
Xpdf is licensed under the GNU General Public License (GPL), version 2.

4. HTMLDOC (http://www.easysw.com/htmldoc/)
HTMLDOC converts HTML source files into indexed HTML, PostScript, or Portable Document Format (PDF) files that can be viewed online or printed.
The program is free software and is distributed under GPL.

5. PDF2HTML (http://pdftohtml.sourceforge.net/)
pdftohtml is a utility which converts PDF files into HTML and XML formats.
It's based on the xpdf 2.02 by Derek Noonburg.
The program is free software and is distributed under GPL.

6. a2ps (http://gnuwin32.sourceforge.net/packages/a2ps.htm)
GNU a2ps is an Any to PostScript filter. Of course it processes plain text files, but also pretty prints quite a few popular languages. Its slogan is precisely `` Do The Right Thing ", which means that though it is highly configurable, everything was made so that a novice user can do complicated PostScript manipulations. For instance, it has the ability to delegate the processing of some files to other filters (such as groff, texi2dvi, dvips, gzip etc.), what allows a uniform treatment (n-up, page selection, duplex etc.) of heterogeneous files. It supports a wide number of encodings, and a very good handling of Latin 2-6 should be noted, thanks to Ogonkify (by Juliusz Chroboczek). Needed fonts are automatically downloaded. The interface is internationalized, the output is customizable and there are as many options as users had wishes (table of content, headings, virtual page layout etc. etc.).
The program is free software and is distributed under GPL.

To test the first results of the adaptation tool, it has been created a test corpus for content adaptation. It contains several documents, each one in different textual format, and precisely:

a) Digital music report 2005 (a state of the art on digital music; formats: pdf, txt)
b) European constitution (the EU constitution in Italian, formats: pdf, rtf, txt)
c) Hibernate documentation (documentation on an object/relational mapping tool for Java environments; formats: pdf, html, txt)
d) Html specification (specification from the w3c on html formatting; formats: pdf, ps, html, txt).

References

Anjewierden, A. (2003) Document Analysis Component Library. University of Amsterdam

Claivaz, J.B., Le Meur, J.-Y., Robinson, N. (2001). From Fulltext Documents to Structured Citations. CERN-ETT-2001-003. Geneva. CERN.

Heintze, N. (1996), Scalable document fingerprinting. In *1996 USENIX Workshop on Electronic Commerce*.

Humphreys, J.-B. K. (2002) PhraseRate: An HTML Keyphrase Extractor. University of California

Robinson., N. (2001). A Comparison of Utilities for Converting from PostScript or Portable Document Format to Text. CERN-OPEN-2001-065. Geneva. CERN.

Websites

DOCFRAC (http://docfrac.sourceforge.net/),
GNU Ghostscript (http://www.cs.wisc.edu/~ghost/)
XPDF (http://www.foolabs.com/xpdf/)
HTMLDOC (http://www.easysw.com/htmldoc/)
PDF2HTML (http://pdftohtml.sourceforge.net/)
a2ps (http://gnuwin32.sourceforge.net/packages/a2ps.htm)

Major Features accessible:
Here's the table of possible transcodings. On the first column the starting format, on the first row the format to which a document has to be transcoded.
Cells contain the library/libraries which will be exploited for the transcoding.
The cells with grey shade show already implemented transcodings by the plugin provided on the AXMEDIS' SVN (https://cvs.axmedis.org/newrepos/Framework/bin/adaptation/document/)
Cells with yellow shade have been developed and will be available in short on the AXMEDIS' SVN.

	TXT	**RTF**	**HTML**	**PDF**	**PS**
TXT	x	DOCFRAC	DOCFRAC	a2ps + **Ghostscript**	a2ps
RTF	DOCFRAC	x	DOCFRAC	DOCFRAC + HTMLDOC	DOCFRAC + HTMLDOC
HTML	DOCFRAC	DOCFRAC	x	HTMLDOC	HTMLDOC
PDF	XPDF	PDF2HTML + DOCFRAC	PDF2HTML	x	XPDF
PS	**Ghostscript**	**Ghostscript** + PDF2HTML + docfrac	**Ghostscript** + PDF2HTML	**Ghostscript**	x

Work to be done:

All the rest of the transcodings have to be implemented exploiting all the functionalities of the libraries.
All libraries, except Ghostscript, have been integrated in mock-ups and evaluation programs and don't need further study or integration problems resolutions. Ghostscript is currently in the integration phase.
The communication system between the plug-in and the converter, in order to avoid GPL issues, has to be defined and implemented.

12 Transcoding Images (DSI, IRC)

12.1 Images : State of the art

Image adaptation tools provide functions which can be used for scaling, resolution improvements, text drawing, and image decomposition etc. of an image file. In the case of Image Objects, the following libraries can be used for implementing the functions described above.

a. ImageMagick Library

ImageMagick, version 6.2.3, is a free software suite designed to create, edit, and compose bitmap images. It can read, convert and write images in a large variety of formats. Images can be cropped, colors can be changed, various effects can be applied; images can be rotated and combined, and text, lines, polygons, ellipses and Bezier curves can be added to images and stretched and rotated.

Here are just a few examples of what ImageMagick can do:

i. Convert an image from one format to another (e.g. PNG to JPEG)
ii. Resize, rotate, sharpen, colour reduce, or add special effects to an image
iii. Create a montage of image thumbnails
iv. Create a transparent image suitable for use on the Web
v. Turn a group of images into a GIF animation sequence
vi. Create a composite image by combining several images
vii. Draw shapes or text on an image
viii. Decorate an image with a border or frame
ix. Describe the format and characteristics of an image

Major Features accessible:
A list of supported formats is given below. The documentation of the ImageMagick is available on http://www.imagemagick.org/.

Tag	Mode	Description	Notes
ART	R	PFS: 1st Publisher	Format originally used on the Macintosh (MacPaint?) and later used for PFS: 1st Publisher clip art.
AVIhttp://www.jmcgowan.com/avi.html	R	Microsoft Audio/Visual Interleaved	
AVS	RW	AVS X image	
BMPhttp://msdn.microsoft.com/library/sdkdoc/gdi/bitmaps_9c6r.ht	RW	Microsoft Windows bitmap	

m			
CGM	R	Computer Graphics Metafile	Requires ralcgm to render CGM files.
CIN	RW	Kodak Cineon Image Format	Cineon Image Format is a subset of SMTPE DPX.
CMYK	RW	Raw cyan, magenta, yellow, and black samples	Set `-size` and `-depth` to specify the image width, height, and depth.
CMYKA	RW	Raw cyan, magenta, yellow, black, and alpha samples	Set `-size` and `-depth` to specify the image width, height, and depth.
CUR	R	Microsoft Cursor Icon	
CUT	R	DR Halo	
DCM	R	Digital Imaging and Communications in Medicine (DICOM) image	Used by the medical community for images like X-rays.
DCX	RW	ZSoft IBM PC multi-page Paintbrush image	
DIB	RW	Microsoft Windows Device Independent Bitmap	DIB is a BMP file without the BMP header. Used to support embedded images in compound formats like WMF.
DPX	RW	Digital Moving Picture Exchange	
EMF	R	Microsoft Enhanced Metafile (32-bit)	Only available under Microsoft Windows.
EPDF	RW	Encapsulated Portable Document Format	
EPI	RW	Adobe Encapsulated PostScript Interchange format	Requires Ghostscript to read.
EPS	RW	Adobe Encapsulated PostScript	Requires Ghostscript to read.
EPS2	W	Adobe Level II Encapsulated PostScript	Requires Ghostscript to read.
EPS3	W	Adobe Level III Encapsulated PostScript	Requires Ghostscript to read.
EPSF	RW	Adobe Encapsulated PostScript	Requires Ghostscript to read.
EPSI	RW	Adobe Encapsulated PostScript Interchange format	Requires Ghostscript to read.
EPT	RW	Adobe Encapsulated PostScript Interchange format with TIFF preview	Requires Ghostscript to read.
FAX	RW	Group 3 TIFF	See TIFF format. Note that FAX machines use non-square pixels which are 1.5 times wider than they are tall but computer displays use square pixels so FAX images may appear to be narrow unless they are explicitly resized using a resize specification of "150x100%".
FIG	R	FIG graphics format	Requires TransFig.
FITS	RW	Flexible Image Transport System	
FPX	RW	FlashPix Format	Requires FlashPix SDK.
GIF	RW	CompuServe Graphics Interchange Format	8-bit RGB PseudoColor with up to 256 palette entires. Specify the format "GIF87" to write the older version 87a of the format.
GPLT	R	Gnuplot plot files	Requires gnuplot3.5.tar.Z or later.
GRAY	RW	Raw gray samples	Use `-size` and `-depth` to specify the image width, height, and depth.
HPGL	R	HP-GL plotter language	Requires hp2xx-3.2.0.tar.gz
HTML	RW	Hypertext Markup Language with a client-side image map	Also known as "HTM". Requires html2ps to read.
ICO	R	Microsoft icon	Also known as "ICON".
JBIG	RW	Joint Bi-level Image experts Group file interchange format	Also known as "BIE" and "JBG". Requires jbigkit-1.0.tar.gz.
JNG	RW	Multiple-image Network Graphics	JPEG in a PNG-style wrapper with transparency. Requires libjpeg and libpng-1.0.2 or later, libpng-1.2.5 or later

			recommended.
JP2	RW	JPEG-2000 JP2 File Format Syntax	Requires jasper-1.600.0.zip
JPC	RW	JPEG-2000 Code Stream Syntax	Requires jasper-1.600.0.zip
JPEG	RW	Joint Photographic Experts Group JFIF format	Requires jpegsrc.v6b.tar.gz
MAN	R	Unix reference manual pages	Requires that GNU groff and Ghostcript are installed.
MAT	R	MATLAB image format	
MIFF	RW	Magick image file format	Open ImageMagick's own image format (with ASCII header) which ensures that no image attributes understood by ImageMagick are lost.
MONO	RW	Bi-level bitmap in least-significant-byte first order	
MNG	RW	Multiple-image Network Graphics	A PNG-like Image Format Supporting Multiple Images, Animation and Transparent JPEG. Requires libpng-1.0.2 or later, libpng-1.2.5 or later recommended.
MPEG	RW	Motion Picture Experts Group file interchange format (version 1)	Requires mpeg2vidcodec v12.tar.gz.
M2V	RW	Motion Picture Experts Group file interchange format (version 2)	Requires mpeg2vidcodec v12.tar.gz.
MPC	RW	Magick Persistent Cache image file format	The native "in-memory" ImageMagick uncompressed file format. This file format is identical to that used by Open ImageMagick to represent images in memory and is read in "zero time" via memory mapping. The MPC format is not portable and is not suitable as an archive format. It is suitable as an intermediate format for high-performance image processing. The MPC format requires two files to support one image. When writing the MPC format, a file with extension ".mpc" is used to store information about the image, while a file with extension ".cache" stores the image pixels. The storage space required by a MPC image (or an image in memory) may be calculated by the equation (5*QuantumDepth*Rows*Columns)/8.
MSL	RW	Magick Scripting Language	MSL is the XML-based scripting language supported by the conjure utility.
MTV	RW	MTV Raytracing image format	
MVG	RW	Magick Vector Graphics.	The native ImageMagick vector metafile format. A text file containing vector drawing commands accepted by convert's -draw option.
OTB	RW	On-the-air Bitmap	
P7	RW	Xv's Visual Schnauzer thumbnail format	
PALM	RW	Palm pixmap	
PBM	RW	Portable bitmap format (black and white)	
PCD	RW	Photo CD	The maximum resolution written is 768x512 pixels since larger images require huffman compression (which is not supported).
PCDS	RW	Photo CD	Decode with the sRGB color tables.
PCL	W	HP Page Control Language	For output to HP laser printers.
PCX	RW	ZSoft IBM PC Paintbrush file	
PDB	RW	Palm Database ImageViewer Format	
PDF	RW	Portable Document Format	Requires Ghostscript to read.
PFA	R	Postscript Type 1 font (ASCII)	Opening as file returns a preview image.
PFB	R	Postscript Type 1 font (binary)	Opening as file returns a preview image.
PGM	RW	Portable graymap format (gray	

		scale)	
PICON	RW	Personal Icon	
PICT	RW	Apple Macintosh QuickDraw/PICT file	
PIX	R	Alias/Wavefront RLE image format	
PNG	RW	Portable Network Graphics	Requires libpng-1.0.2 or later, libpng-1.2.5 or later recommended.
PNM	RW	Portable anymap	PNM is a family of formats supporting portable bitmaps (PBM) , graymaps (PGM), and pixmaps (PPM). There is no file format associated with pnm itself. If PNM is used as the output format specifier, then ImageMagick automatically selects the most appropriate format to represent the image. The default is to write the binary version of the formats. Use `+compress` to write the ASCII version of the formats.
PPM	RW	Portable pixmap format (color)	
PS	RW	Adobe PostScript file	Requires Ghostscript to read.
PS2	RW	Adobe Level II PostScript file	Requires Ghostscript to read.
PS3	RW	Adobe Level III PostScript file	Requires Ghostscript to read.
PSD	RW	Adobe Photoshop bitmap file	
PTIF	RW	Pyramid encoded TIFF	Multi-resolution TIFF containing successively smaller versions of the image down to the size of an icon. The desired sub-image size may be specified when reading via the `-size` option.
PWP	R	Seattle File Works multi-image file	
RAD	R	Radiance image file	Requires that *ra_ppm* from the Radiance software package be installed.
RGB	RW	Raw red, green, and blue samples	Use `-size` and `-depth` to specify the image width, height, and depth.
RGBA	RW	Raw red, green, blue, and alpha samples	Use `-size` and `-depth` to specify the image width, height, and depth.
RLA	R	Alias/Wavefront image file	
RLE	R	Utah Run length encoded image file	
SCT	R	Scitex Continuous Tone Picture	
SFW	R	Seattle File Works image	
SGI	RW	Irix RGB image	
SHTML	W	Hypertext Markup Language client-side image map	Used to write HTML clickable image maps based on a the output of montage or a format which supports tiled images such as MIFF.
SUN	RW	SUN Rasterfile	
SVG	RW	Scalable Vector Graphics	Requires libxml2 and freetype-2. Note that SVG is a very complex specification so support is still not complete.
TGA	RW	Truevision Targa image	Also known as formats "ICB", "VDA", and "VST".
TIFF	RW	Tagged Image File Format	Also known as "TIF". Requires tiff-v3.6.1.tar.gz or later. Note that since Unisys claims a patent on the LZW algorithm (expiring in the US as of June 2003) used by LZW-compressed TIFF files, ImageMagick binary distributions do not include support for the LZW algorithm so LZW TIFF files can not be written. Although a patch is available for libtiff to enable building with LZW support. Users should consult the Unisy LZW web page before applying it.
TIM	R	PSX TIM file	
TTF	R	TrueType font file	Requires freetype 2. Opening as file returns a preview image.
TXT	RW	Raw text file	
UIL	W	X-Motif UIL table	
UYVY	RW	Interleaved YUV raw image	Use `-size` command line option to specify width and height.

VICAR	RW	VICAR rasterfile format	
VIFF	RW	Khoros Visualization Image File Format	
WBMP	RW	Wireless bitmap	Support for uncompressed monochrome only.
WMF	R	Windows Metafile	Requires libwmf. By default, renders WMF files using the dimensions specified by the metafile header. Use the -density option to adjust the output resolution, and thereby adjust the ouput size. The default output resolution is 72DPI so "-density 144" results in an image twice as large as the default. Use **-background color** to specify the WMF background color (default white) or **-texture filename** to specify a background texture image.
WPG	R	Word Perfect Graphics File	
XBM	RW	X Windows system bitmap, black and white only	Used by the X Windows System to store monochrome icons.
XCF	R	GIMP image	
XPM	RW	X Windows system pixmap	Also known as "PM". Used by the X Windows System to store color icons.
XWD	RW	X Windows system window dump	Used by the X Windows System to save/display screen dumps.
YCbCr	RW	Raw Y, Cb, and Cr samples	Use `-size` and `-depth` to specify the image width, height, and depth.
YCbCrA	RW	Raw Y, Cb, Cr, and alpha samples	Use `-size` and `-depth` to specify the image width, height, and depth.
YUV	RW	CCIR 601 4:1:1	

13 Transcoding Multimedia (EPFL)

The goal of content adaptation tools is to adapt the original modality, resolution, and format of multimedia content to potential terminal and network capabilities. A prerequisite for efficient adaptation of multimedia information is a careful analysis of the properties of different media types. Video, audio, images and text require different adaptation algorithms. Consequently, the AXMEDIS framework will provide a number of algorithms to adapt these media types (see sections 13, 15, 16, 17).

13.1 Multimedia: State of the art

The complex nature of multimedia makes the adaptation difficult to design and implement. Indeed, to allow for the composition of the various media streams, most multimedia formats define some XML-like language describing how the various media streams are associated so that tools dealing with such objects shall be able to parse the corresponding textual descriptions. Moreover, objects we refer to as multimedia or rich media may also be composed of synthetic audio (MIDI, MPEG-4 Structured Audio), 2D and 3D graphics, web content, etc… We will now enumerate and describe a number of file formats considered as multimedia.

MPEG-4: The primary uses for the MPEG-4 standard are web (streaming media) and CD distribution, conversational (videophone), and broadcast television. MPEG-4 absorbs many of the features of MPEG-1 and MPEG-2 and other related standards, adding new features such as (extended) VRML support for 3D rendering, object-oriented composite files (including audio, video and VRML (Virtual Reality Modeling Language) objects), support for externally-specified Digital Rights Management and various types of interactivity.

Most of the features included in MPEG-4 are left to individual developers to decide whether to implement them. This means that there are probably no complete implementations of the entire MPEG-4 set of standards. To deal with this, the standard includes the concept of "profiles" and "levels", allowing a specific set of capabilities to be defined in a manner appropriate for a subset of applications.

Scalable Vector Graphics (**SVG**): It is an XML markup language for describing two-dimensional vector graphics, both static and animated. It is an open standard created by the World Wide Web Consortium, which is also responsible for standards like HTML and XHTML.

SVG allows three types of graphic objects:

1. vector graphic shapes (e.g. paths consisting of straight lines and curves, and areas bounded by them)
2. raster graphics images / digital images
3. text

Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. Text can be in any XML namespace suitable to the application, which enhances searchability and accessibility of the SVG graphics. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility.

SVG drawings can be dynamic and interactive. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows straightforward and efficient vector graphics animation via ECMAScript or SMIL. A rich set of event handlers such as *onmouseover* and *onclick* can be

assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same web page.

If storage space is an issue, SVG images are sometimes saved with gzip compression, in which case they may be called "SVGZ files". Because XML contains a lot of redundant data, XML tends to compress very well and these files can be much smaller.

SWF: it is the file format used by Macromedia Flash to describe movies built of mainly two elements: vector based objects and images. The newest versions also allow audio, video and many different possible forms of interaction with the end user. Once created, SWF files can be played by the Macromedia Flash Player, working either as a browser plugin or as an standalone (executable) player. Most of the times, SWF files can also be encapsulated with the player, creating a self-running SWF movie called projector.

VRML: VRML is a text file format where, e.g., vertices and edges for a 3D polygon can be specified along with the surface color, image-mapped textures, shininess, transparency, and so on. URLs can be associated with graphical components so that a web browser might fetch a web-page or a new VRML file from the Internet when the user clicks on the specific graphical component. Animations, sounds, lighting, and other aspects of the virtual world can interact with the user or may be triggered by external events such as timers. A special Script Node allows to add program code (e.g., written in Java or JavaScript (ECMAScript)) to a VRML file.

VRML files are commonly called worlds and have the .wrl extension (for example island.wrl). Although VRML worlds use a text format they may often be compressed using gzip so that they transfer over the internet more quickly. Most 3D modeling programs can save objects and scenes in VRML format.

X3D: X3D is the ISO standard for real time 3D graphics, the successor to Virtual Reality Modelling Language. X3D features extensions to VRML (e.g. Humanoid Animation, Nurbs, GeoVRML etc.), the ability to encode the scene using an XML syntax as well as the Open Inventor-like syntax of VRML97, and enhanced application programmer interfaces (APIs).

SMIL: SMIL is an abbreviation for the **Synchronized Multimedia Integration Language**. It is a W3C Recommendation for describing multimedia presentations using the Extensible Markup Language (XML). It defines timing markup, layout markup, animations, visual transitions, and media embedding, among other things.

SMIL 1.0 became an official recommendation of the World Wide Web Consortium W3C in June 1998. SMIL 2.0 became an official recommendation in August 2001. A W3C working group is currently working on SMIL 2.1, which will include a small number of extensions based on practical experience gathered using SMIL in the Multimedia Messaging System on mobile phones.

A SMIL document is similar in structure to an HTML document in that they are typically divided between a <head> section and a <body> section. The <head> section contains layout and metadata information. The <body> section contains the timing information, and is generally comprised of combinations of two main tags: parallel ("<par>") and sequential ("<seq>"). It refers to media objects by URLs, allowing them to be shared between presentations and stored on different servers

for load balancing. The language can also associate different media objects with different bandwidths.

SMIL enables people without programming or scripting backgrounds to author multimedia presentations in a simple text editor. For example, a developer can write SMIL to display an image after an audio track ends.

3GPP: The 3rd Generation Partnership Project (3GPP) is a collaboration agreement that was established in December 1998. The collaboration agreement brings together a number of telecommunications standards bodies which are known as "Organizational Partners". The current Organizational Partners are ARIB, CCSA, ETSI, ATIS, TTA, and TTC. 3GPP is the new worldwide standard for sharing multimedia content between cell phones, PDAs and computers. It features high quality at extremely low bitrates and is perfect for encoding speech.

As it was seen in AXMEDIS specification DE3-1-2C, there are only a few libraries available to manipulate multimedia files. Among them is GPAC, a multimedia framework based on the MPEG-4 Systems standard (ISO/IEC 14496-1) developed from scratch in ANSI C. As of version 0.4.0 GPAC is released under the GNU Lesser General Public License. At the time of the specification of AXMEDIS (see DE3-1-2C), GPAC was released under a GNU General Public Licensing scheme, which made it unsuitable for integration into the AXMEDIS framework. Furthermore, the GPAC project was at that time in a frozen state and has been restarted since June 2005, so that it is now one of the most advanced projects dedicated to advanced content. Consequently, we propose to use the functionalities of the GPAC library to perform multimedia files adaptation in the AXMEDIS framework. The following is summary of the functionalities provided by the GPAC library.

GPAC aims at integrating the recent multimedia standards (SVG/SMIL, VRML, X3D, SWF, 3GPP(2) tools, etc…) into a single framework. VRML97 and a good amount of the X3D standard have already been integrated into GPAC, as well as some SVG support and experimental Macromedia Flash support. The current GPAC release (0.4.0) already covers a very large part of the MPEG-4 standard, and has some good support for 3GPP and VRML/X3D. It is currently running under Windows, Linux platforms and Windows CE/PocketPC 2002 (2D rendering only, not tested on SmartPhones).

In terms of adaptation capabilities, the MP4Box developed with GPAC should be integrated in the context of AXMEDIS. MP4Box aims at providing all tools needed to produce and distribute MPEG-4, 3GP and 3GP2 content in a single command-line application. Here are is a list of some of the functionalities offered by MP4Box:

- File splitting by size or time, chunk extraction from file and file concatenation, including all supported input files concatenation (e.g. cat a set of AVIs to a single 3GPP/MP4),
- MP4/3GP conversion from MP3, AVI, MPEG-PS, AAC, H263, H264, AMR, QCP, EVRC, SMV, SUB, SRT, TTXT, TeXML…

The goal of content adaptation tools is to adapt the original modality, resolution, and format of multimedia content to potential terminal and network capabilities. A prerequisite for efficient adaptation of multimedia information is a careful analysis of the properties of different media types. Video, audio, images and text require different adaptation algorithms. Consequently, the AXMEDIS framework will provide a number of algorithms to adapt these media types (see sections 13, 15, 16, 17).

13.2 Multimedia: State of the art

The complex nature of multimedia makes the adaptation difficult to design and implement. Indeed, to allow for the composition of the various media streams, most multimedia formats define some XML-like language describing how the various media streams are associated so that tools dealing with such objects shall be able to parse the corresponding textual descriptions. Moreover, objects we refer to as multimedia or rich media may also be composed of synthetic audio (MIDI, MPEG-4 Structured Audio), 2D and 3D graphics, web content, etc… We will now enumerate and describe a number of file formats considered as multimedia.

MPEG-4: The primary uses for the MPEG-4 standard are web (streaming media) and CD distribution, conversational (videophone), and broadcast television. MPEG-4 absorbs many of the features of MPEG-1 and MPEG-2 and other related standards, adding new features such as (extended) VRML support for 3D rendering, object-oriented composite files (including audio, video and VRML (Virtual Reality Modeling Language) objects), support for externally-specified Digital Rights Management and various types of interactivity.

Most of the features included in MPEG-4 are left to individual developers to decide whether to implement them. This means that there are probably no complete implementations of the entire MPEG-4 set of standards. To deal with this, the standard includes the concept of "profiles" and "levels", allowing a specific set of capabilities to be defined in a manner appropriate for a subset of applications.

Scalable Vector Graphics (**SVG**): It is an XML markup language for describing two-dimensional vector graphics, both static and animated. It is an open standard created by the World Wide Web Consortium, which is also responsible for standards like HTML and XHTML.

SVG allows three types of graphic objects:

4. vector graphic shapes (e.g. paths consisting of straight lines and curves, and areas bounded by them)
5. raster graphics images / digital images
6. text

Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. Text can be in any XML namespace suitable to the application, which enhances searchability and accessibility of the SVG graphics. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility.

SVG drawings can be dynamic and interactive. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows straightforward and efficient vector graphics animation via ECMAScript or SMIL. A rich set of event handlers such as *onmouseover* and *onclick* can be assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same web page.

If storage space is an issue, SVG images are sometimes saved with gzip compression, in which case they may be called "SVGZ files". Because XML contains a lot of redundant data, XML tends to compress very well and these files can be much smaller.

SWF: it is the file format used by Macromedia Flash to describe movies built of mainly two elements: vector based objects and images. The newest versions also allow audio, video and many different possible forms of interaction with the end user. Once created, SWF files can be played by the Macromedia Flash Player, working either as a browser plugin or as an standalone (executable)

player. Most of the times, SWF files can also be encapsulated with the player, creating a self-running SWF movie called projector.

VRML: VRML is a text file format where, e.g., vertices and edges for a 3D polygon can be specified along with the surface color, image-mapped textures, shininess, transparency, and so on. URLs can be associated with graphical components so that a web browser might fetch a web-page or a new VRML file from the Internet when the user clicks on the specific graphical component. Animations, sounds, lighting, and other aspects of the virtual world can interact with the user or may be triggered by external events such as timers. A special Script Node allows to add program code (e.g., written in Java or JavaScript (ECMAScript)) to a VRML file.

VRML files are commonly called worlds and have the .wrl extension (for example island.wrl). Although VRML worlds use a text format they may often be compressed using gzip so that they transfer over the internet more quickly. Most 3D modeling programs can save objects and scenes in VRML format.

X3D: X3D is the ISO standard for real time 3D graphics, the successor to Virtual Reality Modelling Language. X3D features extensions to VRML (e.g. Humanoid Animation, Nurbs, GeoVRML etc.), the ability to encode the scene using an XML syntax as well as the Open Inventor-like syntax of VRML97, and enhanced application programmer interfaces (APIs).

SMIL: SMIL is an abbreviation for the **Synchronized Multimedia Integration Language**. It is a W3C Recommendation for describing multimedia presentations using the Extensible Markup Language (XML). It defines timing markup, layout markup, animations, visual transitions, and media embedding, among other things.

SMIL 1.0 became an official recommendation of the World Wide Web Consortium W3C in June 1998. SMIL 2.0 became an official recommendation in August 2001. A W3C working group is currently working on SMIL 2.1, which will include a small number of extensions based on practical experience gathered using SMIL in the Multimedia Messaging System on mobile phones.

A SMIL document is similar in structure to an HTML document in that they are typically divided between a <head> section and a <body> section. The <head> section contains layout and metadata information. The <body> section contains the timing information, and is generally comprised of combinations of two main tags: parallel ("<par>") and sequential ("<seq>"). It refers to media objects by URLs, allowing them to be shared between presentations and stored on different servers for load balancing. The language can also associate different media objects with different bandwidths.

SMIL enables people without programming or scripting backgrounds to author multimedia presentations in a simple text editor. For example, a developer can write SMIL to display an image after an audio track ends.

3GPP: The 3rd Generation Partnership Project (3GPP) is a collaboration agreement that was established in December 1998. The collaboration agreement brings together a number of telecommunications standards bodies which are known as "Organizational Partners". The current Organizational Partners are ARIB, CCSA, ETSI, ATIS, TTA, and TTC. 3GPP is the new worldwide standard for sharing multimedia content between cell phones, PDAs and computers. It features high quality at extremely low bitrates and is perfect for encoding speech.

As it was seen in AXMEDIS specification DE3-1-2C, there are only a few libraries available to manipulate multimedia files. Among them is GPAC, a multimedia framework based on the MPEG-4 Systems standard (ISO/IEC 14496-1) developed from scratch in ANSI C. As of version 0.4.0 GPAC is released under the GNU Lesser General Public License. At the time of the specification of AXMEDIS (see DE3-1-2C), GPAC was released under a GNU General Public Licensing scheme, which made it unsuitable for integration into the AXMEDIS framework. Furthermore, the GPAC project was at that time in a frozen state and has been restarted since June 2005, so that it is now one of the most advanced projects dedicated to advanced content. Consequently, we propose to use the functionalities of the GPAC library to perform multimedia files adaptation in the AXMEDIS framework. The following is summary of the functionalities provided by the GPAC library.

GPAC aims at integrating the recent multimedia standards (SVG/SMIL, VRML, X3D, SWF, 3GPP(2) tools, etc…) into a single framework. VRML97 and a good amount of the X3D standard have already been integrated into GPAC, as well as some SVG support and experimental Macromedia Flash support. The current GPAC release (0.4.0) already covers a very large part of the MPEG-4 standard, and has some good support for 3GPP and VRML/X3D. It is currently running under Windows, Linux platforms and Windows CE/PocketPC 2002 (2D rendering only, not tested on SmartPhones).

In terms of adaptation capabilities, the MP4Box developed with GPAC should be integrated in the context of AXMEDIS. MP4Box aims at providing all tools needed to produce and distribute MPEG-4, 3GP and 3GP2 content in a single command-line application. Here are is a list of some of the functionalities offered by MP4Box:

- File splitting by size or time, chunk extraction from file and file concatenation, including all supported input files concatenation (e.g. cat a set of AVIs to a single 3GPP/MP4),
- MP4/3GP conversion from MP3, AVI, MPEG-PS, AAC, H263, H264, AMR, QCP, EVRC, SMV, SUB, SRT, TTXT, TeXML…
- Media track extractions.
- MPEG-4 BIFS codec and scene conversion from and to MP4, BT and XMT-A formats.
- Conversion of simple Macromedia Flash (SWF) to MPEG-4 Systems (BT/XMT/MP4).
- Conversion to and from BT, XMT-A, WRL, X3D, X3DV formats (GZiped or not).

13.3 Work performed:

The multimedia adaptation functionalities are to be used as plug-ins through the AXCP interface. The plug-in simply consists of a DLL and an XML file describing the functionalities of the DLL. Both the DLL and the XML description should be installed in the plug-in directory of the AXCP compliant tool using the plug-in.

ExtractMediaTrack

This function extracts one track from the original source into a separate file. This extraction can be done in two ways:

Extraction of the track into an mp4 file with a single track.
Extraction in the native format of the track (mpeg, mp3…).

MP4to3GP

This function translates the MP4 input resource into a new resource with the 3GP format.

MP4toISMA

This function translates the MP4 input resource into a new resource conforming to the ISMA specification.

CatMultimediaFiles

This function concatenates two whole multimedia resources and produces a new resource containing the concatenation of the initial resources.

AddMediaFiles

This function imports multimedia resources as new tracks into new or already existing mp4 files. It must be specified the size (amount of seconds) of the multimedia resource that is imported and when should it begin inside the destination file, it is to say, the delay of the new track.

13.4 Formal description of multimedia adaptation algorithm

Formal description of algorithm ExtractMediaTrack

Description: extracts one track from the original source into a separate file. This extraction can be done in two ways:

- Extraction of the track into an mp4 file with a single track.
- Extraction in the native format of the track (mpeg, mp3…).

Signature:

string ExtractMediaTrack(*AxResource* InputResource, *AxResource* OutputResource, *UINT32*, TrackID, *UINT16* ExtractToNativeFormat)

Parameter List:

Name: InputResource
Description: the resource to be converted
Parameter Type: *AxResource*
Default Value:
Constraints:
Ranges:

Name: OutputResource
Description: Where the output resource will be stored
Parameter Type: *AxResource*
Default Value:
Constraints:
Range:

Name: TrackID
Description: The number of the track to extract
Parameter Type: *uint32*
Default Value:
Constraints:

Range:

Name: ExtractToNativeFormat
Description: 1=Native format (.mp3, mp2 etc,,,), 0=Non.Native Format (.mp4)
Parameter Type: *uint16*
Default Value:
Constraints:
Range:

Formal description of algorithm MP4o3GP

Description: translates the MP4 input resource into a new resource with the 3GP format.

Signature:

string Mp4To3Gp(*AxResource* InputResource, *AxResource* OutputResource)

Parameter List:

Name: InputResource
Description: the resource to be converted
Parameter Type: *AxResource*
Default Value:
Constraints:
Ranges:

Name: OutputResource
Description: Where the output resource will be stored
Parameter Type: *AxResource*
Default Value:
Constraints:
Range:

Formal description of algorithm MP4toISMA

Description: translates the MP4 input resource into a new resource conforming to the ISMA specification.

Signature:

string Mp4ToIsma(*AxResource* InputResource, *AxResource* OutputResource)

Parameter List:

Name: InputResource
Description: the resource to be converted
Parameter Type: *AxResource*
Default Value:
Constraints:
Ranges:

Name: OutputResource
Description: Where the output resource will be stored

Parameter Type: *AxResource*
Default Value:
Constraints:
Range:

Formal description of algorithm CatMultimediaFiles

Description: concatenates two whole multimedia resources and produces a new resource containing the concatenation of the initial resources.

Signature:

string CatMultimediaFiles(*AxResource* InputResourceA, *AxResource* InputResourceB, *AxResource* OutputResource)

Parameter List:

Name: InputResourceA
Description: the first resource in the concatenation
Parameter Type: *AxResource*
Default Value:
Constraints:
Ranges:

Name: InputResourceB
Description: the second resource in the concatenation
Parameter Type: *AxResource*
Default Value:
Constraints:
Ranges:

Name: OutputResource
Description: Where the output resource will be stored
Parameter Type: *AxResource*
Default Value:
Constraints:
Range:

Formal Descripiton of Algorithm AddMultimediaFiles

Description: This function imports multimedia resources as new tracks into new or already existing mp4 files. It must be specified the size (amount of seconds) of the multimedia resource that is imported and when should it begin inside the destination file, it is to say, the delay of the new track.

Signature:
string AddMultimediaFiles (*AxResource* InputResource, *UINT32* Delay, *UINT32* ImporLength *AxResource* OutputResource)

Parameter List:

Name: InputResource
Description: Resource to be converted
Parameter Type: *AxResource*

Default Value:
Constraints:
Ranges:

Name: Delay
Description: Delay in milliseconds of the new track
Parameter Type: *UINT 32*
Default Value: 0
Constraints:
Ranges:

Name: Delay
Description: Number of seconds to import from input file (starting from the beginning)
Parameter Type: *UINT 32*
Default Value: 0
Constraints:
Ranges:

Name: OutputResource
Description: Where the produced resource will be stored
Parameter Type: *AxResource*
Default Value:
Constraints:
Ranges:

GPAC

GPAC is a multimedia framework based on the MPEG-4 Systems standard (ISO/IEC 14496-1) developed from scratch in ANSI C. As of version 0.4.0 GPAC is licensed under the GNU Lesser General Public License. Older GPAC versions are available under the GNU General Public License.

The original development goal is to provide a clean (a.k.a. readable by as many people as possible), small and flexible alternative to the MPEG-4 Systems reference software (known as IM1 and distributed in ISO/IEC 14496-5). The MPEG-4 Reference software is indeed a very large piece of software, designed to verify the standard rather than provide a small, production-stable software. GPAC is written in (almost 100% ANSI) C for portability reasons (embedded platforms and DSPs) with a simple goal: keep the memory footprint as low as possible.

The natural evolution has been the integration of recent multimedia standards (SVG/SMIL, VRML, X3D, SWF, 3GPP(2) tools, etc) into a single framework. VRML97 and a good amount of the X3D standard have already been integrated into GPAC, as well as some SVG support and experimental Macromedia Flash support.

The current GPAC release (0.4.0) already covers a very large part of the MPEG-4 standard, and has some good support for 3GPP and VRML/X3D, and features what can probably be seen as the most advanced and robust 2D MPEG-4 Player available worldwide, as well as a decent 3D player - have a look at the screenshots page.

GPAC also features MPEG-4 Systems encoders/multiplexers, publishing tools for content distribution for MP4 and 3GPP(2) files and many tools for scene descriptions (MPEG4<->VRML<->X3D converters, SWF->MPEG-4, etc...).

GPAC is currently running under Windows, Linux platforms and WindowsCE/PocketPC 2002 (2D rendering only, not tested on SmartPhones).

IBM Toolkit for MPEG-4

The IBM toolkit for MPEG-4 consists of a set of technologies compliant with the MPEG-4 standard. It is implemented as a set of Java classes and APIs which can be used to develop MPEG-4 applications for authoring and playback. It is the main toolset around which the XMT specification has been built, as a case study for compatibility of the different multimedia XML-based languages such as SMIL, XMT and VRML. Since the toolkit is Java-based, the applications built on top of it will run on any platform that supports Java.

Yet, the IBM toolkit for MPEG-4 is released under a commercial licensing scheme and is available only for 90 days in trial license. License costs range from 500$ (1000 toolkits for internal use only) to 5000$ (500 toolkits for internal and/or external use and distribution).

SMIL to MPEG-4 BIFS conversion

Though SMIL and BIFS have very similar functionalities and target applications, the former appears to be easier to author while the latter seems more suited to broadcasting applications and video-on-demand and offers mechanisms for copy protection and intellectual property management. Moreover, there is a broad potential for BIFS in the PDA sector since these appliances are based on chip sets and not downloadable software.

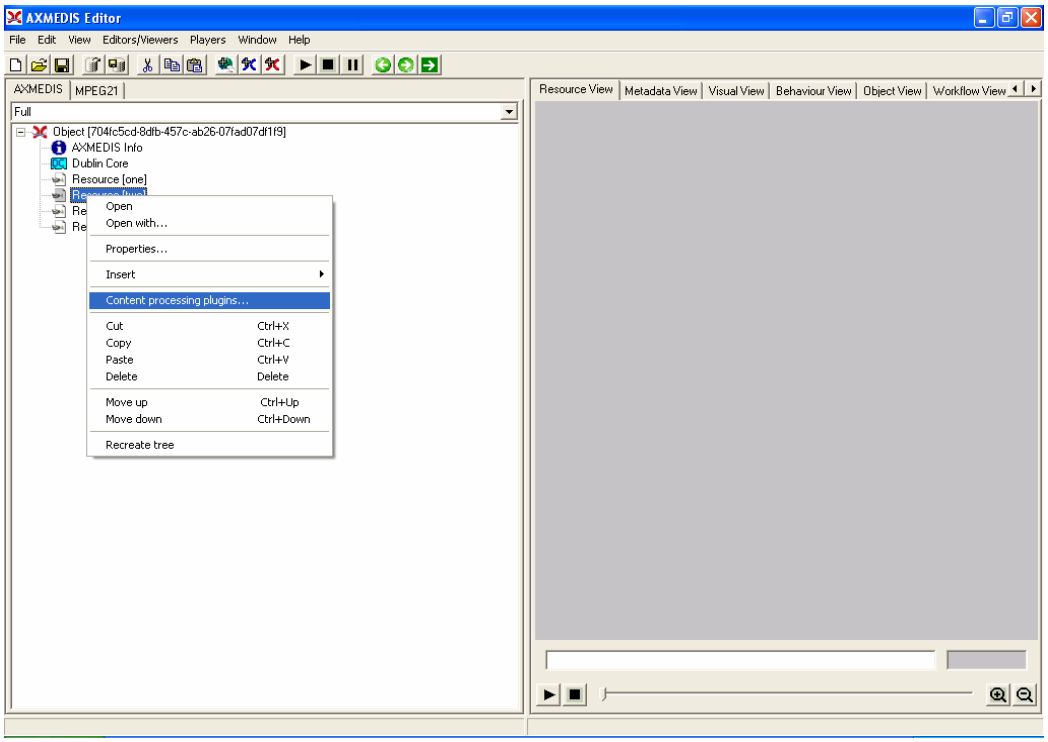
Consequently, to take advantage of SMIL content in a larger range of applications or simply to benefit from existing SMIL authoring tools to develop BIFS content (such as the SMIL editor developed in the context of AXMEDIS), it seems interesting to convert SMIL presentations to BIFS format.

Yet, no existing tool or library exists to do this conversion, so that a SMIL to MPEG-4 BIFS converter will be developed in the context of AXMEDIS.

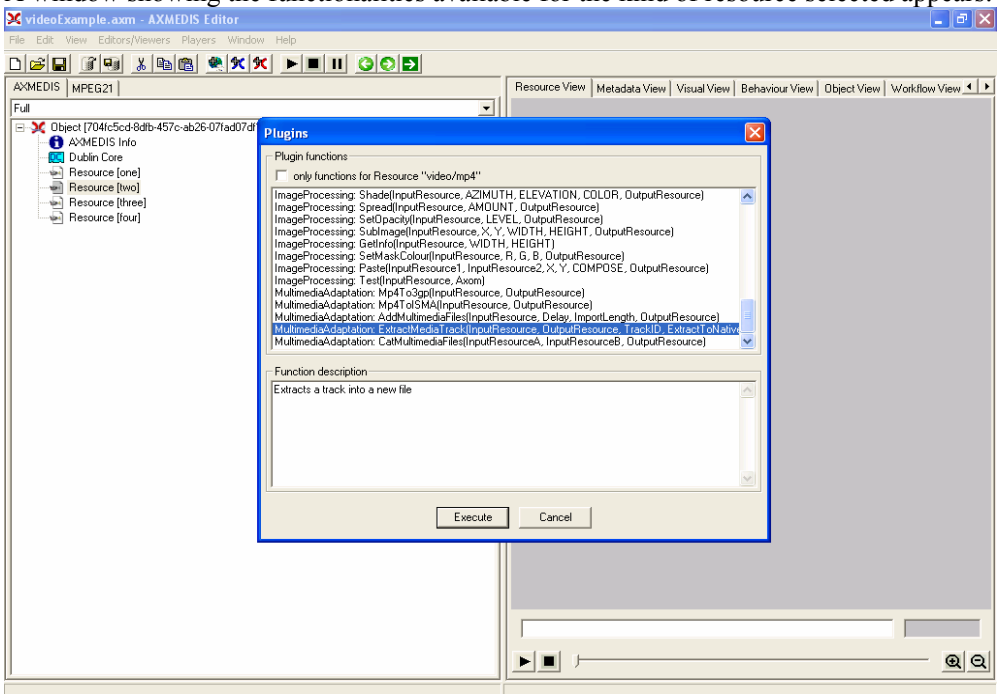
13.5 Multimedia adaptation plug-in

Here's an example on how to use the multimedia adaptation track extraction function as a plug-in for the AXMEDIS editor.

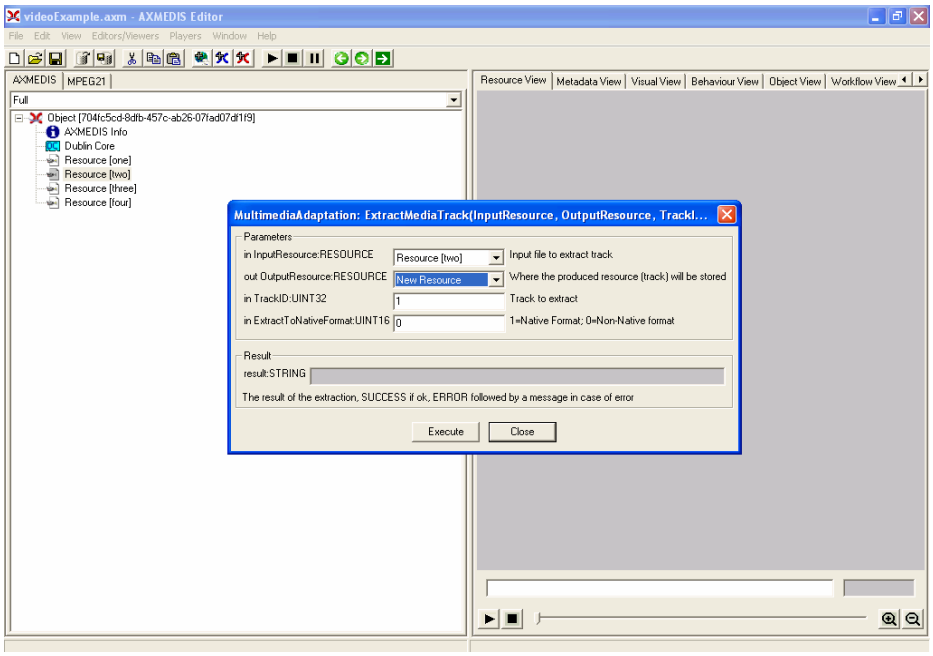
The plug-in must be applied on an MP4 resource of an AXMEDIS object. The adaptation plug-in is called by right-clicking on the interesting resource and selecting the 'Plugin…' command:



A window showing the functionalities available for the kind of resource selected appears:



The ExtractMediaTrack function allows extracting a media track from an input MP4 resource into a new resource. In this example, we extract the first track of the input resource. The first track is an audio track in MP3 format. The extraction is selected to be in non-native format i.e. the extracted media track will not be a MP3 file but it will be converted to MP4. Once extracted, further adaptation can be performed with other plug-ins.



Work to be done:

closed

14 Transcoding/adaptation PAR and Licenses (FUPF)

DRM adaptation involves the adaptation of the related licenses, as derived AXMEDIS objects or digital resources within the AXMEDIS objects can be seen as new creations with regard to original ones. Therefore, new licenses must be created during the adaptation process, always respecting the terms and conditions fixed in the original license or licenses for the adapted AXMEDIS objects or contents within these objects.

Nevertheless, DRM information (mainly licenses and PARs) inside the AXMEDIS project, that are related to AXMEDIS Objects will be expressed in XML language by using MPEG-21 REL.

In order to adapt this information to different rights expression languages, also based in XML or to adapt a license to be more compact in order to use it into portable devices (for instance, mobile phones or PDAs), we will make use of existing libraries for manipulating XML documents.

The following sections describe the research work performed in the adaptation of PAR and Licenses. Currently, we do not have a prototype that demonstrates the work described in this section, but isolated modules that provide a brief part of the functionality. Nevertheless, the implementation of this prototype was not planned for this period and it is expected that the corresponding prototype will be ready for the planned date.

14.1 PAR and Licenses: State of the art

14.1.1 MPEG-21 Rights Expression Language (REL)

The different parties involved in the online distribution and consumption of multimedia resources need to exchange information about the rights, terms, and conditions associated with each resource at each step in the multimedia resource lifecycle. For example in distribution and super distribution business models, the information related to the rights and the terms and conditions under which the rights may be exercised needs to be communicated to each participant in the distribution chain.
In an end-to-end system, other considerations such as authenticity and integrity of Rights Expressions become important. For example, any content provider or distributor who issues rights to use or distribute resources must be identified and authorized. In addition, different participants may access a Rights Expression, which requires mechanisms and semantics for validating the authenticity and integrity of the Rights Expression. A common Rights Expression Language that can be shared among all participants in this digital workflow is required.

Part 5 of the MPEG-21 standard specifies the syntax and semantics of a Rights Expression Language. MPEG chose XrML [1] as the basis for the development of the MPEG-21 Rights expression language.
MPEG-21 Rights Expression Language (REL) [2] specifies the syntax and semantics of the language for issuing rights for Users to act on Digital Items, their Components, Fragments, and Containers.
The most important concept in REL is the license that conceptually is a container of grants, each one of which conveys to a principal the sanction to exercise a right against a resource. A license if formed by the following elements:

- Title: this element provides a descriptive phrase about the License that is intended for human consumption in user interfaces. Automated processors must not interpret semantically the contents of such title elements.
- Inventory: this element is used for defining variables within a License. In the Inventory element of a license can be defined LicensePart elements that in turn can have licensePartId attributes that can be referenced from elsewhere in the license.

Therefore, REL provides a syntactic mechanism for reducing redundancy and verbosity in Licenses that can be used throughout a License.

- Grant or GrantGroup: The Grants and GrantGroups contained in a license are the means by which authorization policies are conveyed in the REL architecture. Each Grant or GrantGroup that is an immediate child of a license exists independently within that license, no collective semantic (having to do with their particular ordering or otherwise) is intrinsically associated with the presence of two or more of them within a certain license.
- Other information: Using the wildcard construct from XML Schema, a License provides an extensibility hook within which license issuers may place additional content as they find appropriate and convenient. This can be useful for conveying information that is peripherally related to, for example, authentication and authorization, but is not part of the REL core infrastructure.
  It should, however, be carefully understood that not all processors of REL licenses will understand the semantics intended by any particular use of this extensibility hook. Processors of the license may choose wholly at their own discretion to completely ignore any such content that might be present therein.

Next figure shows the structure of a REL License.

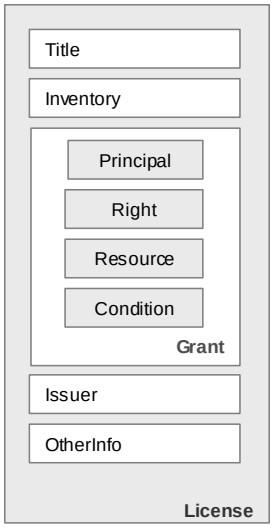


Figure REL License Structure

The most important concept within a license is the grant that conveys to a particular principal the sanction to exercise some identified right against some identified resource, possibly subject to the need for some condition to be first fulfilled. A Grant is an XML structure that is at the heart of the rights management and authorization policy semantics that REL is designed to express.

A grant is formed by four elements, a Principal that represents the unique identification of an entity involved in the granting or exercising of Rights. A Right that specifies an action or activity that a Principal may perform on, or using, some associated target Resource. A Resource that represents the object against which the Principal of a Grant has the Right to perform. The use of a digital resource in a Grant provides a means by which a sequence of digital bits can be identified within the Grant. The Condition element represents grammatical terms, conditions and obligations that a Principal must satisfy before it may take advantage of an authorization conveyed to it in a Grant. The issuer element that may contain two pieces of information, a set of issuer-specific details about the circumstances under which he issues the license, and an identification of the issuer, possibly coupled with a digital signature for the license. The optional issuer-specific details are found in the details element of the issuer. These details optionally include any of the following information the specific date and time at which this issuer claims to have effected his issuance of the license and an

indication of the mechanism or mechanisms by which the Issuer of the license will, if he later Revokes it, post notice of such revocation. When checking for revocation, REL processing systems may choose to use any one of the identified mechanisms, that is, they are all considered equally authoritative as to the revocation status of the issuance of the License.

The structure of a REL license is the one described if it is in clear text, but a REL license can contain only an encryptedLicense element if the license is encrypted. The encryptedLicense element provides a mechanism by which the contents of a License may be encrypted and so hidden from view from inappropriate parties. This mechanism makes straightforward use of XML Encryption Syntax and Processing (XML Encryption). Specifically, the XML content model of a License is a choice between a sequence containing the elements previously described in this section and an encryptedLicense element that represents the encryption of the contents of the License element.

The principals, rights, resources and conditions of the REL are organized in three main groups. The first one, the Core specifies structural elements and types and how are they related. The standard extension and the multimedia extension specifies standard or multimedia principals, rights, resources and conditions.

At the heart of REL is the REL Core Schema whose elements and types define the core structural and validation semantics that comprises the essence of the specification. The REL Core Schema includes different elements and types organised in four main groups:

- Principals: Within REL, instances of the type Principal represent the unique identification of an entity involved in the granting or exercising of rights. They represent the subject that is permitted to carry out the action involved in exercising the Right. The principal element and its type are conceptually abstracts. Then, it does not indicate how a particular principal is actually identified and authenticated. Rather, this is carried out in types that are derivations of Principal. Such derived types may be defined in extensions to REL in order to provide, for example, a means by which Principals who are authenticated using some proprietary logon mechanism may be granted certain Rights using the REL License mechanism.
  There are derivations that are important and central enough to be defined within the REL core itself:
- allPrincipals: Structurally, an AllPrincipals Principal is a simple container of Principals. Semantically, an AllPrincipals represents the logical conjunct of the Principals represented by all of its children.
- keyHolder: Instances of a KeyHolder Principal represent entities which are identified by their possession of a certain cryptographic key. For example, using a KeyHolder, a Principal that uses public-key cryptography may be conceptually identified as that Principal which possesses the private key that corresponds to this-here public key.
- Rights: Within REL, instances of the type Right represent a verb that a Principal may be authorized to carry out under the authority conveyed by some authorized Grant. Typically, a Right specifies an action or activity that a Principal may perform on or using some associated target Resource. The semantic specification of each different particular kind of Right should indicate which kinds of Resource if any may be legally used in authorized Grants containing that Right.
  The element right and its type are conceptually abstract. Therefore, the type Right itself does not indicate any actual action or activity that may be carried out. Rather, such actions or activities are to be defined in types that are derivations of Right. Such derived types will commonly be defined in extensions to REL. However, the following rights are related to the domain of the REL core itself:
- issue: When an Issue element is used as the right in an authorized grant, it is required that resource against which the right is applied in fact be a grant or grantGroup. The grant then conveys the authorization for the principal to issue the resource.
  At the instant a License is issued, the issue right must be held by the issuer of the License with respect to all the grants and grantGroups directly authorized therein.

- obtain: When an obtain element is used as the right in an authorized grant, the resource must be present and be a grant or a grantGroup. The use of the obtain right can be conceptualized as an offer or advertisement for the sale of the contained grant
- possessProperty: The possessProperty right represents the right for the associated principal to claim ownership of a particular characteristic, which is listed as the resource associated with this Right.
- revoke: The authorized act of exercising the revoke right by a principal effects a retraction of a dsig:Signature that was previously issued and thus accomplishes a withdrawal of any authorization conveyed by that dsig:Signature.
- Resources: An instance of type resource represents the direct object against which the subject principal of a grant has the right to perform some verb. The actual element resource and its type are conceptually abstracts. That is, the type resource itself does not indicate any actual object against which a Right may be carried out. Rather, such target objects are to be defined in types that are derivations of Resource. Such derived types will commonly be defined in extensions to REL. The relevant resources defined within the REL core:
- digitalResource: Use of a digitalResource resource in a grant provides a means by which an arbitrary sequence of digital bits can be identified as being the target object of relevance within the grant. Specifically, such bits are not required to be character strings that conform to the XML specification, but may be arbitrary binary data. The means by which this is accomplished breaks down into several cases. For example, the bits are to be physically present within the digitalResource or the bits are to be physically located at some external location (e.g. in a Web site).
- propertyAbstract: An instance of type propertyAbstract represents some sort of property that can be possessed by principals via possessProperty right.
- Conditions: Within REL, instances of the type Condition represent grammatical terms and conditions that a Principal must satisfy before it may take advantage of an authorization conveyed to it in a grant containing the condition instance. The semantic specification of each different particular kind of condition must indicate the details of the terms, conditions, and obligations that use of the Condition actually imposes. When these requirements are fulfilled, the Condition is said to be satisfied.
  The actual element condition and its type are conceptually abstracts. That is, the type Condition itself does not indicate the imposition of any actual term or condition. Rather, such terms and conditions are to be defined in types that are derivations of Condition. Such derived types will commonly be defined in extensions to REL. The conditions defined within the REL core that we consider relevant to detail:
  - AllConditions: Structurally, an allConditions is a simple container of conditions. Semantically, the allConditions represents a logical conjunct of the conditions represented by all of its children.
  - validityInterval: A ValidityInterval condition indicates a contiguous, unbroken interval of time. The semantics of the condition expressed is that the interval of the exercise of a right to which a validityInterval is applied must lie wholly within this interval. The delineation of the interval is expressed by the presence, as children of the condition, of up to two specific fixed time instants:
  - notBefore element, of type xsd:dateTime, indicates the inclusive instant in time at which the interval begins. If absent, the interval is considered to begin at an instant infinitely distant in the past
  - notAfter element, also of type xsd:dateTime, indicates the inclusive instant in time at which the interval ends. If absent, the interval is considered to end at an instant infinitely distant in the future.

The Standard Extension schema defines terms to extend the usability of the Core Schema, some of them are:

- Right Extensions: Right Uri.
- Resource Extensions: Property Extensions and Revocable.
- Condition Extensions: Stateful Condition, State Reference Value Pattern, Exercise Limit Condition, Transfer Control Condition, Seek Approval Condition, Track Report Condition, Track Query Condition, Validity Interval Floating Condition, Validity Time Metered Condition, Validity Time Periodic Condition, Fee Condition and Territory Condition.
- Payment Abstract and its Extensions: Payment Abstract, Rate, Payment Flat, Payment Metered, Payment per Interval, Payment per Use, Best Price Under, Call for Price and Markup.
- Service Description: WSDL and UDDI
- Country, Region and Currency Qualified Names: Namespace URI Structure, Country Qualified Names, Region Qualified Names and Currency Qualified Names.
- Matches XPath Function: Regular Expression Syntax and Flags.

The REL Multimedia Extension expands the Core Schema by specifying terms that relate to digital works. Specifically describes rights, conditions and metadata for digital works, that includes:

- Rights: Modify, Enlarge, Reduce, Move, Adapt, Extract, Embed, Play, Print, Execute, Install, Uninstall and Delete.
- Resources: Digital Item Resources.
- Conditions: Resource Attribute Conditions, Digital Item Conditions, Marking Conditions, Security Conditions and Transactional Conditions.
- Resource Attribute Set Definitions: Complement, Intersection, Set and Union.

14.1.2 ODRL

The ODRL [3] is a proposed language for the DRM community for the standardisation of expressing rights information over content. The ODRL is intended to provide flexible and interoperable mechanisms to support transparent and innovative use of digital resources in publishing, distributing and consuming of electronic publications, music, audio, movies, digital images, learning objects, computer software and other creations in digital form.
Using ODRL it is possible to specify, for a digital resource (music work, content, service, or software application), which is allowed to use that resource, the rights available to them and the terms, conditions or restrictions necessary to exercise those rights on the resource. The ODRL function is to express rights granted by some parties for specific resources and the conditions under which those rights apply.
ODRL is based on an extensible model for rights expressions, which involves three core entities and their relationships: **Party** (identifies an entity such as the person, organisation, or device to whom rights are granted), **Right** (includes permissions, which can then contain constraints, requirements, and conditions) and **Asset** (includes any physical or digital content).

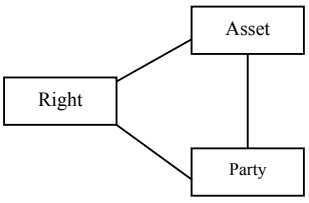


Figure Core elements of ODRL

14.1.3 OMA DRM Rights Expression Language

OMA (Open Mobile Alliance) has developed the OMA DRM Rights Expression Language versions [4] based on ODRL [3].

Rights are the collection of permissions and constraints defining under which circumstances access is granted to DRM Content. The structure of the rights expression language enables the following functionality:

1. Metadata such as version and content ID
2. The actual rights specification consisting of
    a. Linking to and providing protection information for the content, and
    b. Specification of usage rights and constraints

Models are used to group rights elements according to their functionality, and thus enable concise definition of elements and their semantics. The following models are used throughout this specification:

- Foundation model: constitutes the basis for rights. It contains the rights element bringing together meta information and agreement information. The foundation model serves as the starting point for incorporating the agreement model and the context model.
- Agreement model: expresses the Rights that are granted over an DRM Content. It consists of the agreement element connecting a set of Rights with the corresponding DRM Content specified with the asset element. The agreement model incorporates the permission model and the security model
- Context model: provides Meta information about the rights. It augments the foundation model, the agreement model, and the constraint model by expressing additional information.
- Permission model: augments the agreement model. It facilitates the expression of permissions over assets by specifying the access granted to a device. The permission model incorporates the constraint model allowing fine-grained consumption control of DRM Content. The set of permissions comprises play, display, execute, print, and export. Usage of the DRM Content MUST only be granted according to the permissions explicitly specified by the corresponding Rights Object(s). A permission that does not contain a constraint child element is unconstrained and access according to the respective permission element(s) MUST be granted. Note that the REL only specifies consumption and export rights and not management rights, e.g., install, uninstall, delete, or distribution rights. This is made possible by the separation of DRM Content and Rights Objects (although DRM Content and Rights Objects may be delivered together) freeing the REL from unnecessary complexity and overhead. Content can be stored; however, it can only be accessed if a corresponding Rights Object is available. Similarly, encrypted content can be super-distributed without unnecessarily complicating the REL; no separate distribution permissions are necessary, since DRM Content without the decryption key is of no value. The DRM Agent MUST ignore unknown or unsupported permission elements. The DRM Agent MUST NOT grant alternative, not explicitly specified rights to access Content instead. Known and supported permission elements defined by the same Rights Object MUST remain unaffected and the DRM Agent MUST grant access according to those. A Permission that is not granted due to unknown or unsupported constraints (section 5.5) MUST NOT affect the granting of other permissions.
- Constraint model: enhances the permission model by providing fine-grained consumption control of content. Constraints are associated with one permission element at a time. For a permission to be granted all its constraints MUST be fulfilled. If a constraint is not understood or cannot be enforced by the consuming device the parent permission is invalid and must no be granted. If present, a constraint element should contain at least one of its child elements. If a constraint element does not contain any constraints such as count, datetime, etc. it is unconstrained, and a DRM Agent must grant unconstrained access according to the permission containing such an unconstrained constraint element.
- Inheritance model: describes how a parent Rights Object can specify Permissions and Constraints for one or more pieces of DRM Content each governed by a child Rights Object, using a limited subset of the ODRL inheritance model. The DRM Agent must not accept parent child Rights Objects constellations with more than one level of inheritance (i.e. parent-child). In other words, a parent Rights Object must not inherit Permissions and Constraints from another Rights Object.
- Security model: Security constitutes an important part of a DRM system. OMA DRM 2.0 provides confidentiality for the CEK of Rights Objects, integrity of the association between Rights Objects

and DRM Content and Rights Object integrity and authenticity. The ODRL security model, which forms the basis for the security model of this specification, is based on XMLENC [5] and XMLSIG [6].

14.1.4 PAR and Licenses: The problems

Different systems define different rights expression languages. If we want to make them interoperable, or simply work with different kinds of devices we may be obliged to adapt rights expressions.

Moreover, changing versions of specifications make difficult to implement any kind of software to solve this issue.

14.1.5 PAR and Licenses: Work performed

In order to be able to transform rights expressions expressed in different rights expression languages, the first step to take is to study their common points. We have studied in detail the relationship between MPEG-21 REL and ODRL, based on the OMA DRM REL, as described next.

14.1.5.1 OMA-based MPEG-21 REL DTD

In order to perform adaptation of rights between OMA REL and MPEG-21 REL, we propose an equivalent structure of the rights expression language of OMA DRM v1.0, but defined as a subset of MPEG-21 REL, and not as a subset of ODRL. This research work was presented in [7], [8].

The main concept in this equivalent structure is the r:license element that conceptually is a container of a r:grant or a r:grantgroup elements. A r:grant element conveys to someone the sanction to exercise a right (mx:play, mx:execute or mx:print are considered) against a resource. In this case, a resource is represented with the r:digitalResource element.

A r:license element also contains a r:otherinfo element that it is useful to include OMA DRM REL v1.0 information not considered by MPEG-21 REL, and it provides meta information about the rights.

The conditions considered are: sx:exerciseLimit that specifies the number of allowed exercises of a certain right, validityInterval that specifies an interval of time within which a right can be exercised and validityIntervalDurationPattern that specifies a period of time within which a right can be exercised.

```
<!ELEMENT r:license ( (r:grantgroup|r:grant), r:otherinfo? )>
<!ATTLIST r:license
       xmlns:r CDATA #FIXED
       "urn:mpeg:mpeg21:2003:01-REL-R-NS"
       xmlns:dsig CDATA #FIXED
       "http://www.w3.org/2000/09/xmldsig#"
       xmlns:mx CDATA #FIXED
       "urn:mpeg:mpeg21:2003:01-REL-MX-NS"
       xmlns:sx CDATA #FIXED
       "urn:mpeg:mpeg21:2003:01-REL-SX-NS">

<!ELEMENT r:grantgroup (r:grant+)>
<!ELEMENT r:grant
       ((mx:play|mx:execute|mx:print)?,
         r:digitalResource,
         r:allConditions?)>

<!ELEMENT mx:play EMPTY>
<!ELEMENT mx:execute EMPTY>
<!ELEMENT mx:print EMPTY>

<!ELEMENT r:digitalResource (r:nonSecureIndirect) >
<!ELEMENT r:nonSecureIndirect EMPTY>
```

```
<!ATTLIST r:nonSecureIndirect URI CDATA #IMPLIED>

<!ELEMENT r:allConditions
       (sx:exerciseLimit?,
        validityInterval?,
        validityIntervalDurationPattern?)>
<!ELEMENT sx:exerciseLimit (sx:count)>
<!ELEMENT sx:count (#PCDATA)>
<!ELEMENT r:validityInterval (r:notBefore?, r:notAfter?)>
<!ELEMENT r:notBefore (#PCDATA)>
<!ELEMENT r:notAfter (#PCDATA)>
<!ELEMENT sx:validityIntervalDurationPattern (sx:duration)>
<!ELEMENT sx:duration (#PCDATA)>
<!ELEMENT r:otherinfo (version?,KeyValue?)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT KeyValue (#PCDATA)>
```

Figure OMA-based MPEG-21 REL DTD

14.1.5.2 Interoperability between MPEG-21 REL and OMA DRM REL v2.0

In this section, we introduce different tables with XML equivalences, between the OMA DRM REL v2.0 and the MPEG-21 REL subset that will lead us to achieve interoperability between the MPEG-21 REL subset for the mobile domain and OMA DRM REL v2.0 specification. This work has been presented in [9], [10].

We use different models to group the XML equivalences according to their functionality and license structure. The models described in this section are: Basic equivalences, Rights and Conditions. They define the elements that form the subset of MPEG-21 REL that fulfils OMA DRM REL v2.0 specification.

Basic equivalences

The basic equivalences constitute the basis for licenses and include the necessary elements in any license. The OMA DRM REL <rights> and <asset> elements are represented with the MPEG 21 REL <license> and <digitalResource> elements. The OMA <context> element provides meta information about the rights, and is represented with the MPEG-21 <otherinfo> element.

Table Basic model

OMA DRM REL v2.0	OMA-based MPEG-21 REL
<o-ex:rights>	*<r:license>*
<o-ex:context> <o-dd:version>2.0</o-dd:version> <o-dd:uid>RightsObjectID</o-dd:uid> </o-ex:context>	*<r:otherinfo> <version>1.0 </version> <uid>RightsObjectID</uid> </r:otherinfo>*
<o-ex:asset> <o-ex:context> <o-dd:uid>ContentID </o-dd:uid> </o-ex:context> </o-ex:asset>	*<r:digitalResource> <r:nonSecureIndirect URI='ContentID' /> </r:digitalResource>*

Rights

In the following table are introduced the MPEG-21 REL rights equivalent to the rights specified in OMA DRM REL. The <display> and <play> elements are represented with the <play> element, the <export - move> element with the <move> element and the <export - copy> element with the <adapt> element and <prohibitedAttributeChanges> elements.

Table Rights model

OMA DRM REL v2.0	OMA-based MPEG-21 REL
<o-dd:display/>	*<mx:play />*

<o-dd:play/>	<mx:play />
<o-dd:execute/>	<mx:execute />
<o-dd:print/>	<mx:print />
<oma-dd:export oma-dd:mode="move"> <o-ex:constraint> <oma-dd:system> <o-ex:context> <o-dd:version> 1.0 </o-dd:version> <o-dd:uid> XYZ </o-dd:uid> </o-ex:context> </oma-dd:system> </o-ex:constraint> </oma-dd:export>	<mx:move/> <r:digitalResource> <r:nonSecureIndirect URI="ContentID"/> </r:digitalResource> <r:allConditions> <mx:destination> <r:keyHolder> <r:info> <version>1.0</version> <uid>XYZ</uid> </r:info> </r:keyHolder> </mx:destination> </r:allConditions>
<oma-dd:export oma-dd:mode="copy"> <o-ex:constraint> <oma-dd:system> <o-ex:context> <o-dd:version> 1.0 </o-dd:version> <o-dd:uid> XYZ </o-dd:uid> </o-ex:context> </oma-dd:system> </o-ex:constraint> </oma-dd:export>	<mx:adapt/> <r:digitalResource> <r:nonSecureIndirect URI="ContentID1"/> </r:digitalResource> <mx:prohibitedAttributeChanges> <set definition= "urn:mpeg:mpeg21:2003:01- RDD-NS:2346"/> <set definition= "urn:mpeg:mpeg21:2003:01- RDD-NS:2347"/> </mx:prohibitedAttributeChanges> <r:keyHolder> <version>1.0</version> <uid>XYZ</uid> </r:keyHolder>

Conditions

The following tables introduce different kinds of MPEG-21 REL conditions equivalent to the ones specified in OMA DRM REL.

One kind of conditions represented are time conditions. The <datetime> element represented in MPEG-21 REL with the <validityInterval> element specifies an interval of time within which a right can be exercised. The <interval> represented in MPEG-21 REL with the <validityIntervalDurationPattern> element specifies a period of time within which a right can be exercised. Finally, the <accumulated> element represented in MPEG-21 REL with the <validityTimeMetered> specifies the maximum period of metered usage time during which the rights can be exercised.

Table Time conditions model

OMA DRM REL v2.0	OMA-based MPEG-21 REL
<o-ex:constraint> <o-dd:datetime> <o-dd:start>... </o-dd:start> <o-dd:end>... </o-dd:end> </o-dd:datetime>	<r:allConditions> <r:validityInterval> <r:notBefore>...</r:notBefore> <r:notAfter>...</r:notAfter> </r:validityInterval> </r:allConditions>

</o-ex:constraint>	
<o-ex:constraint> <o-dd:interval> </o-dd:interval> </o-ex:constraint>	<r:allConditions> <sx:validityIntervalDurationPattern> <sx:duration>...</sx:duration> </sx:validityIntervalDurationPattern> </r:allConditions>
<o-ex:constraint> <o-dd:accumulated> PT10H </ o-dd:accumulated > </o-ex:constraint>	<r:allConditions> <sx:validityTimeMetered> <sx:duration>PT10H</sx:duration> </sx:validityTimeMetered> </r:allConditions>

The next table introduces the rest of MPEG-21 REL conditions considered in the mobile subset we are defining equivalent to the ones specified in OMA DRM REL. The <count> element represented in MPEG-21 REL with the <exerciseLimit> element specifies the number of allowed exercises. The <timed-count> element specify the number of times a permission may be granted over an asset or resource, with the addition of an optional timer attribute. This timer attribute specifies the number of seconds after which the count state can be reduced. As the timer attribute is not specified in MPEG-21 REL, we have defined the <exerciseLimitTime>, that consist of <count> and <duration> elements. The <individual> represented in MPEG-21 REL with the <keyHolder> element specifies the individual to which content is bound. The <system> represented in MPEG-21 REL with the <renderer> element specifies the target system to which DRM Content and Rights Objects can be exported.

Table Other conditions model

OMA DRM REL v2.0	OMA-based MPEG-21 REL
<o-ex:constraint> <o-dd:count > 1 </o-dd count> </o-ex:constraint>	<sx:exerciseLimit> <sx:count>1</sx:count> </sx:exerciseLimit>
<o-ex:constraint> <o-dd:timed-count timer=”30”>1 </o-dd:timed-count> </o-ex:constraint>	<r:otherinfo> <exerciseLimitTime> <sx:count>1</sx:count> <sx:duration>30 </ sx:duration> </exerciseLimit> </r :otherinfo>
	<r:grant licensePartId="Asset-1"> <r:allConditions> <sx:exerciseLimit> <sx:count>1</sx:count> </sx:exerciseLimit> </r:allConditions> </r:grant licensePartId="Asset-1"> <r:otherinfo> <grant licensePartIdRef="Asset-1"> <exerciseLimitDuration> 30 </exerciseLimitDuration> </grant> </r:otherinfo>
	<sx:exerciseLimit> <r:serviceReference licensePartIdRef="externalService"/> <sx:count>1</sx:count> </sx:exerciseLimit>
<o-ex:constraint> <o-dd:individual> <o-ex:context> <odd:uid> XYZ	<r:grant> <r:keyHolder> <r:info> <uid>XYZ</uid>

</odd:uid> </o-ex:context> </o-dd: individual> </o-ex:constraint>	</r:info> </r:keyHolder> </r:grant>
<o-ex:constraint> <oma-dd:system> <o-ex:context> <odd:uid> XYZ </odd:uid> </o-ex:context> </oma-dd system> </o-ex:constraint>	<mx:renderer> <r:keyHolder> <r:info> <uid>XYZ</uid> </r:info> </r:keyHolder> </mx:renderer>

14.1.6 PAR and Licenses: Updates on the work performed

The work performed in this task is mainly based on the transcoding of expressions described in MPEG-21 Rights Expression Language to OMA DRM Rights Expression Language and the other way around. To do so, we have used as a starting point the work done in WP4.5 on MPEG-21 REL and WP4.7 on OMA DRM REL. In these work packages several tools, libraries and databases have been defined and implemented in order to support these languages.

For the MPEG-21 REL case, we have defined an structure which allows the expression of most of the conditions present in the language, by means of a very flexible database structure. The current conditions implemented are those present in the requirements of the AXMEDIS project, but it is planned to add more as the project evolves and some more conditions, not present in the language, have to expressed in to support current business models.

For the OMA DRM REL case, a database and a library for supporting it has been defined. The current conditions implemented are the ones defined by Open Mobile Alliance on its version 2.0 of OMA DRM REL. Two conditions, system and individual conditions have not been implemented due to problems in the interpretation of the standard (its behaviour is not completely described). There is also missing a better control of what to do if there are conditions defined at different levels, as the behaviour of an application using this license is not completely described in the standard

14.1.6.1 Summary of UML and Relational models defined for MPEG-21 REL

The following figures show the UML and relational models for defining MPEG-21 REL rights expressions.

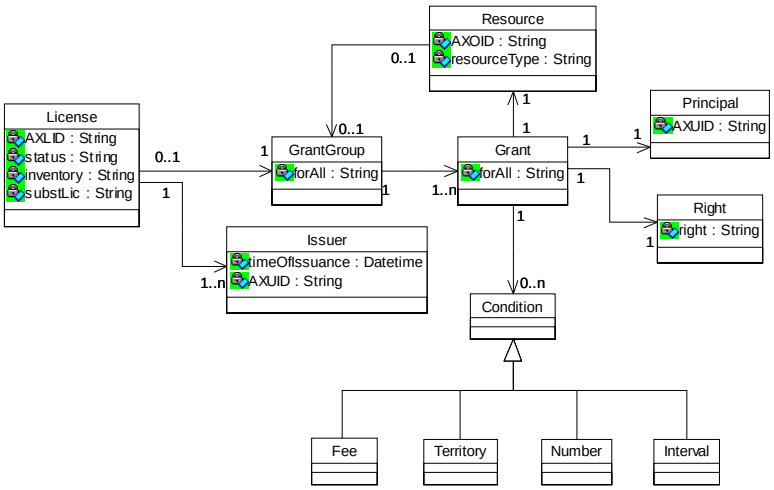


Figure- UML model

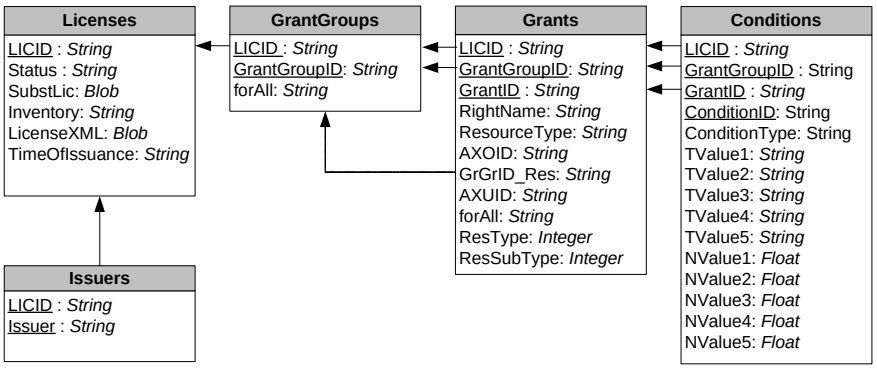


Figure – Relational model

14.1.6.2 Summary of UML and Relational models defined for OMA DRM REL

The following figures show the UML and relational models for defining OMA DRM REL rights expressions.

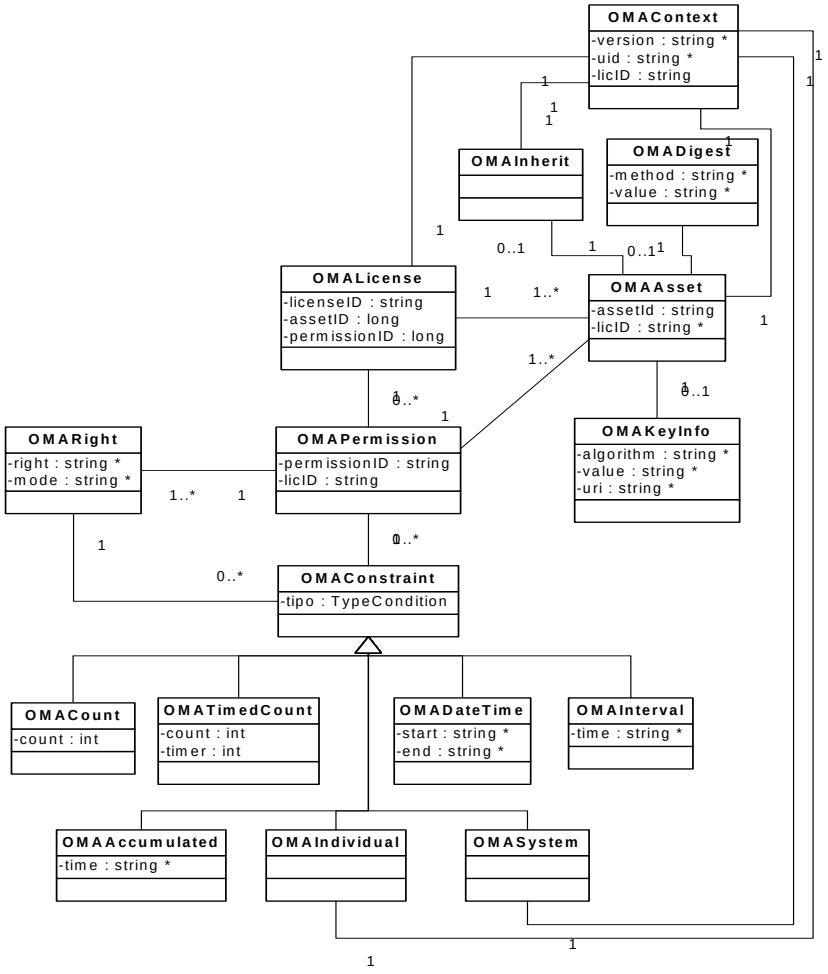


Figure - OMA DRM REL UML Class Diagram

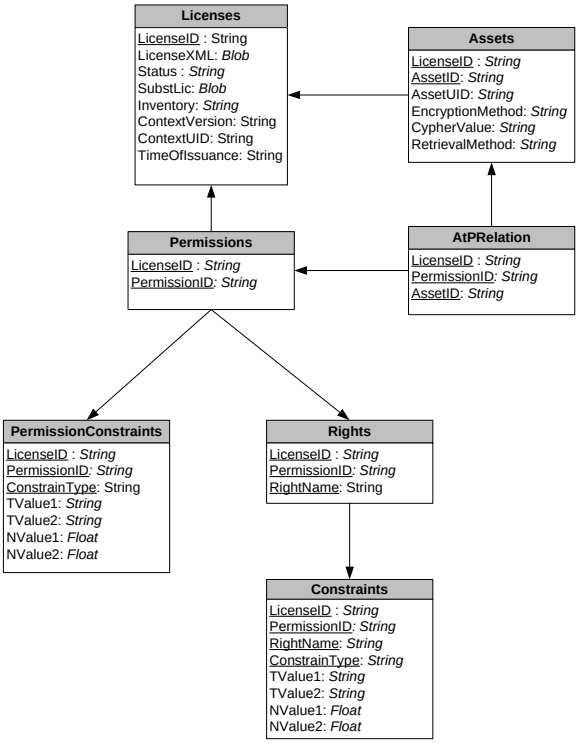


Figure - OMA DRM REL ER Diagram

14.1.6.3 Translation of rights expressions

Our first approach to the transformation of licenses form MPEG-21 REL to OMA and the other way around was to use XSLT (Style Sheets). Using this technique the structure of the licenses being transformed is very limited and restricted (this has been briefly described in previous sections of this deliverable). So we tried to do it in a different way, taking advantage of the UML and the entity relationship model to define two different alternatives to solve the problem.

Based on UML

OMA DRM and MPEG-21 REL express a license format, nevertheless the structure of these two kind of licenses is quite different. So it is not possible to use a syntactic approach to perform the transformation but a semantic one based on the UML model.

The *License* class in the MPEG-21 diagram is equivalent to *OMALicense* class, and it is the object that holds the relationship of the model.

Another similarity is found at the *Condition* and *OMAConstraint* classes, which can be translated directly.

The critical point is that in MPEG-21 each *Grant* stores information about one right, one resource, one right holder and the conditions of this relationship. On the other hand OMA DRM stores all that information in different classes with different relationships (*OMAAsset*, *OMAPermission* and *OMARight*).

In order to translate the OMA DRM to MPEG-21 REL model, we have to generate a *Grant* for each relation that involves one *OMAAsset* with one *OMAPermission* and one *OMARight*. Moreover, we have to take into account that a set of *OMAConstraints* can affect one specific *OMARight* or the *OMAPermission* (that involves all the rights related with it).

Note that exist some elements in both languages that cannot be exactly translated and which have to be transformed not in the syntactic way but semantic (For example the TimedCount Constraint from OMA, or the adapt Right from MPEG-21).

If you want to translate from MPEG-21 REL to OMA DRM, you have to take care about the number of *OMAPermissions* that would be generated. To do that you have to group the *Grants* by the applied set of Conditions, in order to generate the minimum number of *OMAPermissions* and *OMARights*.

Based on a relational model

The other way to translate the licenses from one language to the other is to use the Relational model. In this case, we can use SQL sentences like selects and inserts to perform the transformations.
To transform from OMA DRM ER model to the MPEG-21 one you can use two intermediate tables, which contains both the grants and the conditions table. These two intermediate tables can be fulfilled with a complex select that involves the join of Permissions, Assets, and Rights tables on OMA DRM ER model. And these two intermediate tables can be directly transformed to the Grants and Conditions tables of the MPEG-21 ER model.
To transform the information from the MPEG-21 ER model to the OMA DRM one, we cannot grant that the resulting OMA license has the optimal structure. This is because every Grant in the MPEG-21 will be translated to a Permission with only one asset and one right.
In the case we want to obtain the optimal OMA license from the MPEG-21 one, we must find which Grants could belong to the same Permission. These ones would have the same set of rights with the same set of conditions for every right for a set of resources. This cannot be done using only SQL sentences, and requires a quite complex algorithm. Because of that we prefer to use the UML model to try to perform this transformation.

14.1.6.4 Major features accessible

The following features regarding transcoding of PAR and Licenses are currently accessible:

- License and PAR Database for MPEG-21 REL
- Libraries to support the creation and management of license and PAR in MPEG-21 REL format

To be uploaded, but also available:

- License Database for OMA DRM REL
- Libraries to support the creation and management

The available work is related to other workpackages. MPEG-21 REL licenses work has been done and described in WP4.5 while OMA DRM REL work has been done and partly described in WP4.7. The work done in this workpackage relates both aspects, as it is needed to have a deeper understanding of the source and destination rights expressions in order to transcode them.

14.1.7 PAR and Licenses: Work to be done

The work to be done involves several tasks:

- Finish implementation and perform integration of the tools described for transcoding PAR and Licenses. The translation mechanism defined in this deliverable have to be effectively implemented. Some tests have been done, but a complete implementation is missing.
- Study of new versions and profiles of rights expression languages in order to improve the transcoding mechanisms. MPEG-21 REL is defining profiles to support different scenarios and business models. Also a new version of ODRL has been deployed, so its features have to be studied in order to provide a more complete transcoding mechanism.
- Contribute to the corresponding standaridation bodies and forums with the results of the above tasks, in order to allow between interoperability among different rights expression languages.

15 Transcoding Metadata (UNIVLEEDS)

This module provides a collection of algorithms and tools for adaptation of XML metadata. The main adaptation functions needed by the AXMEDIS Framework could be summarised in:

- Scaling metadata by filtering elements
  - This can be done by specifying look-up tables (including XSLT) to define the valid/invalid elements and processing the adaptation
- Adapting field (could be different name or size or others)
  - Xerces can also use a given schema to validate the elements. With this validation function, the elements can be detected for adaptation. Look-up tables (including XSLT) have to be setup in order to adapt metadata from one standard to the other.

For XML metadata transcoding, the *Xerces Libraries* are used to parse a given piece of XML data.

The AxMetadata Model has been developed to provide the functionality for the transcoding of AXMEDIS metadata. Current functionalities include:

- Loading, saving, writing to string and parsing an XML document using the Xerces SAX2 implementation
- Changing the Element name and Uri without validation
- Changing the Element values (currently there is no validation with respect to date and time elements
- Changing the Element Attribute values name, Uri and values without validation

Demonstrating the current functionality can be viewed in the Metadata Editor (see deliverable DE4.1.1 content modelling and managing).

15.1 Metadata Transcoding: State of the art

The transcoding of metadata has two levels of adaptation that need to be considered. The first is transcoding the metadata to represent the adaptation applied to the object such as scaling an object, reformatting etc. This requires methods to adapt the metadata to represent the object in its new state. The second transcoding depends on the devices to use the object. In this scenario, for small memory devices, the metadata may need to be scaled and tables for this process are required to be devised.

There are no specific tools for the adaptation of metadata in this manner, however libraries are being developed such as Xerces, Xalan, and Expat for the parsing and handling of XML documents. This is an ongoing research and development process. Xerces and Xalan provide libraries in both C++ and Java and initial support of Schemas. The Xerces libraries provide methods for the programmatic generation and validation of XML and customisable error handlers (see AXMEDIS FW Specifications in DE 3.1.2d). Xalan libraries are an XSLT processor for transforming XML documents to HTML or another XML document types. These methods can be utilized in the adaptation of the metadata for not only transcoding the format changes info in the AxInfo elements but in the scaling of the metadata for different clients.

Validation is an important issue that needs to be considered where adapted metadata is valid after transcoding. Due to the complex nature of AXMEDIS object, there may be one or more metadata sections with different schemas, including Dublin Core, AXInfo and Mpeg7 and the next development method is to devise methods to solve these problems.

15.2 Metadata Transcoding: The problems

Some of the desired functionality has not been resolved:

- Adding and deletion of elements with validation to keep the metadata as valid XML using the schemas.
- Adding and deletion of element attributes with schema validation.

- Moving elements to another part of the XML structure, this may or may not need to reallocate children elements.

15.3 Metadata Transcoding: Work performed

The following classes have been developed for the adaptation of XML AXMEDIS Metadata providing functionalities for loading, saving, parsing, manipulating and writing XML files.

The implementation was performed using C++ MSVC7 and supported by wxWidgets ver. 2.4.2; XERCES 2.6.0 libraries and Xalan 1.9. For the transcoding of metadata, the following classes were developed for the first prototype to adapt generic XML metadata elements.

The following image shows the stand alone Metadata Mapper GUI linking XML nodes to generate the XSLT

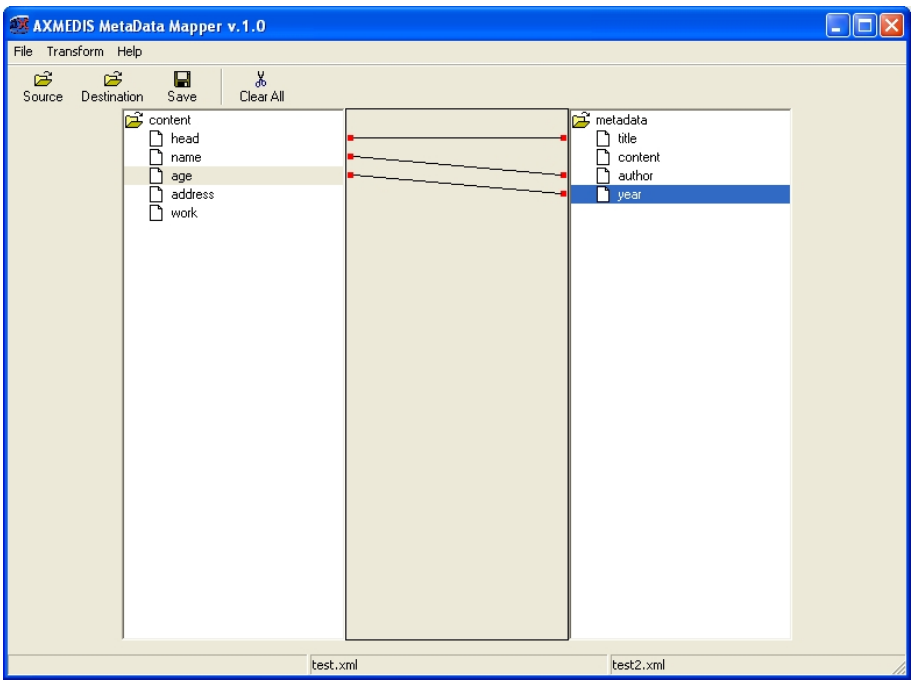


Figure: Snapshot of stand alone Metadata Mapper GUI with linked nodes to map transformations

The following image shows the transformation options available in the Metadata Mapper GUI

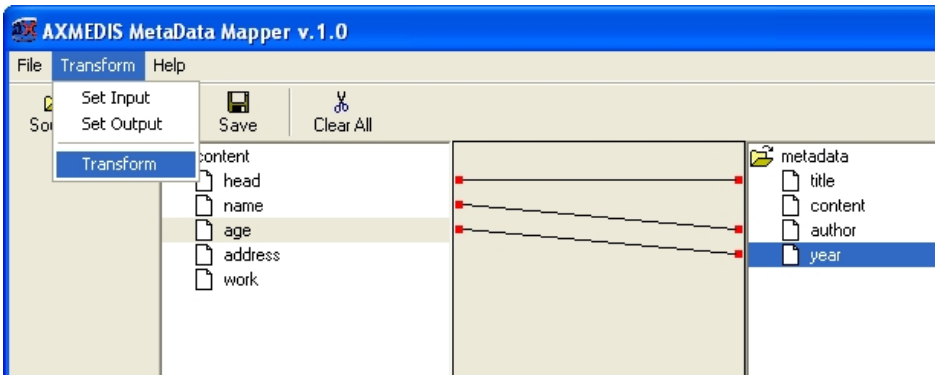


Figure: Snapshot of transformation options available in the Metadata Mapper GUI

The following code examples demonstrates the use of the JS_MetadataMapper functions:

```
// Example 1: create a new file containing transformed metadata
var mdm = new MetaDataMapper();
mdm.LoadXSLFile("style.xsl");
mdm.TransformFile("input.xml","output.xsl");

// Example 2: create a new file containing transformed metadata
var input="<?xml……";             // input xml string
var style="<?xml……";             // stylesheet xml string
var output;                      // output xml string
var mdm = new MetaDataMapper();
mdm.LoadXSL(style);
mdm.Transform(input,output);
```

Major Features accessible:

- Metadata Mapper GUI (stand alone)
  - Load input XML
  - Load destination XML file
  - Connect nodes for transformation
  - Generate XSLT
  - Transform Source XML to a destination file using the Generated XSLT
- JS_MetadataMapper (JS_MDM)
  - Create Metadata Mapper Object
    - *MetaDataMapper( )*
  - Load XSLT file either from file or string for metadata trancoding
    - *LoadXSLFile(string xsltFilePath)*
    - *LoadXSL(string xsltString)*
  - Transform metadata from a file or string to an output file or string
    - *TransformFile(string inString, string outString)*
    - *TransformFile(string infile, string outfile)*

Work to be done:

- Metadata Mapper GUI
  - Integrate GUI to the AXCP and AxEditor to generate XSLT files
  - Load using the AXOM with namespaces (Needed for the integration with the Rule Editor and AxEditor)
  - Generating XML Source and Destination XML files from Schemas with user selection of metadata elements
  - Multiple node connections for transformation
- JS_MetadataMapper (JS_MDM)
  - Load XSLT file either from file or string for metadata trancoding
  - Do namespace checking for transformations of Dublin Core and AxInfo

16 Bibliography

- [Bes01] F. Bes, M. Jourdan, F. Khantache "A Generic Architecture for Automated Construction of Multimedia Presentation", Amsterdam, The Nederlands, 2001.
- [Bol99] S. Boll, W. Klas, J. Wandel "A Cross-Media Adaptation Strategy for Multimedia Presentations", Ulm, Germany, 1999.
- [Bol01] S. Boll, W. Klas "ZYX - a multimedia document model for reuse and adaptation of multimedia content", Wien, Austria, 2001.
- [Bol03] S. Boll "MM4U - A framework for creating personalized multimedia content", 2003.
- [Bul98] D.C.A. Bulterman "User-centered abstractions for adaptive hypermedia presentations", Bristol, United Kingdom, 1998.
- [Bul05] D.C.A. Bulterman, L. Hardman "Structured Multimedia Authoring", Amsterdam, The Nederlands, 1993-2005.
- [Gei03] J. Geigel, A. Loui, "Using Genetic Algorithms for Album Page Layouts", IEEE Multimedia, Oct.-Dec. 2003
- [Gol89] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Kluwer Academic Publishers, Boston, MA, 1989.
- [Goo94] E. D. Goodman, A. Y. Tetelbaum, V. M. Kureichik, "A Genetic Algorithm Approach to Compaction, Bin Packing, and Nesting Problems", technical report, Michigan State University, 1994.
- [Har99] L. Hardman, J. van Ossenbruggen, K. Sjoerd Mullender, L. Rutledge, D.C.A. Bulterman "Do you have the time? Composition and linking in time-based hypermedia", Darmstadt, Germany, 1999.
- [Jou98] M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismail, L. Tardif "Madeus, an authoring environment for interactive multimedia documents", Bristol, UK, 1998.
- [Lem03] T. Lemlouma, N. Layaida "Adapted content delivery for different contexts", 2003.
- [Oli99] A. Oliver, N. Monmarchè, G. Venturini, "Interactive Design of Web Sites with a Genetic Algorithms", University of Tours, France, 1999.
- [Pih03] K. Pihkala "Extensions to the SMIL Multimedia Language", Helsinki, Finland, 2003.
- Ponce-Perez, A. Perez-Garcia, V. Ayala-Ramirez, "Bin-packing using genetic algorithms", Proceedings of the 15th International Conference on Electronics, Communications and Computers (CONIELECOMP 2005).
- [Ric97] J. R. Ricketts, "Optimization of newspaper display advertising layout: using a genetic algorithm with a novel rearrangement step", Master of Applied Science, RMIT University, 1997.
- [Roi03]C. Roisin, V. Kober, V. Quint, P.Genevès, P. Navarro "Editing SMIL with Timelines", 2003.
- [Rut98] L. Rutledge, L. Hardman, J. van Ossenbruggen, D.C.A. Bulterman "Structural Distinctions Between Hypermedia Storage and Presentation", Bristol, United Kingdom, 1998.
- [Thu02] T.T. Thuong, C. Roisin "A Multimedia Model Based on Structured Media and Sub-elements for Complex Multimedia Authoring And Presentation", 2002.
- [Van01]J. van Ossenbruggen, J. Geurts, F. Cornelissen, L. Hardman, L. Rutledge "Towards second and third generation web-based multimedia", Hong Kong, 2001.

- [Vil01] L. Villard "Authoring transformations by direct manipulation for adaptable multimedia presentations", Atlanta, Georgia, USA, 2001.
- [W3C05] W3 Consortium (Bulterman et al.) "Synchronized Multimedia Integration Language (SMIL 2.0)", 2005.
- [Wei94] L. Weitzman, K. Wittenburg "Automatic presentation of multimedia documents using relational grammars", San Francisco, CA, USA, 1994.
- [Zha02] K. Zhang, D.Q. Zhang, Y. Deng "Graphical Transformation of Multimedia XML Documents" Red Bank, NJ, USA, 2002.
- [Zha05] K. Zhang, J. Kong, M. Qiu, G. Song "Multimedia layout adaptation through grammatical specifications", Dallas, TX, USA, 2005.

17 Other reference

- [1] ISO/IEC, ISO/IEC IS 21000-5 – Rights Expression Language.
- [2] XrML, http:// www.xrml.org/.
- [3] Open Digital Rights Language (ODRL). http://odrl.net.
- [4] OMA DRM Rights Expression Language, OMA-Download-DRMREL-V2_0-20041210-C. 10 December 2004.
- [5] XML Encryption Syntax and Processing, W3C Candidate Recommendation 10 December 2002, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
- [6] XML Signature Syntax and Processing, W3C Recommendation 12 February 2002, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
- [7] Prados, J., Rodríguez, E., Delgado, J., Profiles for interoperability between MPEG-21 REL and OMA DRM, CEC 2005, Munich (Germany), 19 – 22 July 2005, ISBN 0-7695-2277-7.
- [8] Delgado, J., Prados, J., Rodríguez, E., Interoperability between MPEG-21 REL and OMA DRM: A profile?, ISO/IEC JTC1/SC29/WG11 MPEG2005/M11580, January 2005.
- [9] Delgado, J., Prados, J., Rodríguez, E., Interoperability between different Rights Expression Languages and Protection Mechanisms, AXMEDIS 2005, Florence (Italy), 30 November – 2 December 2005, To be published.
- [10] Delgado, J., Prados, J., Rodríguez, E., A subset of MPEG-21 REL for interoperability with OMA DRM v2.0, ISO/IEC JTC 1/SC 29/WG 11/ M11893, April 2005.