Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE5.1.1
AXMEDIS Framework Infrastructure

Version: 2.4-public
Date: 8/9/2005
Responsible: DSI

Project Number: IST-2-511299 Project Title: AXMEDIS Deliverable Type: private Visible to User Groups: Yes Visible to Affiliated: Yes Visible to Public: Yes

Deliverable Number: DE5.1.1 Contractual Date of Delivery: M12 (end of August 2005) Actual Date of Delivery: 8 Sept. 2005 Work-Package contributing to the Deliverable: WP5 Task contributing to the Deliverable: all of WP5 Nature of the Deliverable: document, report Author(s): DSI, FUPF, EXITECH, EPFL, CRS4, UNIVLEEDS, etc.

Abstract:
This document describe the AXMEDIS Framework

Keyword List:
Content production, framework, libraries, programme and publication, Certifier and supervisor, plug in editor.

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

Table of Contents

1 Executive Summary and Report Scope (mandatory)

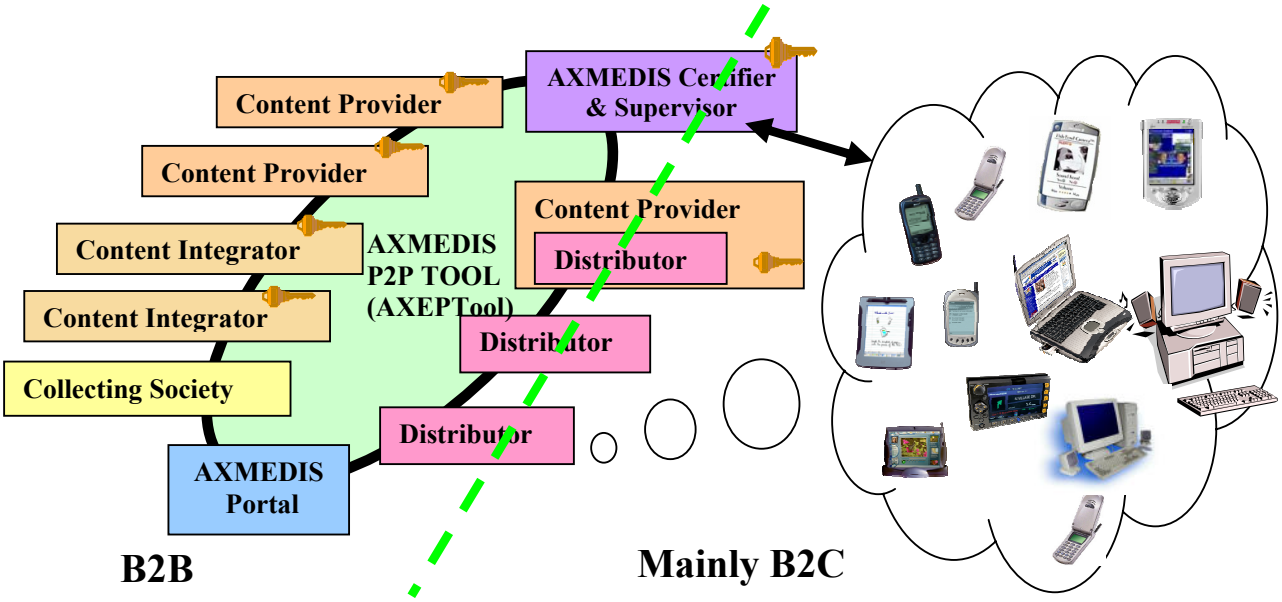
This document includes 4 Annexes:
AXMEDIS-DE5-1-1-Annex DMP-MPEG21-analysis-v1-13.doc
AXMEDIS-DE5-1-1-Annex-AXCS-Specification-v3.doc
AXMEDIS-DE5-1-1-Annex-EPFL-MP4-Player.doc
AXMEDIS-DE5-1-1-Annex-IM1-OSMO-Relazione-v13.doc

2 AXMEDIS General Architecture

The AXMEDIS digital content and content components (in the following, AXMEDIS content in general) will have a specific format capable of integration inside any kind of cross media format (video, images, animations, document, audio, etc.), adding metadata, identification, classification, categorization, indexing, descriptors, annotation, relationships and play activities and protection aspects. The format will permit the combination of content components, their secure distribution, etc., in the respect of the copyright laws, supporting a large variety of DRM rules and models according to concepts of interoperability among DRMs (mainly, but not only, based on MPEG-21, with both binary and XML low level formats). Within the AXMEDIS content any type of cross media content can be included from simple multimedia files to games, software components, for leisure and entertainment, infotainment, etc.

The General Architecture of AXMEDIS is represented in the next figure, which highlights both:

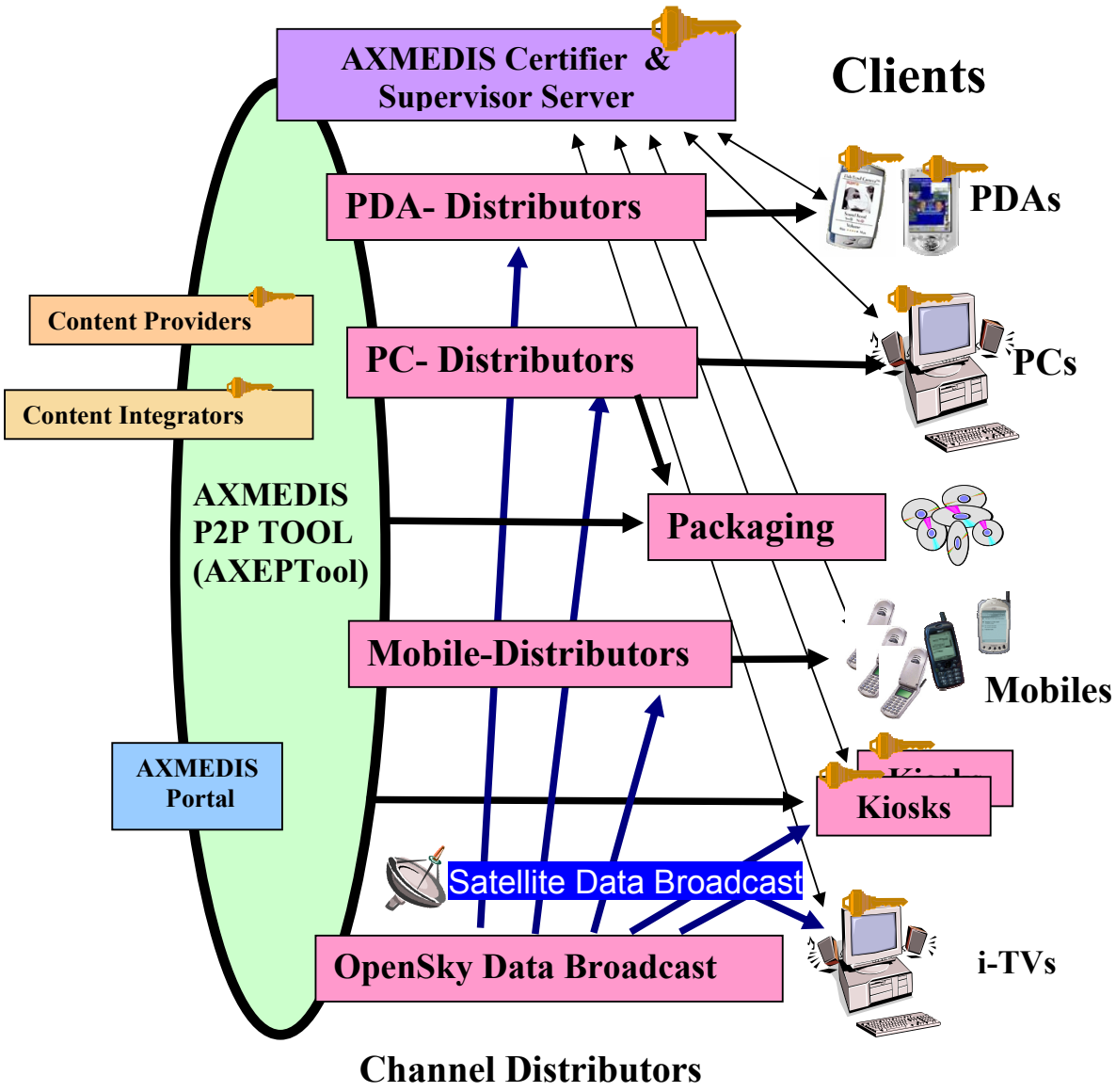
- **production** of AXMEDIS digital content and content components in connection with the AXMEDIS P2P tool (AXEPTool) that follows business mechanisms of B2B and support DRM with a certification authority (AXMEDIS Certifier and Supervisor). This can be connected to the Collection Societies as well as to each Content Provider and to Distributors if needed;
- **distribution** of AXMEDIS digital content towards clients via specific distributors that realize the last level of the distribution chain. This last level can also support a B2B transaction if the distribution is targeted at institutions. Also at this level the sharing via mechanisms of P2P is allowed and stimulated. This will not invalidate the protection model of AXMEDIS DRM.



AXMEDIS General Architecture, mainly B2B side

The standard distribution channel is today a single distribution path for each type of content, and often, multiple proprietary systems of representation for the same content. The definition of distribution channel editorial formats would provide one way, unified and rock-solid content format for multipurpose applications. Alternative solutions support multichannel distribution by using an XML model of content into the Content management systems of the content provider that also include multiple transcoding engines for transforming the XML model of content into the format suitable for the channel. This approach is not flexible enough since the transcoding of content at the source strictly limits the management of Digital Rights. In fact, in models such as CONTESSA the DRM can be applied only to the content in its final version. This creates key problems for the content providers since the content distributors are entitled to receive unprotected content. This is almost unacceptable in most cases.

In AXMEDIS, the channel distributors may maintain their distribution process. They can continue to use the same format for reaching the final users. In AXMEDIS, the content is distributed by using the P2P tool, namely AXEPTool, by using an evolution of the MPEG-21 format, with the AXMEDIS contribution. This content will easily contain and deliver MPEG-4, MPEG formats, PDF, HTML, SVG, images, documents, videos, audio file, etc. (in open standard format for continuation, without the use of proprietary technologies) on demand and for all platforms according to the final format produced by the Distributor. The received content will be formatted by using AXMEDIS tools on the basis of specific editorial formats.



AXMEDIS General Architecture, mainly B2C

The possible Channel Distributors have a large variety of capabilities, they are both of pull and push, and may include off-line and on-line connection from the client to the distributor.

Channel Distributors are interested in:

- Getting AXMEDIS content and components from the Content Providers and using them for distributing content via their channels for redistribution for both B2B and B2C transactions.

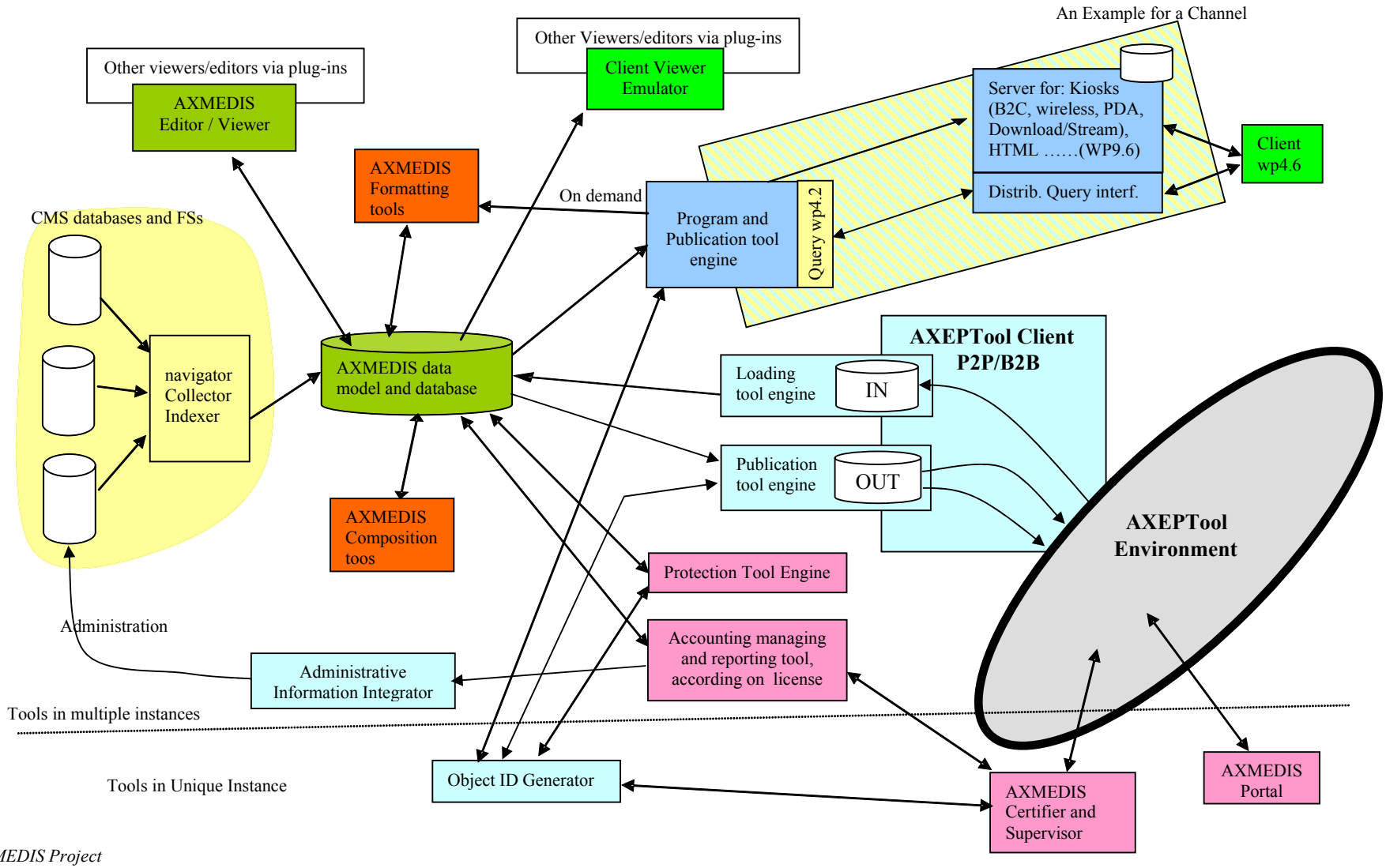
- Collecting AXMEDIS contents in a local database for preparing the production content Programme that is the agenda/menu proposed to the customers and final users.
- Using AXMEDIS content for creating attractive content for their customers. For this reason, they need to have the possibility of inspecting content in their internal LAN on a client PC.
- Receiving and satisfying requests from their customers for delivering to them the proposed content
- Receiving and satisfying queries performed by their customers that are looking for specific content. This activity is one of the most interesting added value of the AXMEDIS architecture.
- Getting updated information about the possible content that can be recovered from all Content Providers. This activity is performed via a service of the AXMEDIS portal. The updating of the database of the available content is performed in push via satellite data broadcast with specific policies.
- Accessing statistics produced by the AXMEDIS Certifier and Supervisor about the content usage.

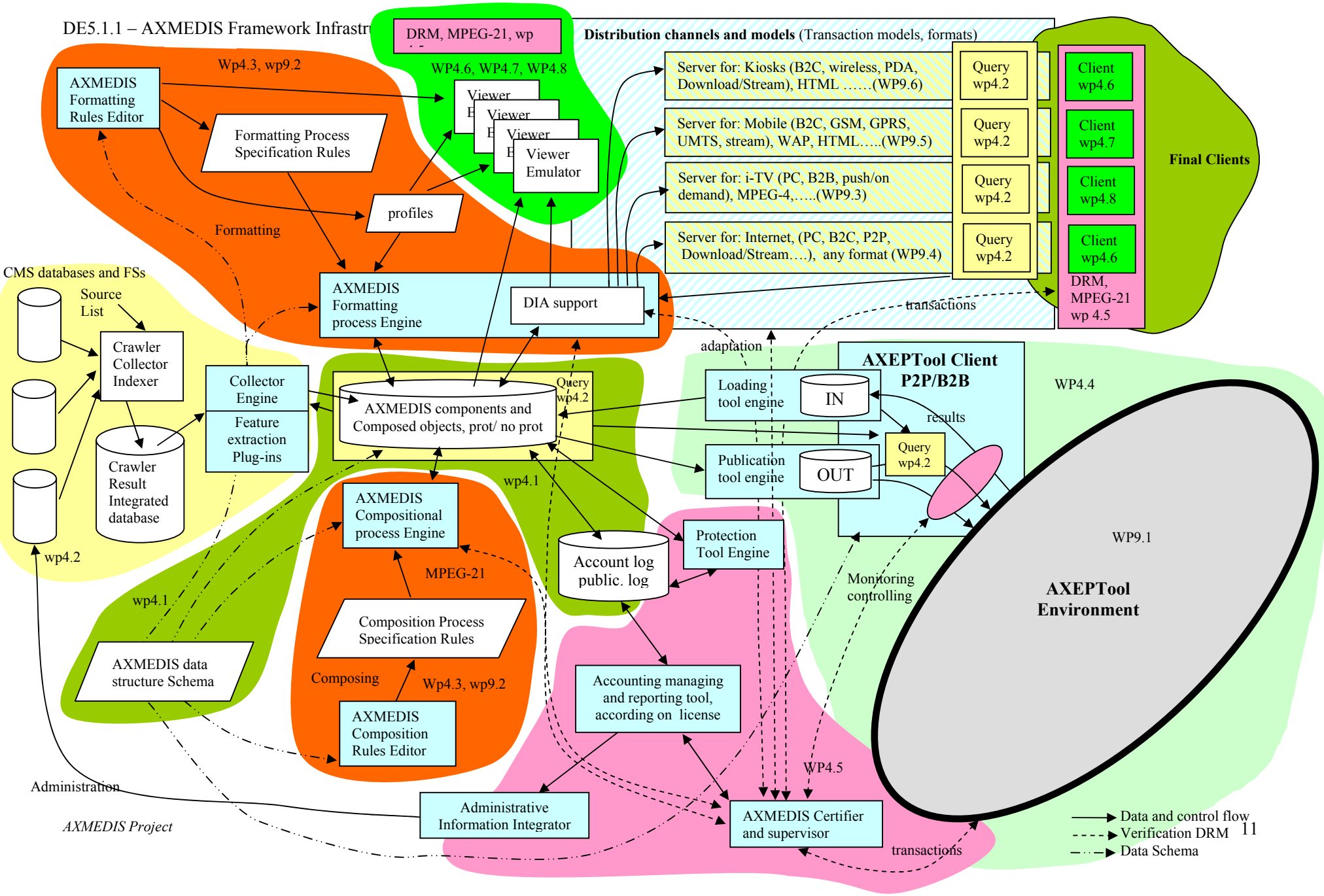
Satellite Data Broadcast It is a content distribution mechanism that permits the distribution of the AXMEDIS content in a very efficient manner. This improves the quality of service of the data delivery process (dependent on broadband availability in client location), and Distributors and also PC users can also rely on Satellite Broadcast. This technology, provided by EUTELSAT's Opensky platform, allows large quantities of data to be pushed via satellite directly on the user's PC without congesting local networks. The use of this technology is completely transparent with regard to the AXMEDIS process and only acts as a cost effective and efficient transport mechanism. The same technology also allows the content providers to bring live multimedia streaming content directly to the user's PC either for free to air content (mainly for marketing purposes) or paying on-demand channels. The pushing mechanism can be used to renovate the catalogue of the Distributors periodically at low cost.
This platform appears to be ideally suited for distributing AXMEDIS content and components. It represents an excellent opportunity for content providers for new business and for accelerating the distribution decreasing their costs.

The satellite distribution channel can be used for several activities of content distribution for both B2B and B2C business models:

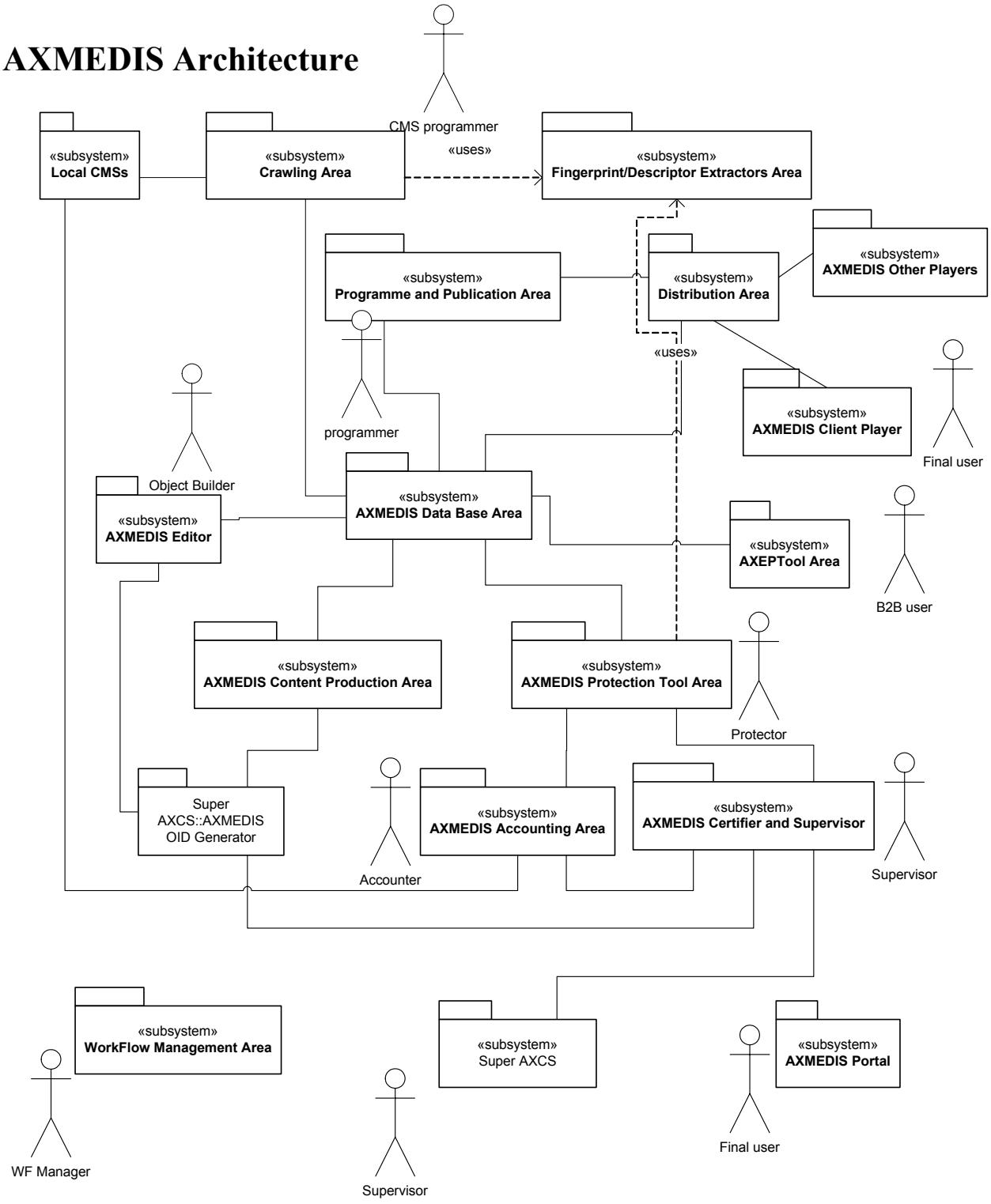
- The push of content
  - updating the AXMEDIS content and components in the databases of the Distributors and of the Providers;
  - updating the general indexing databases of the Distributors with updated information regarding the available AXMEDIS content and components of the Providers;
  - updating the AXMEDIS content on Kiosks;
  - delivering AXMEDIS content on demand directly to the consumers connected to the satellite i-TV according to their interactive requests;
  - delivering AXMEDIS content to the consumers connected to the satellite i-TV-PC according to their selection performed from the programmed content of the day and week.
- The streaming of AXMEDIS content on MPEG-4 on one or more channels for:
  - promoting Content Providers' content;
  - promoting Distributors' services, for example stimulating the acquisition of content in push with a business model based on subscription or pay per view;
  - Creating specific B2B channel with large institutions and consumers.

2.1.1 AXMEDIS General Architecture




DRM, MPEG-21, wp
WP4.6, WP4.7, WP4.8
Wp4.3, wp9.2
AXMEDIS Formatting Rules Editor
Formatting Process Specification Rules
profiles
Formatting
Viewer Emulator
Distribution channels and models (Transaction models, formats)
Server for: Kiosks (B2C, wireless, PDA, Download/Stream), HTML ……(WP9.6)
Server for: Mobile (B2C, GSM, GPRS, UMTS, stream), WAP, HTML…..(WP9.5)
Server for: i-TV (PC, B2B, push/on demand), MPEG-4,…..(WP9.3)
Server for: Internet, (PC, B2C, P2P, Download/Stream….), any format (WP9.4)
Query wp4.2
Client wp4.6
Client wp4.7
Client wp4.8
Client wp4.6
Final Clients
DRM, MPEG-21 wp 4.5
transactions
CMS databases and FSs
Source List
Crawler Collector Indexer
Crawler Result Integrated database
wp4.2
AXMEDIS Formatting process Engine
DIA support
Collector Engine
Feature extraction Plug-ins
AXMEDIS components and Composed objects, prot/ no prot
Query wp4.2
wp4.1
adaptation
AXEPTool Client P2P/B2B
Loading tool engine
IN
Publication tool engine
OUT
results
Query wp4.2
WP4.4
AXMEDIS Compositional process Engine
MPEG-21
Composition Process Specification Rules
Composing
Wp4.3, wp9.2
AXMEDIS Composition Rules Editor
AXMEDIS data structure Schema
wp4.1
Account log public. log
Protection Tool Engine
Monitoring controlling
WP9.1
AXEPTool Environment
Accounting managing and reporting tool, according on license
WP4.5
Administration
Administrative Information Integrator
AXMEDIS Certifier and supervisor
transactions
Data and control flow
Verification DRM
Data Schema

AXMEDIS Architecture



AXMEDIS Framework is composed by the following areas and components:

- **Local CMSs** – represent the set of CMSs which are of interest to be connected to AXMEDIS system via Crawling Area tools.
- **Crawling Area** – includes Focuseek adaptation, Collector Engine, etc… includes a set of plug-ins which allows interaction among proprietary CMSs and AXMEDIS system to migrate digital resources towards AXMEDIS factory
- **Fingerprint Extractors Area** – includes access to a set of algorithms capable of estimating fingerprints (for collection of metadata and for certification) of a large variety of possible resource/content type for extracting a large mount of features. Such algorithms should be designed as a set of plug-ins usable in different AXMEDIS tools.
- **Programme and Publication Area** – includes all the components which allow the interconnection among AXMEDIS networks and databases to the distribution channels for producing programs to public content on the distribution channel. It also allows the management of request for content production/adaptation on demand.
- **Distribution Area** – includes tools to allow content distribution on specific channels and terminals.
- **AXMEDIS Client Players** – includes a reduced version of AXMEDIS Editor only capable to read and show AXMEDIS objects. This Area also includes all the plug-in versions of this tool which allow AXMEDIS object visualization in other players.
- **AXMEDIS Other Players** – includes all the plug-in versions of this tool which allow AXMEDIS object visualization in other players.
- **AXMEDIS Editor** – includes all the modules to create an AXMEDIS objects such as (i) a set of different views (ii) a command manager capable of validating user request (iii) a set of plug-in to use external resource/content editors/viewers (iv) a set of plug-ins to allow the integration of AXMEDIS Editor within other editing applications (v) interface with workflow, etc.
- **AXMEDIS Database Area** – includes AXMEDIS Database manger, access to the content, AXMEDIS Query Support, AXMEDIS Database, etc…
- **AXEPTool Area** – includes all components related to the B2B and P2P communication via AXEPTool application.
- **AXMEDIS Content Production Area** – includes all those tools needed for the automatic content production such as Formatting and Compositional Engine.
- **AXMEDIS Protection Tool Area** – includes DRM, encryption and decryption support, a support which allow communication towards clients and AXMEDIS OID Generator and Certifier and Supervisor, a tool for protecting objects, etc…
- **AXMEDIS OID Generator** – it is the unique responsible of object ID generation and it will elaborate all new-id requests coming from the client or automatic engines.
- **AXMEDIS Accounting Area** – includes a set of tools which allow final producers or distributors to collect information about what has been done on their objects in the AXMEDIS Circuit and thus to produce statistic analysis on content usage and other relevant statistical aspects. It also reports to the administrative side of the CMS the administrative information to prepare the bill at the content user.
- **AXMEDIS Certifier and Supervisor** – it is the responsible of user authentication and software certification, registration and tracking of the activities performed on AXMEDIS objects.
- **Workflow Management Area** – includes support and plug-ins to allow the content flow control, and programming, in all AXMEDIS subsystems.
- **Super AXCS** – is the unique responsible for supervising and collecting general information of monitoring for the actions performed on the content and for the registration of tools, users, etc.
- **AXMEDIS Portal –** includes services for partners, for managing the project, for working together, for supporting take up actions, for providing information during dissemination and demonstration.

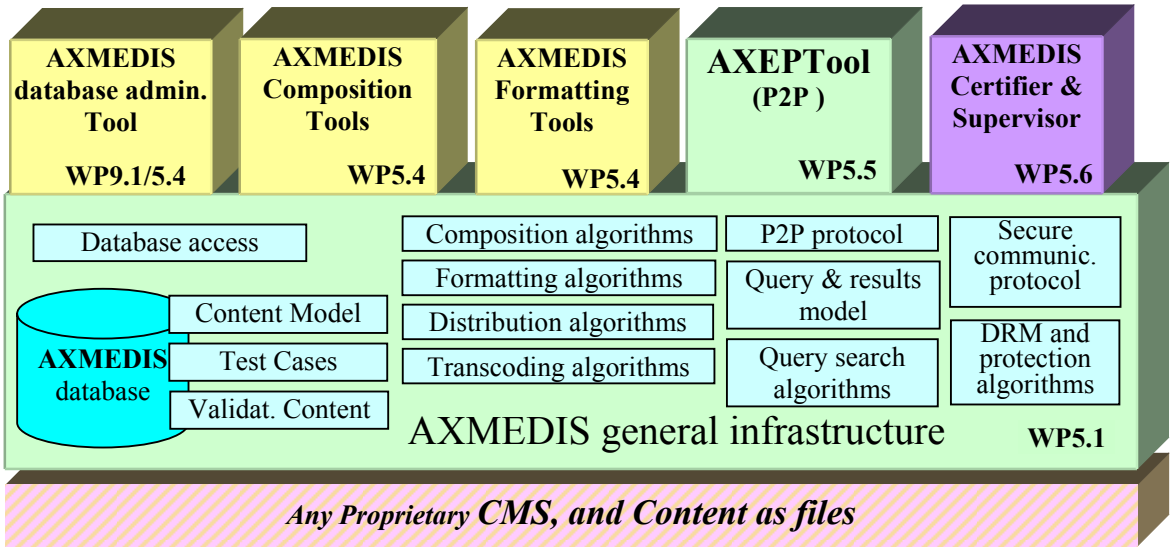
All these tools and areas will be deeply analyzed in the following sections.

3 AXMEDIS Framework overview, General Architecture (DSI, all)

The added value of AXMEDIS will be in the research results producing innovative tools for content production (reducing costs and allowing production and formatting on demand) integrated in an open P2P tool (AXEPTool) at which any CMS could be joined for content production and distribution in the B2B environment of AXMEDIS. These aspects integrated with a set of sustainable demonstrators and take up actions will signal the age of real alliance in the content production process respecting the business of SMEs and permitting at them to access at relevant content and thus at producing content at a reasonable costs.

Most of the components mentioned in this WP are already available in the repository of the partners in an initial version and with less functionalities with respect to the real needs and the goals of the research activity planned for the AXMEDIS project. Some of the objectives have very ambitious objectives.
Some of the available components are in Open Source, supported by GPL or LPGL or other licenses, others are in the public domain, others still are proprietary. All the components provided by the partners will be made available to the community according to the rules specified in the Consortium Agreement in which each case is examined individually and carefully considered (the Consortium Agreement reports also a description of pre-existing knowledge and tools and the rules for their exploitation). A limited set of possibilities that range from LGPL, GPL, to proprietary modules have been used for labelling the IPR brought by the partners into the project. Some of these components will have to be customised to become compliant to the AXMEDIS framework, others will have to be transformed into multiplatform, only a few of them are to be built from scratch. Their maintenance along the project life is a mandatory activity to maintain their alignment to the AXMEDIS platform during its development. Partners that will provide proprietary modules have the duty to ensure they are AXMEDIS compliant.

The AXMEDIS Framework
In the following figure, the structure of the AXMEDIS Framework is reported. It contains all the necessary tools to manage the content workflow from the content production to the distribution over different channels, they are listed in the following.



The general infrastructure gives a common ground on the base of which other AXMEDIS-compliant applications can be built. The most relevant parts of the AXMEDIS **general infrastructure** are:

- **a common model for content** representation which allows that the information about the content can be exchanged between the different tools for the manipulation/fruition, a suitable solution could be the MPEG-21 format (see WP4.1).
- **Repository Administrator Tool** (including import/export, ingestion tools, from proprietary CMS solutions. This Module will be developed in WP9.1 for all the CMS of the partners involved in the project: SEJER, ILABS, WAN, XAURA, OD2, etc.).
- **access methods to the local AXMEDIS database** which stores the content with its metadata and it could be used for searches, classification, etc. as used in the AXEPTool and in any other content processing. All tools will use these access methods to get the object, write back the modification, create compound objects, etc. (see WP4.1)
- **AXEPTool and its P2P engine** for developing P2P applications allowing the content sharing, see WP5.5. P2P communication model, used by AXEPTool and available for other AXMEDIS compliant tools in the future.
- **Composition Tools** and algorithms that perform the composition of AXMEDIS objects, developed in WP5.4 as general tools and in WP9.1 with some more specific tools. Including AXMEDIS rules editor.
- **Formatting Tools** and algorithms that manage the presentation description of the AXMEDIS content, developed in WP5.4 as general tools and in WP9.1 with some more specific tools. Including AXMEDIS style editor for formatting.
- **all the transcoding and digital item adaptation algorithms** developed in WP4.3 and the other algorithms of the same type that will be developed in the demonstrators of WP9. Including multilingual management for metadata and for text, synchronisation tools and algorithms, etc.
- A protection model including:
    - **AXMEDIS Certifier & Supervisor** which controls the DRM and supervise the traffic on the AXEPTool.
    - DRM/Protection solutions, DRM engines, guidelines for licensing and contract definition, protection tools, monitoring tools, fingerprint estimation, enforcing and readers as developed in WP4.
- **test cases and validation content** for validating the identified algorithms and software components and tools included in the framework and for accepting new algorithms in WP5.2.
- **a format for query & results** to represent the request coming from the other peers about the shared content and the results that list the content corresponding the request criteria (see WP4.2). This has to be specified for simple client queries and for complex technical queries.
- **all the distribution algorithms** developed in WP4.6, WP4.7 and WP4.8. They are models for distributing content via I-TV, mobiles, PDA, PC, kiosk, etc.

The main idea of the AXMEDIS Framework

- Production of the AXMEDIS framework as a structured and organised set of software components in the area of cross media production, content sharing, content distribution towards multi-channel. These tools can be mainly used for the production process enabling content production on demand and automatic and semiautomatic production and formatting.
- Production of a general set of AXMEDIS production tools on the basis of the AXMEDIS framework.
- Improvement and maintenance of the AXMEDIS framework and tools.
- Preparation of guidelines to allow the development of the above mentioned points.

The main activities have been divided in the following subWPs:

- **WP5.1 -- Development of the general infrastructure**

- **WP5.2 -- Component Validation and Acceptance, see for this DE5.2.1**
- **WP5.3 -- AXMEDIS framework integration and maintenance, see for this DE5.2.1**
- **WP5.4 -- AXMEDIS Content production Tools**
- **WP5.6 -- AXMEDIS Certifier and Supervisor and DRM tools**
-

3.1.1.1 WP5.1 -- Development of the general infrastructure

This WP is coordinated by DSI
Period: M6-M30
The work in this WP includes several aspects and is quite distributed among all partners with a special attention to industrial partners such as COMVERSE, HP, ILABS, TISCALI that are interested at the exploitation of the produced results from the project.

T5.1.1: Production and maintenance of AXMEDIS guidelines
Managed by DSI with the work of all partners.
The algorithms developed in WP4 will have to provide a suitable interface to be joined each other to build the AXMEDIS framework designed and developed in this WP5. To this end, each software components including specific algorithms for content manipulation will have to be compliant to a specific interface. For example, the identified typical interfaces are those of the algorithms for (i) content transcoding, (ii) content composition, (iii) estimating a fingerprint, (iv) content formatting, (v) reading watermark, etc. These interfaces will be defined in this SubWP at which all the WP responsible of WP4 are called to work. Most of the algorithms have to share a common format and in some cases they have to be capable to communicate according to a unique data model. This data model will be based on the Object Oriented paradigm. The partners have a large experience in this sense. In order to reach these objectives, a set of guidelines will be produced according to the AXMEDIS specification to help the developers to work in precise manner. This will guarantee the interoperability of the research results obtained. This will be quite similar to what has been done in the past for other projects.

T5.1.2: Production of Stub Components of the general framework
Managed by EPFL. The work should be assigned on the basis of competencies to all the partners involved on this WP.
Based on the guidelines the second step will lead to the creation of the stubs of the major components and algorithms. Some of these stubs may become real modules (with common interfaces) in a short time by using the components provided by the partners and reported in the above table (i.e. transcoding algorithms). The creation of the stubs will help to avoid integration problems of the various components/algorithms in the framework in the later developments.
This process will start in the early months and continue through the project improving the interfaces of the components and their number in the AXMEDIS framework.

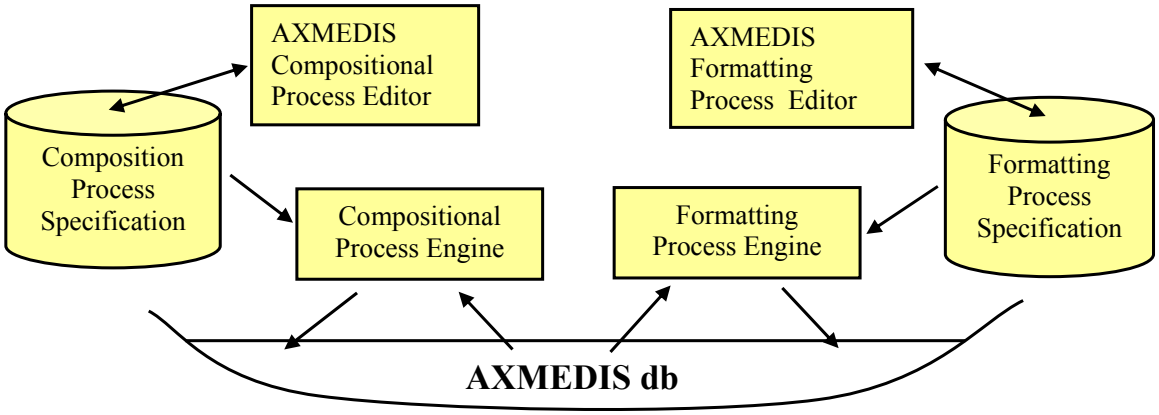
Intended activities and targets of WP5.1 include:
- M10: first version of the AXMEDIS framework guidelines for the production of AXMEDIS components and tools, content for test cases, validation processes, etc.
- M11: Realisation of the first version of the general infrastructure.
- M16, M20, M24, M28, M30: refined version of guidelines and infrastructure.

3.1.1.2 WP5.4 -- AXMEDIS Content production Tools

Managed by DSI.
Period: M9-M48
In this WP the tools that concern the content production will be developed. They are the Composition Tool that performs the composition of AXMEDIS objects, the Formatting Tool that manage the presentation

description of the AXMEDIS content and the Repository Administrator tool which manage the objects in the AXMEDIS database and retrieves content from internal archives (file system, existing CMS). They integrate transcoding and text translation tools.

All of these tools are built upon the general infrastructure described in WP5.1 using algorithms, content model, communication protocol etc. In these WPs, the focus is on the development of a nice user friendly GUI (following the guidelines of the WP4.9) according to their different functionalities. The GUI of both Composition and Formatting Tools will include a rule/language editor to specify more complex composition/formatting constraints and processes using a visual representation of rules and process steps, relations, etc. the produced procedures are interpreted by a specific engine which invokes the framework algorithms as depicted in the following figure.



Each content production tool can handle both protected and/or unprotected objects, by automatically obtaining the key to perform the available operations on the manipulated objects. This is performed by using the DRM module of the AXMEDIS framework, while the authorisation at opening performing the activities is granted from the AXMEDIS certifier and supervisor if the DRM rules enforced in the object allow the requested operation.

T5.4.1: Composition Tools
Managed by DSI.
These tools have to automate or accelerate the content production by composition of other contents components and/or files. Each composition tools may have a set of compositional algorithms as those developed in WP4. The compositional tool developed in this WP will be the general tools for testing and providing these functionalities. The content producers, aggregators and distributors will have the possibility of using related algorithms for developing compositional tools working ion their databases as described into WP9.2. The following features will be included into the Composition tool:

- composition is based on rules on technical details and metadata associate to the objects and object components.
- Production of the AXMEDIS rules editor and engine.
- Grouping objects into a complex one, giving semantic relation between them. These object can be physically stored in the AXMEDIS local repository or could be only the semantic description (metadata) of object owned by other peers actually not completely downloaded.
- Performing a composition of entities which are not objects but query proposal which will locate the real AXMEDIS objects in the P2P at different time (typically preformed by the Content distributor while formatting the compound object).
- Set up of template compounded objects with predefined semantic relation between the simple objects included based on metadata. Using this template a set of compound objects (one for each opera title) can be easily created and eventually browsed and customized.

- Select the proper composition algorithms provided by the AXMEDIS infrastructure.
- Help the user in the composition by showing a composition wizard to manage step by step each property of the compound object.
- Modify DRM and metadata properties for single or group of objects. This is needed when a content integrator use protected object components to create a new content object with added value with other DRM rules. These rules include those for distributors and those per eventual direct end user usage. The Distributor may use the content component for creating more complete and sophisticated content. The Rule that one would like to impose to the final user do not have to violate the initial rules. Thus, a verification with formal models is needed. In addition, the distributor may add new rules casting the previous rules, etc.
- Import from the file system any file containing digital content in any format.

In order to develop a demonstrable AXMEDIS content production workflow, partners will integrate their existing production technologies and where possible, build AXMEDIS-compliant plug-ins or interfaces to link into the current market leading content production tools, such as Quark Xpress, Adobe Photoshop, Avid, Apple Final Cut Pro, Macromedia Flash and Dreamweaver. By interfacing with existing production tools, the project can:

- avoid "reinventing the wheel"
- enable AXMEDIS development to focus on the leading edge research aspects of the new workflow such as P2P, DRM, true cross-media, etc.
- reduce learning curves for designers, producers and editors.
- increase potential exploitation partnerships with the industry key players such as Adobe and Macromedia, who AXMEDIS industrial partners have existing commercial relationships with.

The interfacing and development of the AXMEDIS production tools will include the close involvement of the content creation partners. XIM will advise on its multimedia workflow requirements, and provide constant feedback and validation of the development to ensure that the AXMEDIS tools will form a seamless and streamlined new workflow.

T5.4.2: Formatting Tools

Managed by DSI with the support of XIM, ILABS, HP, EPFL.

This tool has to automate the formatting process before effective distribution over the channels, thus this tool will allow the high-level description of the content presentation and interaction with the final user using a rule based language as described into WP4.3. The formatting tool uses a set of algorithms developed in WP4.3, based on optimisation and specific for each distribution channel. The compositional tool developed in this WP will be the general tool for testing and providing these functionalities. The content distributors will have the possibility of using related algorithms for developing formatting tools working ion their databases as described into WP9.2. The Formatting tool provides a nice, user-friendly user interface as recommended by the usability WP, to:

- express the presentation constraints using both text-based and language including user event reaction, this will be produced from WP4.
- realise the AXMEDIS style editor and engine for formatting rules.
- enable a spatial-temporal view of the described constraints.
- specify a format layout that could be reused or associated with particular compound objects or templates generated by the Composition Tool.
- show a simple layout preview to instantly validate the expressed constraint (eventually differentiated by channel, device profile).
- select the proper formatting algorithms provided by the AXMEDIS infrastructure
- help the user in the formatting task by showing a layout wizard to manage step by step each property of the desired view.

- take into account user profile and formatting style.
- transcode files form one format to another. The transcoder can be plugged into the tool with a dynamic interface.
- modify DRM and metadata properties for single or group of objects, according to the right imposed if the objects are protected.

T5.4.3: AXMEDIS database, administrator tools and support
Managed by DSI
This tool has to mange the AXMEDIS local repository, allowing:
- Browsing/Filtering of the repository according to the object relations.
- Integration of the query engine and visualise developed in WP4.2 by EXITECH.
- Navigating the AXMEDIS content object, nesting levels, etc. activating plug-in viewers, etc.

T5.4.4: AXMEDIS Editor and Viewer, and verification on AXMEDIS terminals
Managed by DSI and EPFL
The development of this tool is performed in collaboration with WP4.1. This tool will be the authoring tool for creating AXMEDIS objects by means of a user interface and/or by using scripting. Practically, it will act as the user interface for testing the main functionalities of the AXMEDIS model. A subset of this tool is also to be used for AXMEDIS clients when these are used on PC platform. This task includes the final implementation of the client tools for AXMEDIS compliant content. Among these, a general viewer will be developed. In addition, this task also includes the design, the integration and implementation of a AXMEDIS viewer based on Mozilla development environment, by SEJER.

T5.4.5: Programme and Publication engine for enabling the on demand
Managed by UNIVLEEDS
This tool is partially developed by exploiting the work performed for the Publication tool in WP4.4. This will allow the reception of specific commands (requests) for creating content and this content will be produced by exploiting the capabilities of the AXMEDIS formatting engine. In addition, the tool will also have capabilities for producing the programme on the basis of specific rules.

T5.4.6: Content Workflow integration
Managed by XIM
The objective of this task is to integrate the AXMEDIS framework tools (AXMEDIS tools for content production and AXEPTool) with one or more workflow management environments. Among those that will be selected at least one of them should be open source. The workflow management environments that will have to be experimented and integrated-in/(customized for) AXMEDIS have to be very powerful and well-known in the market. The integration of workflow capabilities in AXMEDIS has to be based on the preliminary work performed in the T4.3.3, and can be realised with the implementation of some interfaces into the AXMEDIS tools or/and Workflow tools, exploiting the capabilities of AXMEDIS into the workflow and vice versa.
The workflow support regarding the AXEPTool is analysed and developed in WP4.4 by CRS4. In both cases a uniform interface for workflow control and integration will be realised. In this way, different content providers, aggregators and distributors may define and follow the content evolution and flow in an uniform manner considering both the internal work and activities demanded to third party companies on the AXEPTool.
The interface will be developed to ensure maximum compatibility with the requirements of various parts of the 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.

Intended activities and targets of WP5.4 include:

- M10: Analysis and requirements specification for the workflow support for AXMEDIS content production.
- M12: first version of AXMEDIS database administration tool and support
- M12: Identification of suitable workflow engines and selection of the most suitable open source workflow environment to support AXMEDIS content production (the same Workflow engine as provided under T4.3.3 may well prove also best suited here).
- M12: first version of AXMEDIS authoring tools and early version of terminals integrated. AXMEDIS viewers including a first version of one based on Mozilla development environment managed by SEJER.
- M13: first version of integrated Composition tool engine.
- M13: first version of integrated formatting tool engine.
- M14: Analysis and requirements specification of the interface to the workflow support for AXMEDIS content production.
- M16: First version of the Programme and Publication engine for enabling the on demand for distributor channels.
- M17: First version of the workflow support interface for the AXMEDIS content production framework.
- M18, M24, M30, M36, M42, M48, other intermediate and final versions of the same tools
- M25: Final revised specification of the workflow support interface for the AXMEDIS content production framework.
- M29: Updated final specification of the workflow support interface for the AXMEDIS content production framework.
- M36: Updated final version of workflow support interface for the AXMEDIS content production framework.

3.1.1.3 WP5.5 -- AXMEDIS P2P Cooperative Content Sharing

This WP is coordinated by CRS4
Period: M9-M48
The result of this WP will be a complete peer-to-peer tool, namely the AXEPTool, capable to produce, share and distribute content components and complete objects. It is developed upon the message-based communication framework prototyped in WP4.4 and integrated in WP5.1.

The work performed in this WP will be the development the AXEPTool, which require the development of a nice, user friendly interface based on the guidelines produced by WP4.9 on usability, such interface has to provide the following functionalities:

- A login form to authenticate the user identity to the Certifier and supervisor.
- A query builder, that allows the selection of all the technical properties of the shared content on the P2P network, also looking for content into compound objects.
- A result list collecting all the search results, specifying peer source, technical details (bandwidth, object size, category, rating), DRM properties (owner, allowed operations and cost), with the possibility of a complete ordering for all listed fields.
- an intuitive download/purchase function which allow to select directly from the result list one or more AXMEDIS objects.
- a preview option allowing a brief preview of the object without the complete downloading where available.
- a continuous report of the P2P traffic (current downloads and uploads).

T5.5.1: Final Integration and Development of AXEPTool
Managed by CRS4.
The AXEPTool must provide these additional functionalities:

- the downloaded objects are stored in the AXMEDIS local repository to be managed and accessed from the other tools of the framework (composition/formatting tools, DRM).
- a common interface to the proprietary CMS is used to redirect to the CMS the query coming from the

other peers and the results sent back and the download requests to the on-demand export which take the desired objects from the proprietary CMS to the local AXMEDIS repository to be downloaded from the peers (see also WP 9.1).
- a specific technical user interface for creating complex queries including technical, licensing and metadata aspects.

T5.5.2: Loading Tool Engine
Managed by CRS4.
Final implementation phase of work started in the WP4.4. The Loading Tool Engine will be capable of automatically defining rules (selections) for loading content from the AXEPTool framework and make it available for the distributors according to specific activation of Formatting.

T5.5.3: Publication Tool Engine
Managed by CRS4.
Final implementation phase of work started in the WP4.4. The Publication Tool Engine will permit the automatic publication of AXMEDIS content into the Output database of the AXEPTool according to specific rules and selections. The Publication Tool may automatically invoke the protection tool to protect non protected content coming from the AXMEDIS database.

T5.5.4: AXEPTool and protection aspects
Managed by FUPF
Integration of the P2P service with the AXMEDIS certifier and supervisor to authenticate the software and the user identity, enabling to handle both protected and/or unprotected objects, in any the content has to be certified AXMEDIS content.
- Verification of fingerprint contained in each object passing on the AXEPTool in order to: (i) Trace its movements for workflow purpose, (ii) Verifying and avoiding the usage on non-certified content in the P2P network. XIM will provide constant feedback and validation of the development to ensure that the AXMEDIS tools will form a seamless and streamlined new workflow.
- A tool for tracking the evolution of each content object reporting statistics amount the content usage to content providers, etc. This will allow the support for clearance of rights information according to the DRM rules enforced into the objects.

T5.5.5: Workflow management in AXEPTool
Managed by CRS4.
The main idea is provide a support of creating on the AXEPTool a support for realising workflows. This can be done by integrating an open source tool for workflow management into the AXEPTool and providing some integration for: defining a path of the content flow, time schedule, alarms when action are performed or not, alarms when a given content is available, responsibilities, level of management, groups of work, time out to perform actions, etc.
- General workflow management in the P2P tools among providers and distributors. In the same production chain, editorial companies can be present that produce final or semi-final products for other distributors or publishers. Using content management tools also opens the possibilities of wider content syndication, sharing content across multiple concurrent channels. In this confusion, it is very important to keep trace of all the evolutions, ownership and usage that has been done by each single content component. The DRM is attached at the single content level. The added value and content has its DRM, and the percentage division and distribution is directly performed on this basis. It a sort of formalization of the production process.
- Automated workflow support in the form of change control management. As the origin and added value

chain of each content element will be identifiable, it will be possible to automatically update - via the content management system - final multi-channel content as a specific content source is updated somewhere in the peer-to-peer network. For example, a freelance, on-location news reporter may update her news video report, which would be available across the peer-to-peer network, this could trigger an automatic localisation by another company's translation/subtitling tool and then automatically be fed into rich media content streams for simultaneous viewing on a commercial news website, an interactive TV channel and a 3G wireless news portal. All rights would be managed in this workflow to ensure correct distribution of revenues according to contracts.

Intended activities and targets of WP5.5 include:

- M13: First prototype of the integrated AXEPTool with draft query support.
- M15: Loading and publication tools, first integrated version into the AXEPTool.
- M15: Definition of the DRM properties needed inside the AXEPTool.
- M18: refined prototype of the integrated AXEPTool with draft query support and workflow
- M24: Integration of DRM modules, and monitoring aspects.
- M22: first version of the AXEPTool with workflow management support.
- M30: The peer-to-peer architecture will be refined and all project needs will be addressed.
- M36: The final peer-to-peer architecture with all new functionalities required after a real usage of the system.

3.1.1.4 WP5.6 -- AXMEDIS Certifier and Supervisor and DRM tools

This WP is coordinated by FUPF.
Period: M9-M48.

T5.6.1: AXMEDIS Certifier and Supervisor, General Architecture

Managed by EXITECH with the support of DSI.
General architecture of the AXMEDIS Certifier and Supervisor. The AXMEDIS Certifier and Supervisor will be developed with the support of the general infrastructure (DRM/Protection). The AXMEDIS Certifier and Supervisor is NOT a super database of the contents. The content is stored into the databases of the Content Providers and temporary in the databases of the Channel Distributors. AXMEDIS Certifier and Supervisor is the AXMEDIS certification authority that provides services for Content Providers and Distributors and verify the correctness of the Clients. If one AXMEDIS object is obtained by any client by copying it, by getting it via P2P applications, etc., it will be possible, via the AXMEDIS portal and via the AXMEDIS Certifier and Supervisor to get the authorization to use the objects. Thus the distribution of rights will be performed according to the Content Provider and Distributors involved and from these to the owners (by using also the collecting society).

- Identification, specification and development/reuse of the WEB service interface and secure communication protocols.
- Identification, structuring of the AXMEDIS Certifier and Supervisor database structure considering also the distribution of services provided with aim of defining a scalable architecture capable of supporting a huge amount of transactions per second. These transactions can be: (i) requests of key, requests of verification, requests of logs, registrations, etc.
- The architecture of AXMEDIS Certifier and Supervisor should to be flexible enough to support both centralised Certification and Supervision and distributed. In the centralised version only one AXMEDIS Certifier and Supervisor is set up for the whole network, in the other case each distributor and P2P network may have a distinct Certifier and Supervisors. They could be hierarchically organised or stand alone, limiting in this case the navigation of content.
- The architecture of the AXMEDIS Certifier and Supervisor should be scalable and the internal services should be well separable to cope with large traffic for the certification and supervision and to allow the decentralisation of some of the services in an easy and reconfigurable manner.

T5.6.2: Registration, Certification, key generation and Check

Managed by FUPF.
The following functionalities have to be taken into account for the AXMEDIS Certifier and Supervisor:

- registration and certification of the clients and of the other tools that manipulate protected and non protected AXMEDIS objects.
- reporting and tracking about monitoring via fingerprinting verification, implementing a service for digital item verification of unique identification based on fingerprint.
- verification of consistency of any transaction and delivering key to clients (final users or Channel Distributors) to properly decrypt the protected objects.
- check that new content creators/providers accomplish the general DRM rules of AXMEDIS framework before they enter it:
  - Block user access if a violation of DRM rules is detected,
  - Block content creators/providers access if they violate DRM rules,
- managing a database of information related to registrations, certification, etc., such as the unique ID of the objects, their key, their DRM rules (if needed), etc.
- managing the following information:
  - Listing possible objects,
  - Listing all client, registering clients,
  - Listing all distributors and higher level, registering distributors;
- supporting content license migration from one device to others devices of the same owner.

T5.6.3: Trace, reporting and static analysis
Managed by FUPF
Trace and report the usage of the content to Content Producer. This will allow them to verify if their Distributors have correctly performed their work. In addition, being a super-parties entity, it guarantees the correctness of the transaction reporting towards Collecting Societies and thus towards authors and right owners. The trace will keep trace of any action performed on the object from those of content aggregators and distributors to those of final users.

- generate any kind of report, accessible via WEB. Statistic analysis of the content usage, very useful for tuning the service and the structure of the ready to use proposed objects.
- compute any statistics about the access, utilisation, distribution etc. of the objects.

T5.6.4: Accounting Managing and Reporting Tool
Managed by FUPF
This tool is derived from the basis research performed in the WP4.5. It allows:

- listing clients of the provider, with the history of their transactions, etc.;
- storing into the AXMEDIS database the transactions, matching who has done the action on what;
- navigating on the licensing information;
- communicating with the AXMEDIS Certifier and Supervisor to get specific information related to the transactions performed on the objects of a given content provider or aggregator.

T5.6.5: Protection Tool engine
Managed by FHGIGD
This tool is derived from the basis research performed in the WP4.5. It allows protecting single as well as set of objects (selections, collection, etc.): updating costs, maintaining trace of established collections and actions to perform again them in automatic when needed, converting AXMEDIS non protected objects in AXMEDIS protected objects, accepting protection commands from other tools, working and managing the information contained in the Accounting Log and public log, interacting with the Accounting Managing and Reporting tool described in the previous task.
It will be capable of requesting the key to the AXMEDIS Certifier and Supervisor notifying at the same time that a new object is published in its protected mode with related administrative information: content type, owner, identification, etc.

Intended activities and targets of WP5.6 include:

- M12: Start-up the Public Key Infrastructure for providing the needed keys and certificates for user access, content protection and any other security aspect needed.
- M12: First description of the AXMEDIS certifier and supervisor module, including:
    - Description of registration and certification of clients and other tools
    - Description of the verification of consistency of transaction
    - Description of trace on report usage (event reporting)
    - Description of checks for new content creators
    - Description of blocking access to users and providers violating DRM rules
    - Description of report generation
    - Description of statistics creation
    - Description of content license migration
- M17: First prototype of the AXMEDIS Certifier and Supervisor, implementing some or all of the following features:
    - Registration and certification of clients and tools
    - Verification of consistency of transaction
    - Trace on report usage
    - Checks for new content creators
    - Blocking access to users and providers
- M24: revised version of the AXMEDIS Certifier and Supervisor
- M30: The tool will be refined and all project needs will be addressed.
    - Report generation
    - Statistics creation
    - Content license migration
- M36: The AXMEDIS Certifier and Supervisor will be ready for supporting the demonstrators.
- M48: The addition of final features such as those related to the statistical analysis of content used and about the workflow and tracking analysis.

4 Comparison of AXMEDIS, DMP and MPEG-21 architectures (DSI)

See Annex AXMEDIS DMP MPEG21 analysis and comparison

5 Implementation Guidelines (DSI)

The AXMEDIS framework is an environment for integrating and validating the new enabling technologies and new knowledge invented with WP4. At the beginning, this task will produce a set of guidelines for creating software components. These software components will be created including research algorithms as described in WP4. The guidelines and most of the components will be publicly delivered to the whole community together with the components developed by partners in the past and listed in the next table.

AXMEDIS is an Open tool since:

- The components of the framework will be available via the AXMEDIS portal in Open Source for the development or for transforming already present architecture in AXMEDIS compliant:
  - channel distributors, content providers,
  - viewers for the different platforms,
  - all the algorithms of the same type will be interchangeable, any innovation in the format, in the process, in the workflow, in the business model, in the DRMs, can be added into the framework without restructuring;
- The structure of the AXMEDIS framework will be formalised in terms of interfaces among its components. These will be mainly algorithms and software modules for managing content: composition, formatting protection, query, etc.;
- No limitation about the number of content providers and distributors and their access to the content and insertion into the AXEPTool network. Only certification will be needed and it will be provided almost automatically;
- Other distribution channels can be added according to the above solutions: direct channels such as those towards PCs or PDAs or, mediate via Channel Distributors such as those towards i-TVs;
- The same channel structure can be duplicated without any problem. You may have more Channel Distributors for i-TV, pay per view, etc. They can be set up for localization of content and services, for language and cultural differences, for providing content towards different technologies for providing different content;
- No limitation about the number of clients;
- No limitation about the number of transactions;

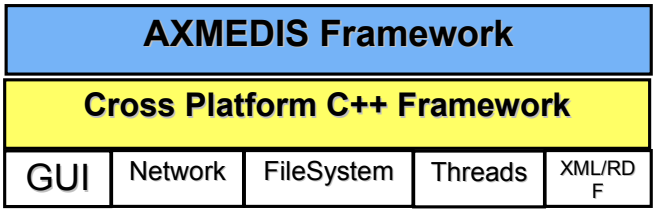
5.1 Decisions on Design, Implementation and Specification Tools (ALL)

- UML: a tool will be selected, VISIO 2003 professional is one of the candidates, other possibilities are UML free tools, etc. They have to be capable of processing UML, verification of consistency and managing packages, action diagrams, IDL, etc….
- Model for API is based on IDL, which in turn is based on UML formalisation
- All the tools including the AXMEDIS Object Manager have to be developed in C++
- AXCS and AXMEDIS object database have to provide for the database access some ODBC or JDBC, this has still to be decided [DSI Chellini-Martini] At the present time, AXCS database accesses are performed by means of JDBC technology
- Technology for AXPMS has not bee fixed yet
- Communications with AXCS can be in SSL
- Database technology has to be scalable: SQL, MYSQL, DB2, Postgress, etc.

- AXEPTool has to be based on non proprietary protocol such as HTTP
    - The same DB used in the AXMEDIS database has to be used in the AXEPTool
- AXMEDIS content processing and editing tools have to be realized in C++, firstly for WINDOWS but taking into account of porting them on MAC/LINUX. So that a GUI abstraction is needed such as Mozilla or WXwin, a scripting tool for the GUI can be taken into consideration..

5.2 Identification of a Framework for Cross Platform C++ Applications Development to be used as base of AXMEDIS Framework

In this section a Framework for Cross Platform C++ Applications Development to be used as base of AXMEDIS Framework is identified.



The work performed is sketched in the following steps:
1. Identification of requirements for the C++ Framework
2. Selection of candidates
3. Evaluation of candidates

5.2.1 Requirements for the C++ Framework

The requirements identified for the C++ Framework to be used as the basis of AXMEDIS Framework are the following.
The **C++ Framework** has to:
1. be multiplatform supporting at least: MS Windows (98?, 2000, XP), MACOS X, Linux
2. NOT be under GPL license
3. NOT be under commercial license (NOT MANDATORY)
4. be rather wide spread
5. be usable from a C++ Application
6. allow the realization of standard user interfaces as well as MDI, drag&drop support, tree view, details view, clip board,…
7. allow run-time generation of user interfaces
8. allow the realization of custom views, where the application can control the visualization and the interaction with the view.
9. allow cross platform access to basic resources like: File system, Network (TCP & UDP), Threads, XML Parsing, RDF.
10. render multimedia information (video, audio, images) (RELEVANT)
11. allow the realization of applications with skin (like Windows Media Player) (OPTIONAL) or not prevent its realization on some platforms (MANDATORY).

5.2.2 WxWidget

For the candidates will be evaluated:
- platforms supported
- licence
- diffusion
- GUI capabilities
    - basic user interfaces

  - advanced user interfaces
  - support for automatic run-time generation of user interfaces
  - custom user interfaces
  - support for multimedia
  - skin support
- Platform abstraction
  - file system
  - network
  - threads
- Tools
  - xml support
  - rdf support

wxWidgets

General	
Platforms	Win32, MAC OSX, Linux, OS2, palmOS, winCE
Licence	LGPL
Diffusion	Good
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible but not directly supported (XRC is used for XML description of GUI)
Multimedia Support	Native support for images (jpg, png, tiff), contributions are present for some multimedia support
Skin support	Feasible but not directly supported
Platform abstraction	
File system	Ok, it supports also zip files
Network	Ok (not UDP)
Thread	Ok
XML	Marginal, a non validating parser is present
RDF	No
Notes	
A library for XML parsing may be used (XERCES)	

In the above evaluation of candidates, are marked in red/orange/yellow the features that are problematic. The features marked in red strongly discourage the adoption of the framework, the ones marker in orange suggest to avoid to use it.
It results that wxWidgets, wxPython, and GTK+ are the frameworks that could be adopted. However it seems that wxWidgets gives more coverage especially since it can be used also in PDAs.

5.3 Supported platforms (All Partners)

The components produced by the research activities will have to be compliant with the languages and platform identified in the specification and in particular:

- Content production tools, editors, publication tools, engines, protection tool editor, etc.:
  - Language: C++
  - OS: Windows XP and 2000, and viable as a second choice also for MACOS X
  - GUI: WxWidget (the new version of the WxWindows)
  - XML Parser: XERCES
  - WebServices/SOAP: gSoap

    - TCP/IP library: that of WxWidget
    - STL can be used but a particular attention has to be given on using only functionalities and data structures that are supported in the STL for PDA (Pocket PC 2003)
    - Avoiding the usage of functionalities that are Microsoft specific
- Protection Manager Support
    - Client: all features as above in C++
    - Domain Home, Domain Factory and Server: Java and C++
- AXEPTool:
    - Publication and loading engines: as the editor area above
    - C++ for the tools
    - P2P virtual database: Java if needed
    - Graphic User Interface: WxWidgets
- AXMEDIS Database and Query Support and AXCS:
    - Database technology: MySQL or Postgres
    - Technology for coding logic: Java
    - DBC: JDBC for the access
    - Operating system: Windows and Linux.
    - User Interface: JSP or PHP, probably better the JSP formalization in classes and graphic design
    - Web Server: Apace, TOMCAT
- Scripting language:
    - Java Script (ECMA Script)
- Workflow Manager:
    - based on OpenFlow or BizTalk
    - Plug in of the workflow: the same as the content production area
- The usage of AXMEDIS Error Manager for all the tools in C++ that use an AXOM
- The usage of AXMEDIS Configuration Manager for all the tools in C++ that use an AXOM
- Fingerprint and Metadata/Descriptors extractors:
    - the same as the content production area

Other information is collected into the AXFW specification document.

5.4 Development frameworks

Development should be done using C/C++ for all code-sharing components, whereas all internal components may be developed using other languages with related tools.

Since the source code of all the tools and components will be accessible to all partners, the choice of a preferred development framework will not limit the possibility to use a different one for regular development and to build for the preferred development framework before upload or on less regular basis.

The following are the preferred development platforms for the project partners under windows:

Partner	Preferred Development Framework
DSI	MS Visual Studio .NET 2003
DIPITA	gcc/CygWin
COMVERSE	MS Visual Studio .NET 2003, MS Visual Studio 6 (Service Pack 5)
EPFL	MS Visual Studio .NET 2003
EUTELSAT	gcc/CygWin
FHGIGD	MS Visual Studio .NET 2003 (MS VS 6.0 and gcc+eclipse as secondary frameworks)
ILABS	MS Visual Studio .NET 2003
HP	
TISCALI	

FUPF	MS Visual Studio 6.0 or gcc (linux tools)
XIM	gcc or MS Visual Studio .NET 2003
CRS4	gcc
SEJER	gcc/CygWin
UNIVLEEDS	MS Visual Studio .NET 2003
IRC	MS Visual Studio .NET 2003
EXITECH	MS Visual Studio .NET 2003 or gcc

Thus the most preferred development framework is MS Visual Studio .NET 2003.

For Java development the choice of a uniform virtual machine will allow to reduce the possibility of integration problems. The chosen JVM is Sun 1.4.2. The migration to a new JVM (1.5.0) will be made all together.
The use of a specific Integrated Development Environment for Java development is not set, but it is mandatory the adoption of ANT for having an easier integration (some IDE use ANT as integrated building system). For testing the use of JUNIT is appreciated. For development of WebServices Java WSDP 1.5 has to be used.

5.5 User Interfaces (All Partners)

For the user interface in C++ applications wxWidgets should be used.

5.6 Communication (All Partners)

For the communications among modules in different processes/machines WebServices is the preferred technology to be used. However in some cases other solutions like XML-RPC or custom protocols could be used, for example for the interaction with existing tools or for performance reason.

For the interaction with modules in the same process static or dynamic libraries should be used. COM/ActiveX technology could be used to interact with existing components.

5.7 Information exchange (EXITECH, All Partners)

This section contains guidelines on the way in which information will be exchanged, through documents posted on the WEB AXMEDIS portal and by synchronizing source code and test cases updates

The main mode for information exchange between the AXMEDIS members is the use of document repository on the WEB PORTAL. Refer to the portal specification for the operative procedures.

The documents editing has to follow the rules defined by the AXMEDIS consortium. The templates to be followed for the document editing have to be posted on the WEB site into the management activity in a dedicated folder. New templates have to be created any time a new type of document has to be produced (i.e. slide template, deliverable template, reports and management reports, UML diagrams, etc.). The document responsible and the activity coordinators have to verify if the received/posted content comply with the template.

In case a new document type is needed, the template can be proposed to the project coordinator, who, if accept it, will post it on the portal.

It is strongly advised against sending documents in attach to the reflector

Also for the code editing a “template” can be created. The programming style template can be based on some standard guidelines as:

- C++ Coding Standard defined at http://www.possibility.com/Cpp/CppCodingStandard.html
- Mozilla Coding Style Guide at http://www.mozilla.org/hacking/mozilla-style-guide.html

- Code Conventions for the JavaTM Programming Language defined by SUN at http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
- Apache coding standards at http://jakarta.apache.org/turbine/common/code-standards.html

The standard guideline can be adapted for the project needs.

6 Guidelines for Code Development (DSI, Bellini)

For Java source code use guidelines http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
For C++ source code:

Source files

- Header files have to have **.h** extension.
- Implementation source files have to have **.cpp** extension.
- All file names have to be lowercase (e.g. *axobject.h*).
- A header file should normally contain the definition of one class, more than one class can be defined in one .h file if they are strongly dependent.
- When including files from directories use forward slash "/"and not back slash "\" (e.g. *#include "wx/dc.h"*)
- Don't use tabs to indent code, use 4 spaces.
- Use 76 columns for coding

Naming conventions

- class names should begin with capital letter (e.g. AxObject)
- class attributes and methods should begin with lower case (e.g. model, getModel())
- local variables and functions should begin with lower case;
- macro, enum values and constants should be in all capital letters with "_" as word separator (e.g. READ_ONLY)
- namespaces should be in all lower case letters
- typedefs names should end with Type suffix (e.g. `typedef ParamValueType<int> ParamIntType`)
- for accessors use get/set prefixes (e.g. getModel(), setValue(x))
- for classes belonging to the AXMEDIS Framework that are exposed to the framework user use the "Ax" prefix (e.g. AxObject), for classes that are not exposed this is not mandatory.

Code formatting

- for brace placement follow one of the following rule, but use the same for the whole module
  - Place brace under and inline with keywords:

```
if (condition)          while (condition)
{                       {
   ...                     ...
}                       }
```

  - Traditional Unix policy of placing the initial brace on the same line as the keyword and the trailing brace inline on its own line with the keyword:

```
if (condition) {        while (condition) {
   ...                     ...
}                       }
```

- in case a line is too long split it leaving the operator at the end of the previous line and align the starting of the continuing line with the expression start, examples:

```
if (aaaaaaaaaa &&
    bbbbbbbbbb &&
    cccccccccc &&
    dddddddddd &&
    eeeeeeeeee)
{
    ...
}

aFunction(aaaaaaaaa,
```

```
                    bbbbbbbbb,
                    ccccccccc,
                    ddddddddd);
```

Class Header Template

```
/**
 * A one line description of the class.
 *
 * #include "XX.h" <BR>
 * -llib
 *
 * A longer description.
 *
 * @see something
 */

#ifndef XX_h
#define XX_h

// SYSTEM INCLUDES
//

// PROJECT INCLUDES
//

// LOCAL INCLUDES
//

// FORWARD REFERENCES
//


class XX
{
public:
    // LIFECYCLE

    /**
     * Default constructor.
     */
    XX(void);

    /**
     * Copy constructor.
     *
     * @param from The value to copy to this object.
     */
    XX(const XX& from);

    /**
     * Destructor.
     */
    ~XX(void);

    // OPERATORS

    /**
     * Assignment operator.
```

```
     *
     * @param from The value to assign to this object.
     *
     * @return A reference to this object.
     */
    XX& operator=(XX& from);

    // OPERATIONS
    // ACCESS
    // INQUIRY

protected:
private:
};

// INLINE METHODS
//

// EXTERNAL REFERENCES
//

#endif  // XX_h_
```

Class Implementation Template

```
// SYSTEM INCLUDES
//

// PROJECT INCLUDES
//

// LOCAL INCLUDES
//

#include "XX.h"                                 // class implemented


////////////////////////////// PUBLIC //////////////////////////////////////

//============================== LIFECYCLE ====================================

XX::XX()
{
}// XX

XX::XX(const XX&)
{
}// XX

XX::~XX()
{
}// ~XX


//============================== OPERATORS ====================================

XX&
XX::operator=(XX&);
```

```
{
    return *this;

}// =

//============================== OPERATIONS ====================================
//============================== ACCESS      ====================================
//============================== INQUIRY    ====================================
////////////////////////////////// PROTECTED  //////////////////////////////////

////////////////////////////////// PRIVATE    //////////////////////////////////
```

for what not explicitly stated try to follow the guidelines provided in:
http://www.possibility.com/Cpp/CppCodingStandard.html

6.1 DOC generation

Documentation of source code (C++ and Java) will be integrated in the code using JavaDoc comments style.

6.2 Adoption of Libraries

Libraries used in the project are:

Library and version	Area	Licence
wxWidget – 2.4.2	GUI	LGPL
FL C++ Lib contribution to wxWidgets Lib	GUI	LGPL
STC C++ based on Scintilla editor, contribution to the wxWidgets Lib	GUI	LGPL
wxImagick	Image processing	LGPL
Imagick	Image processing	LGPL
openssl	Protection	LGPL
cryptlib	Protection	GPL and standard commercial license
xerces-C++ - 2.6.0	XML parser	Apache Licence 2.0
gSoap	Web services	LGPL
CURL – 7.12.13	Crawler	BSD
easysoap	Crawler	LGPL
expat	Crawler	LGPL
libxml2	Crawler	LGPL
IODBC	Crawler	LGPL
PEAR Library	Database	LGPL
Xerces	Database	Apache Licence 2.0
Xalan	Database	Apache Licence 1.1
XPath	Database	
OpenFlow – 1.1	Workflow	GPL 2.0
Zope – 2.7.3	Workflow	ZPL 2.0
Phyton -2.3	Worflow	
Xmlrpclib	Workflow	
Cexpat	Workflow	
Microsoft ASP	Workflow	Microsoft licence
Microsoft .NET	Workflow	Microsoft licence
DirectX SDK	Players	
splay	Players	LGPL

faac	Players	LGPL
im1_dmif_mp4	Players	ISO
im1_dmif_trif	Players	ISO
im1_dmif_remote	Players	ISO
im1_dmifclientfilter	Players	ISO
DOCFRAC	Fingerprints	LGPL
GNU ghostscript	Fingerprints	GPL
XPDF	Fingerprints	GPL
HTMLDOC	Fingerprints	GPL
WordNet (Emglish, Italian, Spanish, French, German)	Fingerprints	Free for English, proprietary for other languages
TreeTagger	Fingerprints	Free for research. Proprietary for commercial use.
CLAM (0.7)	Fingerprints	GPL
Torch3	Fingerprints	BSD
LibSVM – 2.71	Fingerprints	LGPL
Libsndfile – 1.0.11	Fingerprints	LGPL
BeeCrypt	Fingerprints	LGPL
Botan	Fingerprints	BSD-style
CryptLib (?)	Fingerprints	GPL and standard commercial license
RtAudio – 3.0	Fingerprints	BSD-style open source
PortAudio – 18	Fingerprints	BSD-style open source
Libsndfile – 1.0.11	Fingerprints	LGPL
FFTW – 3.0.1	Fingerprints	GPL and Non-free license (see http://web.mit.edu/tlo/www/)
FFMPEG	Fingerprints	LGPL
FOBS	Fingerprints	LGPL
SpiderMonkey JavaScript Engine ver. 1.5 by Mozilla	Engine	LGPL
SoundTouch	Adaptation	LGPL

6.2.1 Licensing terms (All Partners)

- license for libraries and access to code modality as stated in the CA for each module produced and reused

7 Programme and Publication Tools (UNIVLEEDS)

This is a prototype version of the P&P Editor using wxWidget to design the P&P GUI. Currently, the development involves the loading and displaying of the XML data (.pp file). This simulates the retrieval of the data from the P&P programme repository. In this prototype, main GUI are now usable and under continuing enhancements and revisions. Currently working features and functionalities include:

- creating new P&P programme
- loading P&P programme
- editing P&P programme
- saving P&P programme
- activating a P&P programme

Currently the GUI for programme test (quick, full), programme activation, etc are ready. The actual implementations represent the works in hand. The actual implementations for these functionalities are embedded in the P&P Engine (see P&P Engine section), and involving communication via the Workflow and web services which are to be integrated in a later stage.

In preparation of the demo at the AXMEDIS2005 International Conference, the prototype make use of direct socket connection in order to demonstrate the concept of the overall P&P tools, communicating between the P&P Editor and P&P Engine, and also the requests from the On-Demand. This is to be revised and integrated with the Workflow using Web Services.

Currently, the demo is capable of activating a simplified P&P programme from the P&P Editor over the network, to the P&P Engine with test multimedia objects on file in a specified directory.

7.1 Technical Details for the P&P Editor

reference to the AXFW location of the demonstrator	• https://cvs.axmedis.org/repos/Application/pnpeditor/bin/win32 • https://cvs.axmedis.org/repos/Framework/source/pnpmodel/ • https://cvs.axmedis.org/repos/Framework/source/pnpeditor/
List of libraries used	wxWindows 2.4.2 Xerces 2.6.0)xerces-c_2_6.dll or xerces-c_2_6D.dll)
References to other major components needed	• Workflow • P&P Engine
Problems not solved	• Selection editing
Configuration and execution context	• P&P Engine configuration file (for the demo using the socket and port) • Selection-V1-6.xsd • Schemas Rule_Axmedis_PnP.xsd (this is adapted from the original Rule_Axmedis.xsd) in the bin directories for P&P Editor. This is for the P&P programme validation.
Programming language	Visual C++

7.2 AxPnP Model Library

This class was devised for AXMEDIS P&P metadata. This class provides the functionalities of

- Utilises the functionality of Xerces version 2.6.0
- Capable of parsing an XML file or an XML string into a P&P Class using the event driven SAX2 parser
    - Requires the following steps to
        - Load a programme from file

```
AxPnPRule* prog = new AxPnPRule();
```

        - Load a programme from file

```
AxPnPRule* prog = new AxPnPRule();

if(!prog->load(fname)){
      wxLogMessage("Invalid XML AXPnP Rule file", "Open
failure");
      delete prog;
      return;
}
```

        - Load a programme from a string

```
AxPnPRule* prog = new AxPnPRule();

if(!prog->parseXMLString(theString)){
      wxLogMessage("Invalid XML AXPnP Rule string", "Open
failure");
      delete prog;
      return;
}
```

        - Write a programme to a string

```
wxString xmlString = programme->writeXMLString();
```

        - Saving and save a programme to a new filename

```
programme->saveAs(fname);
programme->save();
```

        - Changing fields (see documentation for a complete list of functionalities including get and set functions for the Header, schedule and distribution rules)

```
programme->getHeader()->setRuleName( "new_rule_name");
programme->getSchedule()->setTime(wxDateTime::Now());
```

- The Parser instantiated to however, this is still a work in progress and further checking is required:
    - Do namespaces checking
    - Do schema checking

7.3 Treeview Class (tree.h and tree.cpp)

- Uses the P&P Programme build the wxWindow tree controller used as a view of the P&P programme and also as a workspace for copying header data, schedule data etc. from one programme to another using drag and drop functionalities
- Sub menus have been developed

7.4 MDI Child Class (pnpchildframe.h and pnpchildframe.cpp)

The P&P programme for editing is segmented into three sections which have been tabbed, the Header information, the Schedule information and the Distribution information. The header information is

tabbed again into general header, producer and comments. The text fileds and drop down combo boxes allow the user to change values to the programme and make selections. As a model frame, after editing the programme, the user can cancel the changes closing the window, update the programme from the window using the update button and update the programme and close the window at the same time using the OK button. This is shown in the following figure (see "Example Programm5" window and "New Programme" window).

7.5 GUI Functionalities

- The XML files is loaded using the openFileDialog functionalities using wxWindows. Selection of an XML file with automatically load and parse the file to a DOM and display in a tree view. The GUI also allows .pp files to be dragged and dropped into the tree workspace to automatically be loaded.
- The usual tree functionalities such as opening and closing tree items are provided by wxWindows

7.6 Screenshot of P&P Editor – version 1.0

The following screenshot shows the loading, parsing, and viewing the data of the xml file within the tree structure and the child editing windows for the loaded P&P programmes. The "Example Programme 5" MDI window in the following figure shows the header information that can be edited and saved. The "New Programme" shows the distribution parameters for setting the distribution and terminal id's and the wxGrid class for adding AXOIDS and selections.

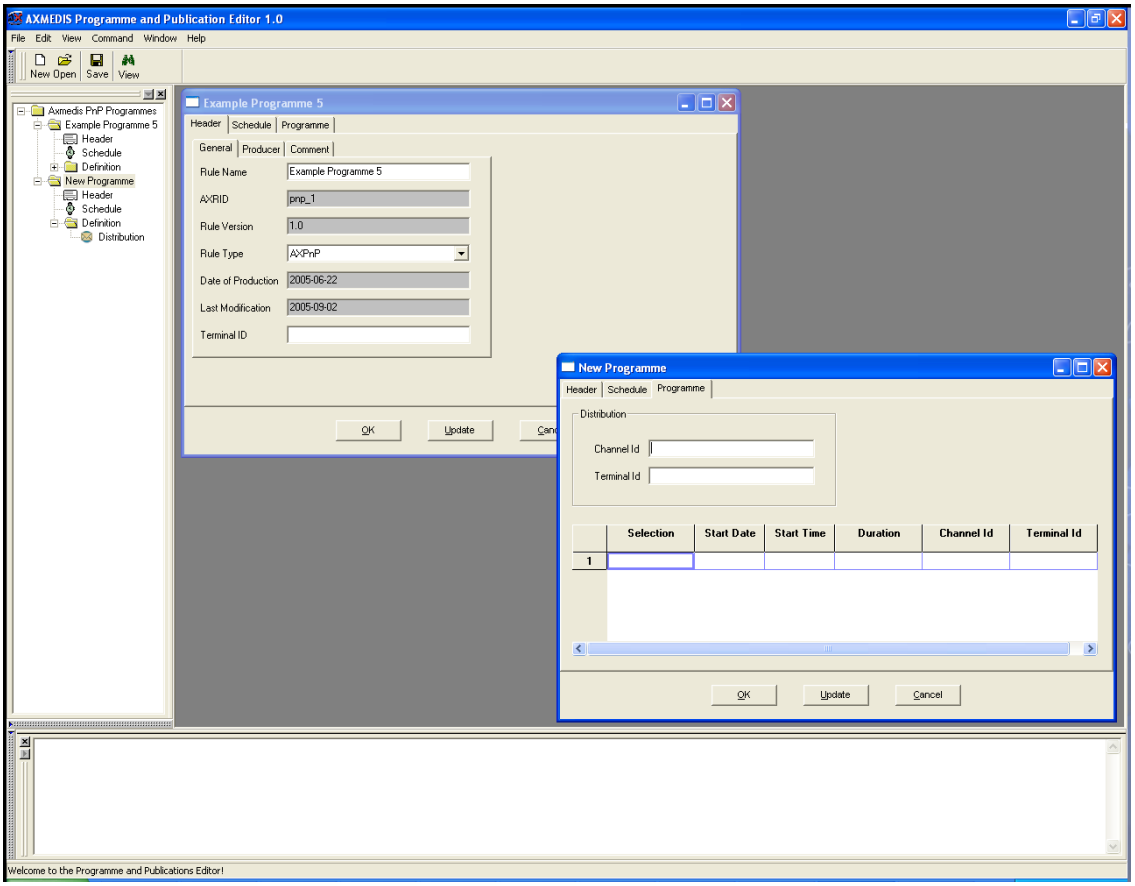


Figure. Screenshot of the current version of the P&P Editor loading the test P&P metadata .pp XML files.

The following figure shows the method and keyboard short cut for activating a programme, the test programme is still under development subject to the functionality of the P&P Engine still in development. Current development connects the P&P Editor to the P&P Engine using wxSockets. Future development is

using GSoap to communicate with the P&P Engine and in future with Workflow. A log message in thebottom window of the main editor displays successful and failed connection to the P&P Engine.

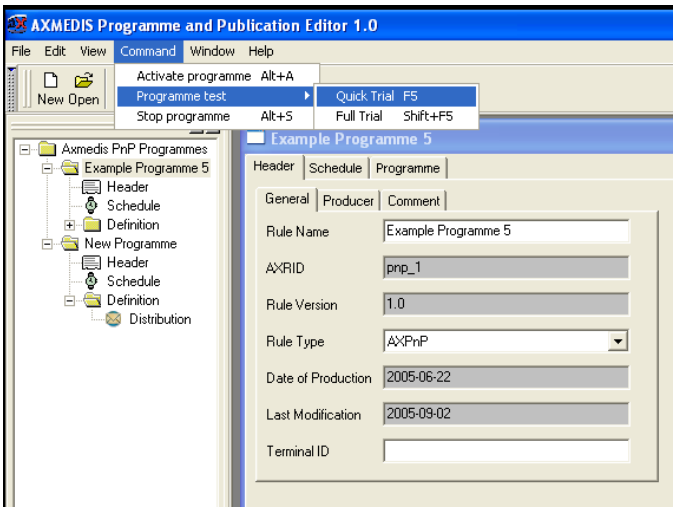


Figure. Testing and activating the P&P programme

Further functionalise include the toolbar for creating new and opening P&P programmes as well as drop down menus in the tree view to use as a workspace including the dragging and dropping of header, schedule and Selection data from one programme to another.

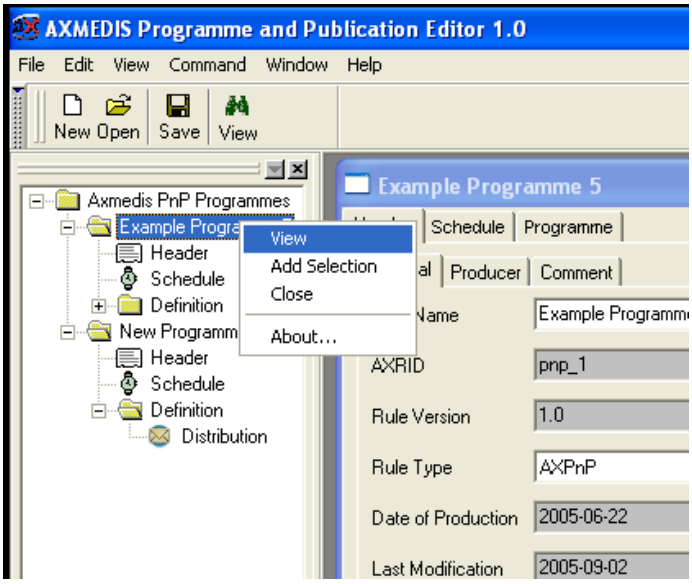


Figure. Dropdown menu selection for the tree view as a workspace

7.7 P&P Engine

P&P Engine is design to run continuously in order to manage the processing and delivery of activated programme in several different levels including quick and full trails, real activation and cancellation of programme. The P&P Engine also include functionalities to mange On-Demand request (T5.4.5).

In essence, the P&P Engine waits for processing requests from either P&P editor or On-Demand. When a request is received, it determines the nature of the request which may be:

- trials from the P&P Editor (quick, full)
- actual programme from the P&P Editor
- On-Demand request

Depending on the nature of the requests, it parse the programme and manage the formatting (if required) and distribute (publication) the requested media at the specified time, take into account the distribution server profile to compensate for the delivery time estimated.

On-Demand request will of course be handled and distributed immediately. For a full programme, a thread will be started to manage the programme. The process parses the P&P programme and checks for any formatting requirements and performs them by forwarding the requests to the formatting engine. The Formatting Engine performs the necessary transformation and creates a new object, and return the new AXOID to the P&P Engine. Once these are ready, the thread waits (sleeps) until the estimated delivery time (before the actual consuming time depending on the distribution server profile).

At the specified time, the thread awakes and performs the delivery as specified in the P&P programmes and rules received, using appropriate technologies for the different channels.

Main works to be carried out include:

- implementations for the delivery to the distribution servers
- better monitoring and logging on the processes
- enhancing and testing larger workload with multiple requests and mixing On-Demand with full programme specifications
- etc

7.8 Technical Details for the P&P Engine

reference to the AXFW location of the demonstrator	• No in this version of t he document
List of libraries used	wxWindows 2.4.2 Xerces 2.6.0 (xerces-c_2_6.dll or xerces-c_2_6D.dll)
References to other major components needed	• Workflow • P&P Editor (not needed but interconnected) • On-Demand (not needed but interconnected)
Problems not solved	• Connection to distribution servers
Configuration and execution context	• P&P Engine configuration file (for the demo using the socket and port) • Selection-V1-6.xsd • Schemas Rule_Axmedis_PnP.xsd (this is adapted from the original Rule_Axmedis.xsd) in the bin directories for P&P Editor. This is for the P&P programme validation.
Programming language	Visual C++

8 AXMEDIS Certifier and Supervisor (DSI, Chellini, Martini)

Activity performed over AXMEDIS Certifier and Supervisor has been mainly focused on databases and web services. Most of the database structure has been defined and implemented with MySql in order to be tested with AXCS modules and components. Moreover it has been projected and implemented most of the software layer needed to access database: this is called AXCS Database Interface. This has been developed using Java 5 and MySql Connector/J library to access MySql tables. At last has been also implemented some web services that uses AXCS Database Interface to access AXCS databases. The platform used for the web services is Tomcat/Axis: Tomcat as servlet container and Axis as soap engine.
The implemented web services are:
Registration web service: this web service is the one put in charge to register users in the AXMEDIS system.
Accounting Web Service: this web service is used by CAMART to gather data for accounting purposes.
Statistic Analysis Web Service: this web service is used by CAMART to gather data for statistic analysis
The implemented databases are:
AXCS Registration and Certification Database: it contains AXMEDIS users related data.
AXCS ObjectsID Database: it contains AXMEDIS object related data.
AXCS Accounting Database: it contains object usage related data.
Since snapshots of libraries, web services or databases have no way and no sense to be represented, in the following paragraphs there will be described some technical details about the AXCS components mentioned above. Descriptions contains also architectural diagrams and schemas such as E-R diagrams describing databases.

8.1 Technical Details

reference to the AXFW location of the demonstrator	A path in the CVS: • No in this version of t he document
List of libraries used	jug, mysql-connector-java, axis related java libraries
References to other major components needed	Tomcat, Axis, Mysql
Problems not completely solved	• adaptation of the web services to the new version of the database access layer (axcs-db-interface) and database structure (second prototypes)
Configuration and execution context	library and web services
Programming language	Java 5

8.2 AXCS databases

As stated in the introduction, the first database prototype has been implemented using MySql. Database structure is quite stable now. Main work about definition of required fields, keys (either primary or secondary) and relation between tables has already done: only minor changes (like adding new fields or changing field types) has to be done. The mostly relevant development tools used as support for database design and implementation are;

- ✓ PhpMyAdmin and MySqlFront as database frontend, used in the implementation phase
- ✓ DBDesigner used in the design and analysis phase.

In order to implement relation between tables, in MySql table definitions has been defined referential integrity constraints across tables stated in the same database. Referential integrity constraint between tables stated across different databases has to be defined and verified in the implementation code used to access database (AXCS Database Interface).

Here are reported ER diagrams of AXCS databases within some small description about the databases and their pertinent tables.

8.2.1 AXCS Registration and Certification Database

Here is reported the list of database entities and the related meaning.

GenericUsers: this entity contains a part of data about AXMEDIS users common to all the specific kinds of users (see below) and requested by AXCS to perform its work.

FinalUsers: this entity represents end users of the AXMEDIS objects and is a specialization of the GenericUsers entity. It contains more details on the status and the registration date of an end user.

B2BUsers: this entity contains general data about B2B users like Creators, Distributors, Collecting Society, Tool Producers and so on. It does not contains information on the registration date and the status of a B2BUser because these data are related to the specific kind of user (Creator, Distributor, etc.).

Domains: this entity contains information about AXMEDIS Domains. A Domain can be referred to a FinalUser (if the PMS is for private use at home) or to a B2BUser (if the PMS is located in an organization like a company or a school). It is linked to the parent entity FinalUser or B2BUsers according to the prefix of the AXUID field which identifies if the AXUID is relative to a FinalUser or a to a B2BUser.

Creators: this entity contains specific data about Object Creators, Integrators and Producers. It is linked to the parent entity B2BUsers.

Distributors: this entity contains specific data about Object Distributors. It is linked to the parent entity B2BUsers.

CollectSoc: this entity contains data about Collecting Societies. It is linked to the parent entity B2BUsers.

ToolProducers: this entity contains data about Tool Producers. It is linked to the parent entity B2BUsers.

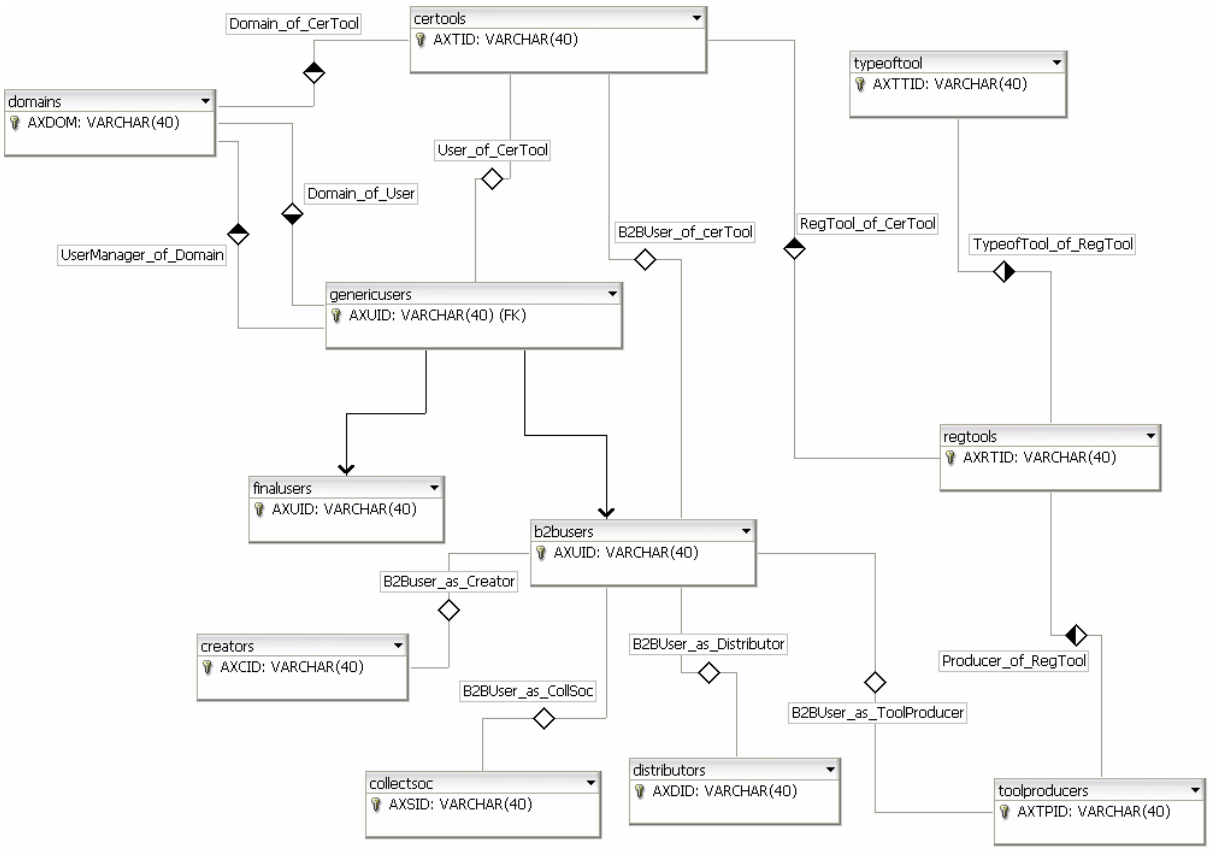
RegTools: this entity contains data about Registered Tools. The “registration” term refers to Tool Off-line Registration scenario. A registered tool is a software product. An instance of a Registered Tool running on a terminal becomes a Certified Tool. A registered tool is identified by an ID called AXRTID.

TypeOfTool: this entity contains all the type of tool that can be used in the AXMEDIS system.

A set of possible tool type is:
Composition Engine
Formatting Engine
Editor PC/MAC
Viewer PC/MAC
Viewer/player: PDA
Viewer/player: mobile
Protection Tool Editor
AXEPTool
Programme & Publication Engine
Publication Tool
Generator from CMS

AXCS
Super AXCS
AXMEDIS OID Generator
AXMEDIS PMS Client
AXMEDIS PMS Home
AXMEDIS PMS Server
PLUGIN xxxxx
PLUGIN yyyyy

CerTools: this entity contains data about Certified Tools. The "certification" term refers to Certification of a Tool/User scenario. A certified tool is an instance of a tool running on a terminal. A certified tool is identified by an ID called AXTID field. It is linked to the parent entity User or B2BUsers according to the value of TypeOdID which identifies if the relative AXID is a AXUID or a B2BUserID.



ER diagram of AXCS Registration and Certification Database

8.2.2 AXCS ObjectsID Database

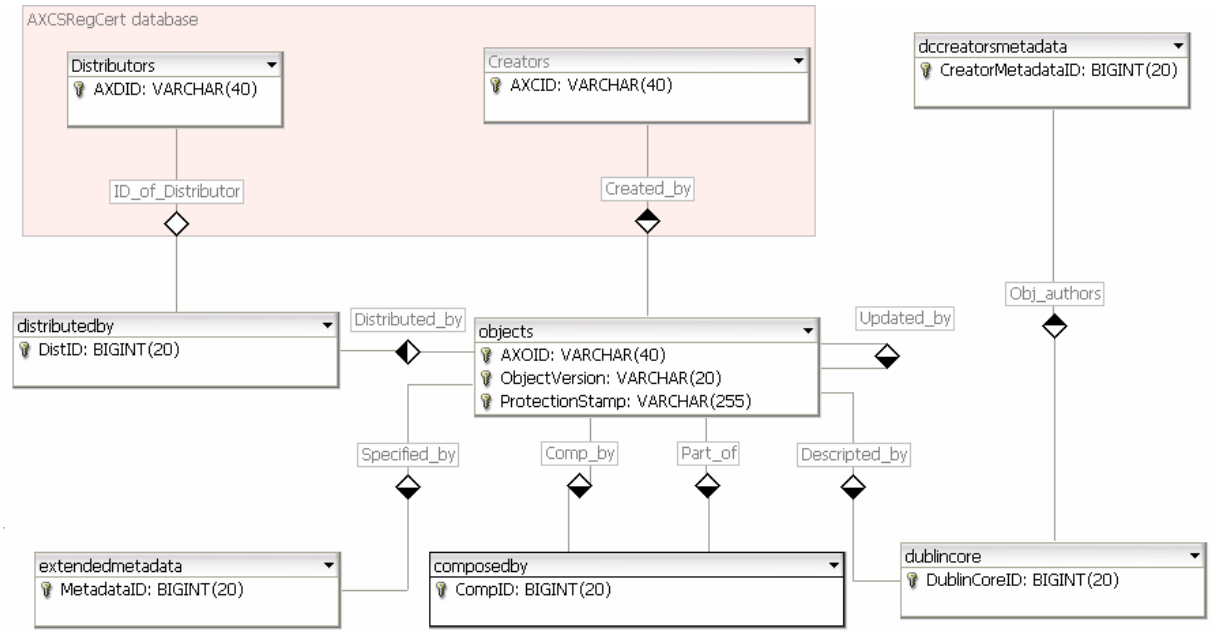
Here is reported the list of database entities and the related meaning.

Objects: this entity contains a part of data about objects. It is linked to Creators entity (located in the AXCS Registration and Certification Database) with an 1-N relation and to the Distributors entity (also located in the AXCS Registration and Certification Database) with an M-N relation implemented as a table called **DistributedBy**. It means that an Object can be distributed by more than one Distributor. It is also linked to itself with an N-M relation implemented as a table called **ComposedBy** to state that an Object can be a Complex Object composed by other Objects.

DublinCore: this entity contains Dublin Core metadata related to an Object. Every raw in the pertinent relational schema table is a Dublin Core set of metadata description about an Object related to a language.

DCCreatorsMetadata: this entity contains Authors pertinent to a specified set of Dublin Core metadata related to a language. It has been introduced to take into account the more than one Author multiplicity. It is linked to the **DublinCore** entity with an 1-N relation.

ExtendedMetadata: this entity contains optional metadata about Object not included in Dublin Core. Every row in the pertinent relational schema table is a single metadata value. This is a way to have a variable number of metadata fields related to every object.



ER diagram of AXCS ObjectsID Database

8.2.3 AXCS Accounting Database

The Accounting database is structured to take care of Action-Logs logic and, once it will be standardized, will have to implement all the necessary parts of MPEG21 Event Reporting. Here is reported the list of database entities and the related meaning.

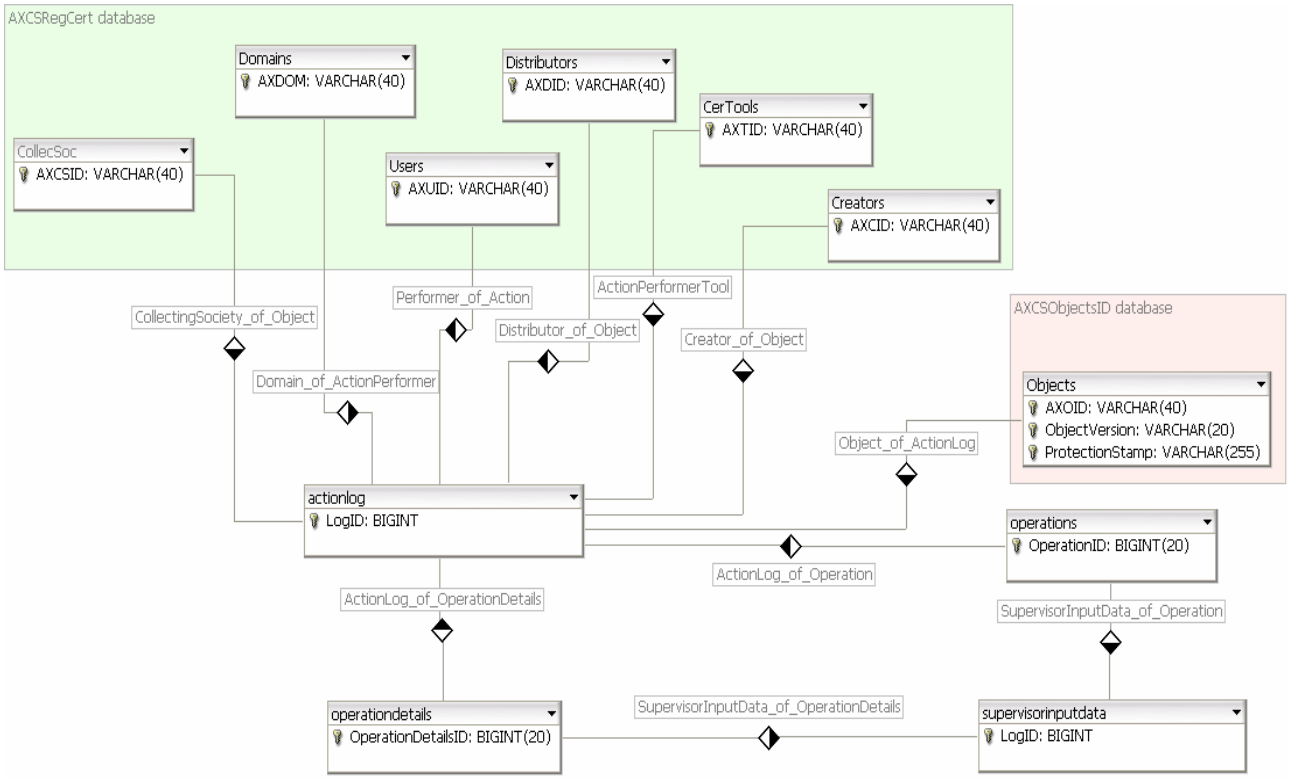
ActionLog: this entity stores the Action-Log as it is with some fixed information such as the AXOID on which the operation is performed, the AXUID of the user that performed the operation, the registration timestamp and execution timestamp (that can be different because of off-line operations performed on the objects). This entity is linked to the **OperationDetails** entity described below.

Operations: this entity will list all the allowed operations for all objects with a description and some other (at the moment not detailed) fields.

These operations are based on the RDD terms of MPEG-21 standard, which have the concept of operations associated to an Event Report (Action Log for us). We have based our proposal of structure from this. We consider only operations made over an AXMEDIS Object. These operations generate an Action Log.
A list of these operations is:
- Modify : To edit an object in order to change it, or to protect it adding rules or metadata.

- Aggregate : To obtain an AXMEDIS Object as a composition of AXMEDIS Objects
- Render : To use or view an Object
- Play : Render as Performance
- Print : Render as Fixation
- Originate : To create a new AXMEDIS Object
- Enlarge : To Add something to an AXMEDIS Object already created
- Reduce : To modify an AXMEDIS Object by taking away from it
- Diminish : To create a new AXMEDIS Object from another one. The Object created is smaller than the source.
- Adapt : To Copy. To edit an AXMEDIS Object creating a new one which has the changes.
- Embed : To put an element or AXMEDIS Object into another AXMEDIS Object
- Delete : To destroy an AXMEDIS Object
- Identify : To nominate an AXMEDIS Object uniquely

OperationDetails: this entity will contain the details of the operations such as channel, duration and other details that are to be specified in deep. It is useful for information purpose.

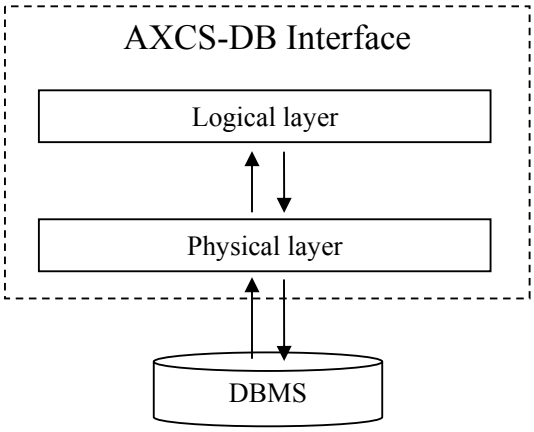


ER diagram of AXCS Accounting Database

8.3 AXCS Database Interface

The AXCS Database Interface (called AXCS-DB Interface) is the abstraction layer that has the task of giving to the rest of AXCS modules a view of the database that is independent from the database that is really employed. This module can be decomposed in other two layers used to realize the abstraction from the database. The first layer (logical layer) is an object model of the database structure: every table is represented as a class and related managing methods have been implemented. The second layer (physical layer) is realized to decouple database model among the specific Database Management System used. In this way,

different DBMS such as Mysql, MS-SQL Server, etc. have their own physical layer, and a change of DBMS does not require a logical layer change.



Each layer (either physical or logical) is composed by a proper interface and a group of classes which realize it.
The logical layer interface defines the minimal set of methods needed to manage data in database tables; each class realizes these methods and an other set of more specific ones related to the pertinent tables.
Since this layer is used to access all the different AXCS databases, it is composed of some specific parts. It can be identified the following modules performing their pertinent tasks:

- Registration: It provides a way to access AXCS Registration and Certification Database; it is implemented in the package *org.axmedis.axcs.db.registration*.
- Objects: It provides a way to access AXCS Objects ID Database; it is implemented in the package *org.axmedis.axcs.db.objects*.
- Accounting: It provides a way to access AXCS Accounting Database; it is implemented in the package *org.axmedis.axcs.db.accounting*.

Each module has the same structure as the others. They are composed by the following four methods defined by the interface used in this layer, and each class realizes a 1 to 1 mapping with the related database table.

Generic class methods (common for each class)

Method	**saveOnDb**
Description	It saves the object on the DB. If the input parameter is set to false no replace is performed but only an insert that can fail, otherwise it performs an update.
Input parameters	boolean replace – it specifies if the operation required is an insert or an update (optional) boolean constarintsCheck – it specifies if the method has to perform the additional referential integrity constraint checks over the related tables and databases. If it is set to *false* these checks are not performed.
Output parameters	int errorCode – It spcifies the operation result: 0 means ok; see the code documentation for other values.
Method	**deleteObject**
Description	This methods eliminates from the current data table of the DB the object having the primary key alredy set .
Input parameters	(optional) boolean constarintsCheck – it specifies if the method has to perform the additional referential integrity constraint checks over the related tables and databases. If it is set to *false* these checks are not performed.
Output parameters	int errorCode – It spcifies the operation result: 0 means ok; see the code documentation for other values.

Method	**loadFromDb**
Description	This method loads the object with the parameters related to PK that is alredy present in the instance of the object. Returns true if the object has been found and loaded correctly, false otherwise. If a false value is returned there is no guarantee the values in the fields have some sense.
Input parameters	
Output parameters	boolean resultStatus – True if the object whose axoid is given as a parameter has been loaded.
Method	**loadFromDbMulti**
Description	This method returns the records with the data related to the constraint specified. Returns a ResultSet if the query has been performed correctly, null otherwise.
Input parameters	string constraint – It specifies the criteria to select records from table (it is the condition after the SQL 'WHERE' clause).
Output parameters	ResultSet result – A ResultSet typed object (eventually empty) if the query could be performed; a *null* value otherwise.

The physical layer interface defines the minimal set of methods needed to take into account the usage of different DBMS. This interface defines methods used to mange the connection, the transactions and the execution of queries at a lower level.

Physical connections to the DBMS used are managed by means of a connection pool manager. It creates a predefined but configurable set of connections (the pool) when it is used for the first time. These connections can be used at request and when they are no more used they are released but not destroyed: they return into the pool to be available again. In this way the overhead due to the creation of connections can be avoided at the time a connection is needed. If the pool of available connections becomes empty it gets incremented by a fixed (but configurable) amount of new connections. The management of the pool is completely transparent and it does not require any operation from the requestors.

8.4 AXCS Web Services

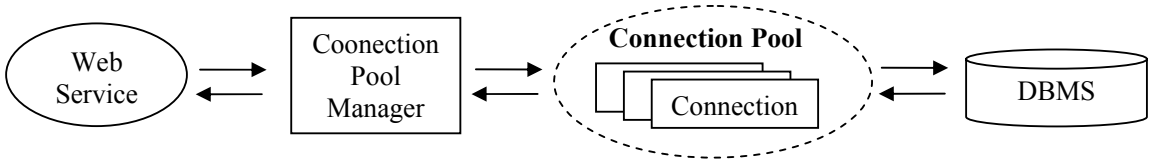
Three Web Services have been realized to expose the main functionalities of AXCS to the others modules (such as CAMART, Distribution Servers, etc.). Two of them are related to the gathering of data needed for specific purposes, the other is used to insert information about new users (every kind they belong to) in the pertinent database.
These three Web Services are shortly described in the following:

- **Registration Web Service** – this Web Service is used by Distribution Servers to add information about new users or to add new data about existing users in the AXCS Registration and Certification database. This Web Service exposes only one method called *Registration*. It requires three parameters to perform its task: two strings specifying credentials of the entity which is requiring the registration operation and an object containing the data to be inserted in the database. The *Registration* method first of all checks the validity of the credentials supplied. If this check is successful, it proceeds with other checks on the data contained in the third parameter to ensure there are not non-sense information (e.g. we are trying to register an user already in the database, or a final user who is qualified as a distributor and so on). If all these checks are passed, the real registration process is started: if the registration is about a new user a unique AXUID is generated and the received data are inserted in the database.
- **Accounting Web Service** – this Web Service is used by CAMART to gather data for accounting purposes. The CAMART can be consider the client part of the reporting system and the AXMEDIS Reporting Web Service can be consider the server part. Distributors, Creators, Integrators, Collecting

Societies have to know all the operations performed over their pertinent objects to receive the correctness fee from the object users. All the accounting needful information are stored in AXCS Accounting Database. This Web Service exposes only one method called *AcceptRequest*, that requires three parameters to perform its task: two strings specifying credentials of the entity which is requiring the registration operation and a third string specifying a *TimeStamp*, i.e. a temporal reference to select interesting data. So the response of the Web Service contains only information related to the requestor and concerning operations performed after the specified timestamp.

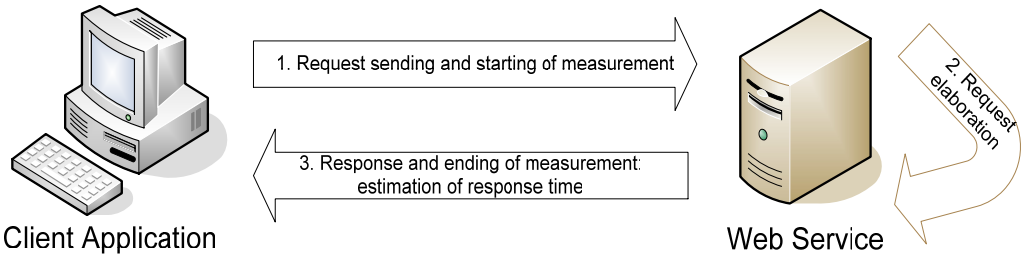
- **Statistic Analysis Web Service** – this Web Service is used by CAMART to gather data for statistic analysis purposes. The CAMART can be consider the client part of the statistic analysis system and the AXMEDIS Statistic Analysis Web Service can be consider the server part. A Distributor, Creator, Integrator could be interested in knowing where, when and how an object is used: this interest could be used for commercial and marketing purpose. Knowing usage, distribution and integration statistics could be an important resource for an AXMEDIS subject to improve his business. All the statistic analysis needful information are stored in AXCS Accounting Database. This Web Service exposes only one method called *AcceptRequest*, that requires three parameters to perform its task: two strings specifying credentials of the entity which is requiring the registration operation and a third string specifying a *TimeStamp*, i.e. a temporal reference to select interesting data. So the response of the Web Service contains the information related to all the objects and concerning operations performed after the specified timestamp.

Concerning the Web services, the target platform used in the development process is the Apache Tomcat 5.5.x as servlet container and Apache AXIS 1.2.x as SOAP engine.
The setting up of a Web Service requires two phases. The first phase consists in writing the code that implements the logic and the methods to be exposed. The second phase consists in the deployment of the realized code to the servlet container to make it available to the requestors. During the deployment phase can be specified the behaviour of the servlet container concerning the deployed code. It can be specified how the application has to manage requests: it can serves every request creating for each a new instance of the server object (in this case the Web Service is said to be "Request" scoped); or the application can create multiple objects, but only one for each requestor serving all of its requests (in this case the Web Service is said to be "Session" scoped); or the application can create an unique object that serves all the requests from all the requestors (in this case the Web Service is said to be "Application" scoped).
In the deployment phase of the described Web Services it has been specified a "scope" parameter with the "Application" value. This has been done in order to minimize the number of connections to the database. Using the "pool of connection" technique (as described in the above paragraph) a lot of fruitless connections were allocated, used once and not set free. Specifying to the "scope" parameter the "Application" value the number of connections to the different databases has been noticeably reduced and the Web Services implementation has been considerably optimized.



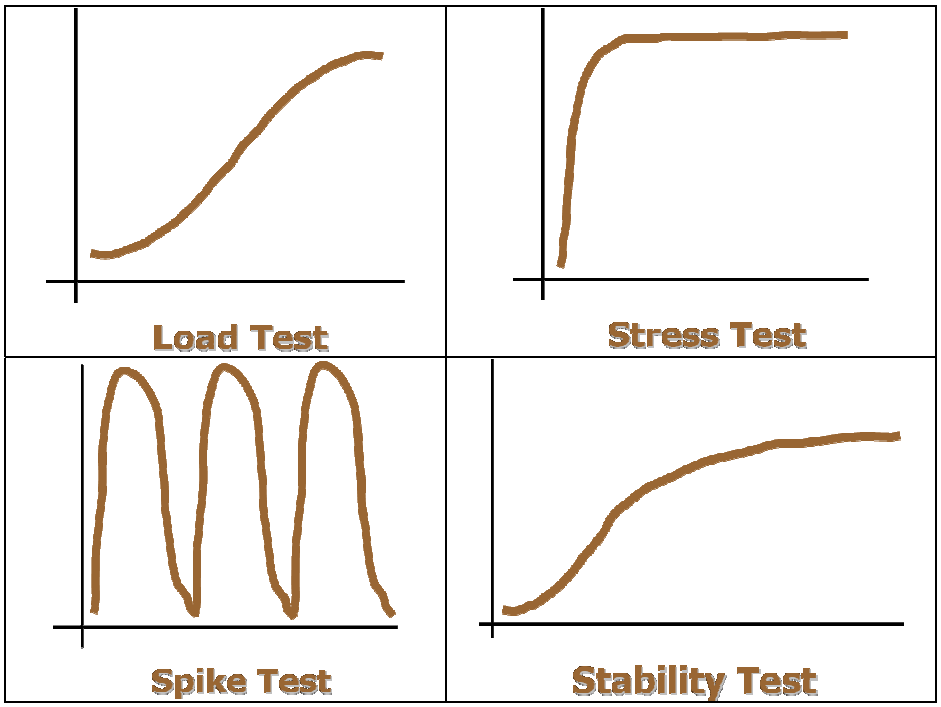
8.4.1 Load tests

These tests have the purpose of getting a performance measure of the Web Services. They are performed with the scope of identifying the trend of the web service response time when the number of concurrency request increases. In this way it can get a sort of scalability measure of the tested web service.
To perform these kind of tests, it is required the creation of a client application which interacts with the Web Service it's written for. This client application (called web service client) has to contain all the instructions needed to deal with the web services and some extra instructions with the aim of collecting a set of data useful to determining the performance measure of the related web service.



There are two main interesting measures: time measure and capacity measure. The first concerns the time needed to complete the required task and is estimated starting the measure when the request is sent and ending when the response is received by the web service client. The second is about the maximum number of simultaneous requests the web service can satisfy without get crashed. All the analysis has to be done taking into account the context and all the external influence, such as connection bandwidth throughput, hardware capabilities, other tasks running on the same machine, and so on.
The most important performance measures are obtained executing the following tests:

- Load Test. The web service is put through a growing number of requests. The number of requests grows until the maximum tolerable number is reached. It is useful to understand the behaviour of the web service put through an overload of requests while operating in its normal environment.
- Stress test. The web service is put through load of requests greater than the standard one. It is useful to understand the behaviour of the web service if the number of request increase highly and unexpectedly, till it reaches a great value (near the maximum tolerable number).
- Spike Test. The web service is put through cyclic peaks of requests. A repeated great number of concurrency requests is useful to understand the web service behaviour when it is put through great load of requests.
- Stability Test. The web service is put through a standard load of requests for a long time. This test is useful to detect unexpected problems such as memory leaks.

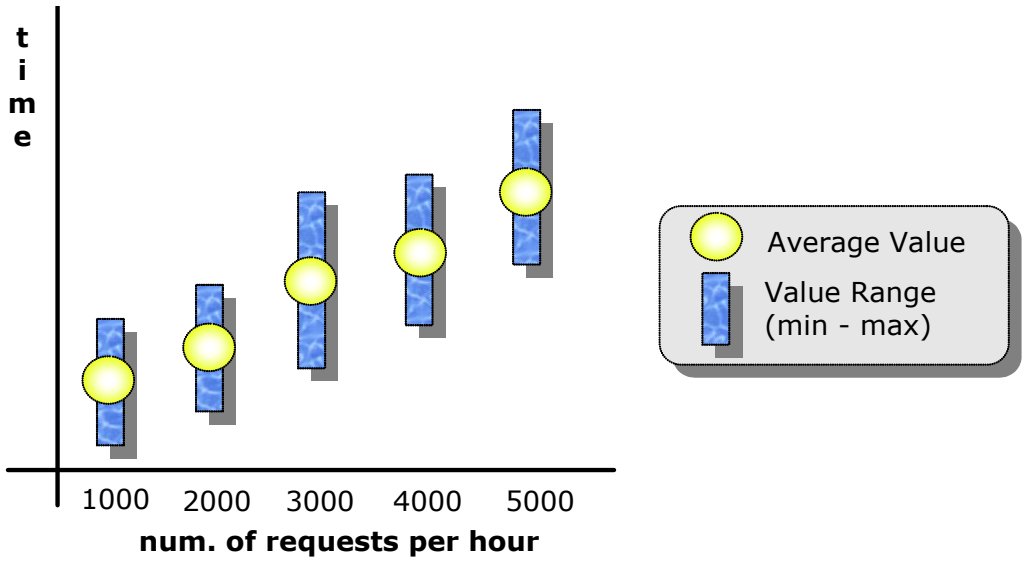


Typical trends of load and stress tests (number of requests per time)

A very common error while performing this kind of tests, is to consider only average values. It is a great error because lower and higher values are not taken into account even if they could be very important. Here is an example. Let us consider a web service which the target response time is 6 seconds and the average response time, measured for 100 clients is about 5 seconds. At a first sight it seems to be a good behaviour, but it is really not. Let us suppose the detected response times are distributed as follows:

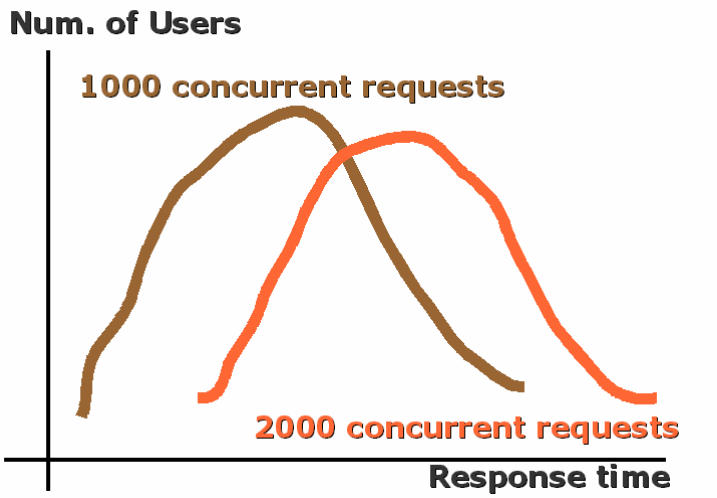
- 40 clients after a time of 3 seconds
- 25 clients after a time of 4 seconds
- 15 clients after a time of 7 seconds
- 20 clients after a time of 9 seconds.

In this scenario there are 40 clients that are fully satisfied, 25 clients that are satisfied, 15 clients that are not satisfied and 20 clients that are terribly unsatisfied. It is a situation quite different from considering only average values. It has to be considered also that an average value of 5 seconds could be produced by distribution times very different from this: 100 clients after 5 seconds (very improbable), 50 clients after 4 seconds and 50 clients after 6 seconds. Each of this distribution reflect a scenario that is very different from the others and from the first one. Here is evident the importance of the lower and the higher values.



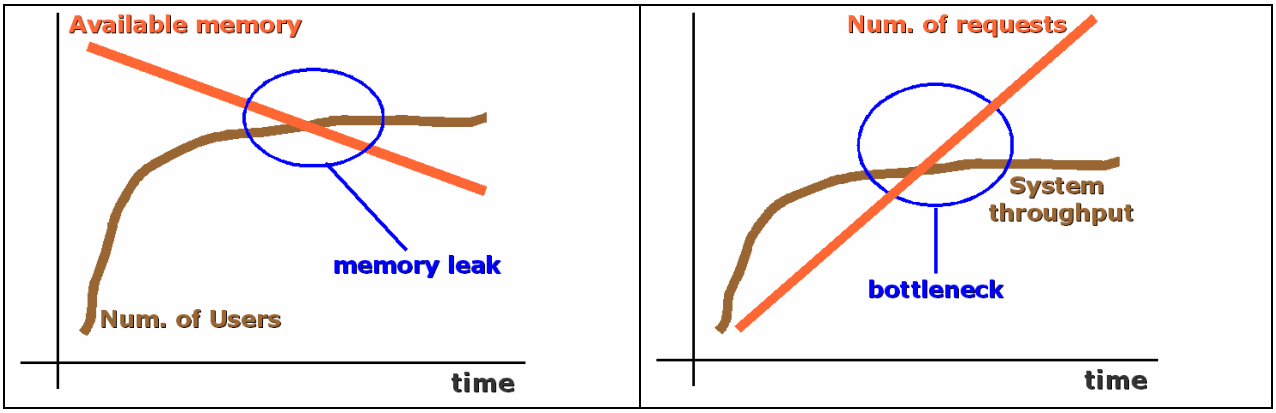
Lower and higher values for a distribution time scenario

Another important parameter that has to be taken into account to have a more accurate analysis is the distribution. There can be various kind of distribution, but the most important for the web services measure analysis is the number of users vs. response time for a fixed number of concurrent requests. It can be clearly represented in a graph where are indicated the different number of users (y-axis) that are served in the corresponding different times (x-axis), for a fixed number of concurrency requests. It is useful to understand the response time performances of the web service. If a different number of concurrency requests will be considered, the distribution trend will change. A suitable distribution is the one that groups small time values for a great number of users.



Number of Users per Response Time for two fixed number of concurrent requests
(1000 req. brown coloured and 2000 req. orange coloured)

Load and stress analysis can also be used to discover implementation bugs. Indeed if system resources are monitored while performing the tests, it can be detected many bad behaviours of the web service such as bottleneck and memory leaks. If the available memory decreases and the load of requests remains nearly constant it will probably be a memory leak. If the number of requests increases and the system throughput doesn't augment it will probably be a bottleneck: the system reached its limit.



Memory leak and bottleneck conditions

It is not necessary to monitor all system resource: often it is sufficient to monitor the usage of base resources as memory, cpu, disk and so on.

8.5 Action Log logic

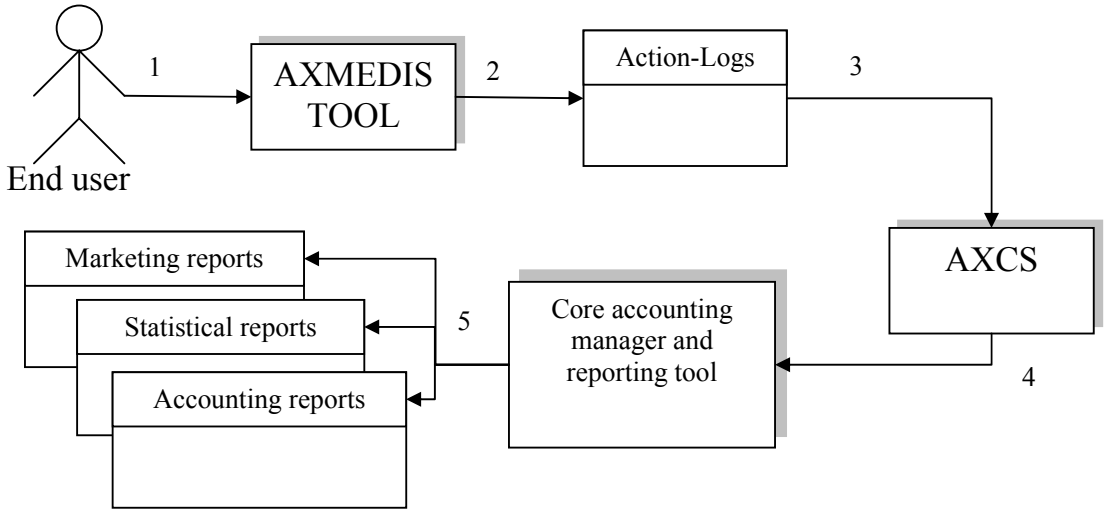
The Action Log is a set of AXMEDIS components that realizes the gathering and elaborating of the accounting and statistical data. These components are:

- AXCS Accounting Database

- AXCS Supervisor
- AXCS Reporting Web Service
- AXCS Statistics Analysis Web Service
- CAMART

Each of them has its own role in the system and is used for a specific function. The AXCS Accounting Database is the recipient where all accounting data are collected. The AXCS Supervisor is the component that writes object usage data in AXCS Accounting Database. The AXCS Reporting Web Service and AXCS Statistics Analysis Web Service are the web services that furnish all data contained in AXCS Accounting Database with some criteria and the appropriate limitations. CAMART is the entity which elaborates data retrieved by these web services.

Here is reported a self-explaining scenario concerning action log data gathering.



1. Distributor performs actions on objects
2. Action-Logs are generated (both on line and off line) reporting actions performed on objects
3. Action-Logs are stored by the AXCS
4. Core accounting manager and reporting tool (CAMART) extract information from AXCS allowing the generation of different report types
5. Marketing reports, Statistical reports and accounting reports can be generated on demand

How it can be seen, the Action log is not a simple tool, but it is a complex distributed system compound by many components that interact in a coordinate way to reach a global goal: take a trace of object usage in the whole AXMEDIS environment and make it available for accounting and statistic purposes.

8.6 AXCS and DMP

As it can be read in the DMP statutes, the DMP is a not-for-profit organisation with the mission to promote continuing successful development, deployment and use of Digital Media that respect the rights of creators and rights holders to exploit their works, the wish of end users to fully enjoy the benefits of Digital Media and the interests of various value-chain players to provide products and services, according to the principles laid down in the Digital Media Manifesto.

The goals of DMP are realised through the development of Technical Specifications and Recommended Practices enabling businesses that support new or improved user experiences, and Recommended Actions to appropriate entities to act on removal of barriers holding up exploitation of Digital Media. Technical Specifications, Recommended Practices and Recommended Actions are collectively called "DMP Approved Documents". DMP contributes the results of its activities to appropriate formal standards bodies and other appropriate entities whenever this is instrumental to achieve the general DMP goals.
The following table reports the mapping of the DMP major technologies/features and their schedule in the phases of the DMP (those features are called Tools in DMP terminology), with the AXMEDIS elements and their module and realization schedule only as far as the AXCS module is concerned.

DMP		AXMEDIS	
Category	Tools/ Major functionalities	Comment	Where
Represent	It is coding		
	• Represent Device Capabilities	HW and SW profile, fingerprint passing	Protection Processor, AXCS table of devices
	Represent Use Data	Action Log, it is an evolution of MPEG21 event reporting, comments and more IDs are in the AXCS Action Log. In the sending transaction and in the Storage. Representing events in DB or in sending them out from the terminal The MPEG-21 Event reporting is not enough for AXMEDIS, this is the real problem…, ….)	Action Log, AXCS DB, etc.
	Represent Resource Format	As MIME type, Technical metadata in AXInfo, + MPEG7	AXInfo, AXCS for the registration of objects
	Represent Use Context (nationality, location, etc., all statically imposed at the installation, probably is more an estimation of certification ??)	IP address, other status conditions to be stated in the Tools profile at the installation (in the disco, in the home, in the office, in car, etc.) Dynamically.	Protection Processor, AXCS, user registration

		In AXMEDIS is missing: status, role, etc..	
Identify	It is a coding		
	Identify Content Data	UUID, fingerprint	AXOID+fingerprint, AXCS for the registration, AXIDG
	• Identify DRM Tool (it is a Content Data Element)	AXTTID, etc.., Fingerprint, SW profile, DLL	Protection Processor, AXCS for the registration of SW tools
	Identify User	UUID + information from the OS	AXUID, AXCS for the registration, AXIDG, web service
	• Identify Distributor (included in identify User)	AXCDID, etc.	AXCDID, AXCS for the registration
	• Identify Producer (included in identify User)	AXCPID, etc…	AXCPID, AXCS for the registration
	Identify Device	AXTID, AXRID, AXTTID as UUID+4code, HW and SW profile, fingerprint	Protection Processor for estimation, AXCS table of devices
	Identify Domain (a domain is a set of Users (for example in a smart card) and/or Devices)	Textual name for the domain, UUID+4 code as device, HW and SW profile, fingerprint	Protection Processor, AXCS for the registration of domains
	Identify Tool (not DRM)	AXTTID, etc.., Fingerprint, SW profile, DLL	Protection Processor, AXCS for the registration of SW tools
	Identify Tool Type	AXTTID, etc.., Fingerprint, SW profile, DLL	Protection Processor, AXCS for the registration of SW tools
	Identify Use Data	An ID to each Action Log.	AXCS, did automatically

Assign	It is info plus protocol, an User ask to some other User to provide the info to be assigned		
	Assign Identifier (all the above identifiers), passing the above information and getting and ID	AXOID Generator, General ID production, local ID generator for Temporary IDs for objects that cannot get out of the AXMEDIS factory. AXMEDIS has a WS formalisation of this protocol	AXMEDIS Editor, AXOM, AXCS
	• Assign Fingerprint, and Descriptors in general as identifier, moved to certify (it can be included in the above Assign Descriptor)	Assign Fingerprint, Fingerprint Technology, AXMEDIS has a WS formalisation of this protocol	FP module, AXCS stores them

Revoke	It is code and protocol		
	Revoke Device (an installed device)	Device Black List into the AXCS DB	AXCS
	Revoke DRM Tool	Device Black List into the AXCS DB	AXCS
	Revoke License	License Black List into the AXCS DB	AXCS
	Revoke User (as Producer, end users, distributor, etc..)	User Black List into the AXCS DB	AXCS
	Revoke Content (an AXMEDIS Object for instance)	Content/Object Black List into the AXCS DB	AXCS
	Revoke Domain	Domain Black List into the AXCS DB	AXCS

Verify	It is protocol and code to passed each other, verification on line with some hash code		
	Verify Data Integrity	Hash code on the client side, Fingerprint on the server	Fingerprint estimation and AXCS, Protection Processor
	• Verify Metadata integrity as belonging to the content, included in the above item	Hash code on the client side, Fingerprint on the server	Fingerprint estimation and AXCS, Protection Processor
	Verify Device Integrity	SW/HW Fingerprint,	Fingerprint estimation and AXCS, Protection Processor
	Verify DRM Tool Integrity	Hash code on the client side, Fingerprint on the server	Fingerprint estimation and AXCS, Protection Processor
	Verify Tool Integrity (not DRM)	Action on Plug-ins and DLLs, Hash code on the client side, Fingerprint on the server, only for AXMEDIS	Fingerprint estimation and AXCS, Protection Processor
Certify	It is protocol	Registration of AXMEDIS includes assign identifier and certification	
	Certify User	User Certification and Authentication, see also registration	Protection Processor, AXCS, a web service is available and a DB
	• Certify Producer, included in the above	User Certification and Authentication, specific IDs are present in AXMEDIS since specify functionalities are present in AXCS for them	Protection Processor, AXCS
	• Certify Distributor, included in the	User Certification and Authentication, specific IDs are present in AXMEDIS	Protection Processor, AXCS

	above	since specify functionalities are present in AXCS for them	
	Certify Device	Device Certification and Authentication, HW and SW fingerprint, local verification and remote verification	Protection Processor, AXCS as counterpart, Web service, etc.
	Certify DRM Tool	Device Certification and Authentication, HW and SW fingerprint local verification and remote verification from AXCS supervision. Each Additional Tool or DLL has to be certified as well	Protection Processor, AXCS as counterpart, Web service, etc.
	Certify Tool (not DRM or not directly)	Device Certification and Authentication, HW and SW fingerprint local verification and remote verification from AXCS supervision. Each Additional Tool or DLL has to be certified as well	Protection Processor, AXCS as counterpart, Web service, etc.

Manage	Means produce, collect and provide??	These actions are more complex since behind the word “Manage” there are many functionalities that could be divided in different functionalities relocated/decomposed to other sections	
	Manage Domain (it includes in DMP also the storing of content)	AXMEDIS Domain Manager as independent Tool to be certified as well, not for storing content	AXMEDIS Domain Manager
	Manage Use Data Confidentiality (including negotiation of their value)	Action Log Server DB and Accounting manager. In AXMEDIS the user has not possibility of deciding much.	AXCS + Accounting manager and DB: Producer, Distributor, Collecting Society., etc.. see only a part of the whole Log

Removed or moved in other section or not needed in DMP			
	Manage Protection Information, it is managed as digital item	Dynamic management of Protection Information (IPMP information), see Access to DRM Tools, etc..	AXCS and Protection Processor, Protection Information Editor and Viewer, etc. Protection Manager Support Server
	Manage Protection Information on Clients	Secure cache on AXMEDIS clients	Protection Manager Support Client and Protection Processor
Access	It is a protocol		
	Access DRM Tool	Protection Information, WEB services of AXMEDIS tools	AXMEDIS Protection Information Editor, AXMEDIS CS for storing Protection information, WEB service
	Access Device capabilities	Access to device profile	AXCS
	Access Use Data Information (coming from Pay section, now almost deprecated or changed in something different)	Access to Action Log	AXCS
Update	It is a protocol		
	Update DRM Tool	Protection Information, WEB services of AXMEDIS tools	AXMEDIS Protection Information Editor, AXMEDIS CS for storing Protection information, WEB service
	Update Metadata (including, Content, Device Capabilities, etc.)	Access to device profile	AXCS

Moved into the Access to Use Data and reworded for generalisation			
	Distributor to Provider	This action is possible in AXMEDIS accessing to the AXMEDIS DB and customizing the tools that allow making reports in XML from the local accounting log to the administrative database. The customisation may allow producing reports for producers/providers or content that ask for the evidence of the distributed content.	AXMEDIS accounting log database and AXMEDIS CAMART tool. AXMEDIS model is based on Multiple AXCS communicating each other and being these certificated by AXMEDIS organisation.
	Distributor Log	Action log tracking access according to the User Type. Information is exchanged for making the bill to other Users or for controlling other Users Business	AXCS, Accounting Database. AXMEDIS model is based on Multiple AXCS communicating each other and being these certificated by AXMEDIS organisation.
	Provider or Integrator Log	Action log tracking access according to the User Type. Information is exchanged for making the bill to other Users or for controlling other Users Business	AXCS, Accounting Database, AXMEDIS model is based on Multiple AXCS communicating each other and being these certificated by AXMEDIS organisation.
	Collecting Society Log	Action log tracking access according to the User Type. Information is exchanged for making the report con IPR owners, recognising revenues returns, etc.	AXCS, Accounting Database, AXMEDIS model is based on Multiple AXCS communicating each other and being these certificated by AXMEDIS organisation.
	Ask for market report	Making a query on the AXCS Accounting and action logs according to the	AXCS, Accounting Database, AXMEDIS model is based on

		level for which the User making the Query has the access to the reported Actions.	Multiple AXCS communicating each other and being these certificated by AXMEDIS organisation.
Test Conformance (postponed for a while)	Protocol and procedure		
	Device registration and acceptance	registration of tools and devices	AXCS acceptance of tools
	Test Conformance of Tamper resistance	Protection Processor, off-line Tool registration, online only as detection of infringement and continuous verification of trusting	Protection Processor, AXCS

9 Table of acronyms and corresponding prefixes

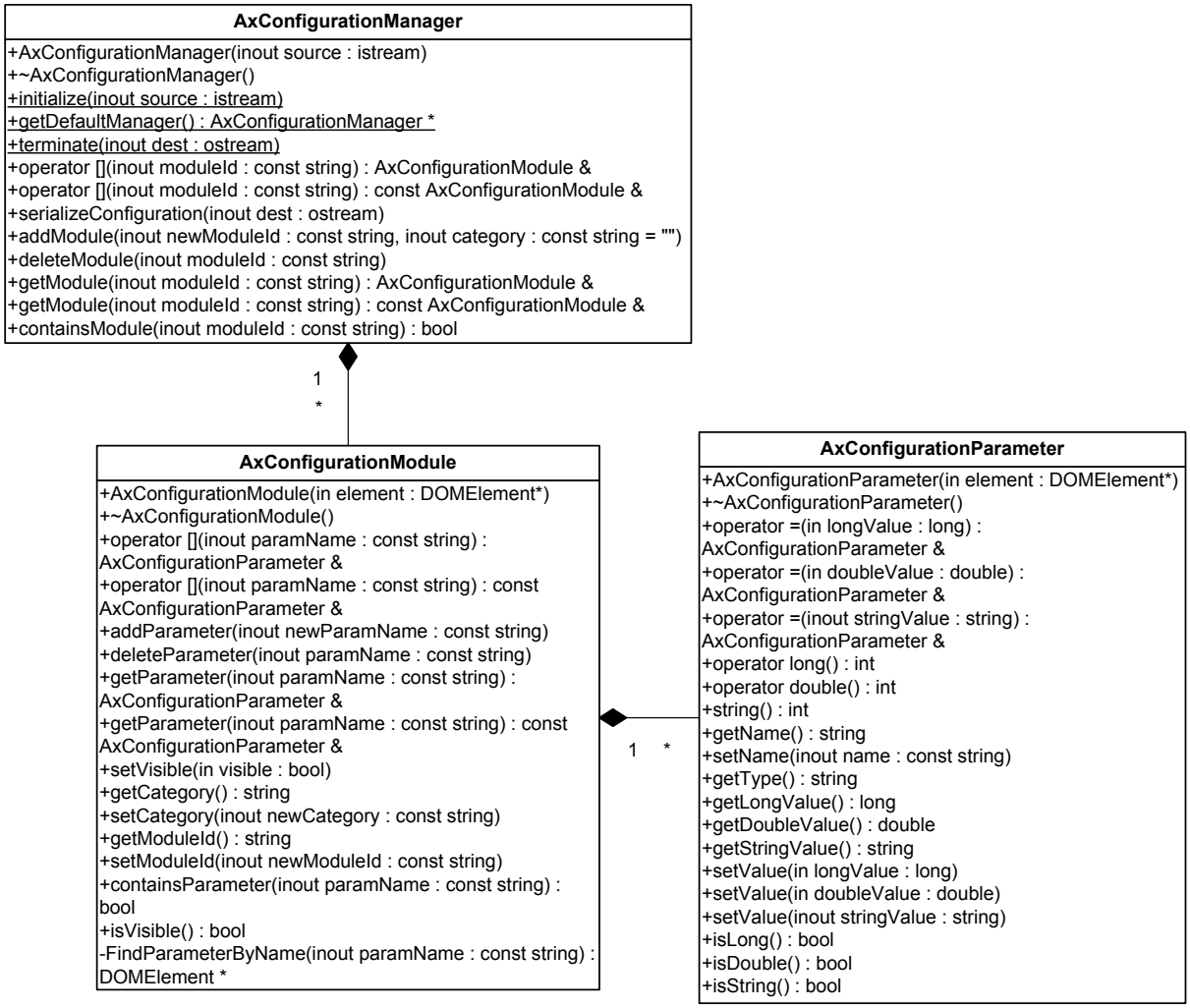
Acronym	Description		ID Prefix
AXCID	Unique AXMEDIS Object Creator ID, can be creators publishers, producers, authors, integrators, etc. All who can produce a new AXMEDIS object. A Creator is the only one entitled to ask for an Object ID.		CRE
AXDID	Unique AXMEDIS Object Distributor ID		DIS
AXDOM	AXMEDIS Domain (associated with the PMS Home or Factory managing the Domain)		DOM
AXLID	Unique AXMEDIS License ID		LIC
AXOID	Unique AXMEDIS Object ID		OBJ
AXRID	AXMEDIS Rule ID		RUL
AXRTID	A specific instance of a AXTID of some type (AXTTID)		RTO
AXTID	AXMEDIS Tool ID, see also AXTTID, versione, anno, lingua, etc…		ITO
AXTOID	AXMEDIS Temporary Object ID, produced directly from the numbers available for the AXMEDIS Creator. It can be produced as AXCID+AXTID+AXUID+an progressive ID of the Tool		TOB
AXTTID	AXMEDIS Tool Type ID Different tools are Editor, Composition Engine, Formatting Engine, Protection Tool Engine, etc.. This is contained into the Profile, several other aspects and details for each tool have to be registered. Some other details depends on the specific instance, this the reason to provide and exploit a Tool Profile.		TOT
AXUID	Unique AXMEDIS User ID	Prefix for Final Users	USR
		Prefix for B2B Users	BUS
AXWID	Work Identification. For each Work you may have several different versions of AXOB, for example differing for resolution, market, format of the digital resources, etc.		WRK
AXCSID	ID of the Collecting Society		CSO
AXTPID	ID of the Tool Producer		TPR

10 Configuration Manager (DSI)

The configuration manager is in charge of storing and retrieving the configuration all the modules of the AXMEDIS framework. An XML schema has been defined to describe how configurations has to be stored in a configuration file which can be read by the configuration manager. The schema defines three elements:

- *Configuration* represents the XML root element
- *Module* contains all the configurations related to a given software module (e.g. AXOM) add comments
- *Parameter* defines a typified couple name/value which can be used to store parameters add comments

The configuration manager allows to parse this type of configuration files and stores the information in a hierarchy of classes. The classes are depicted in the figure below.



AxConfigurationManager provides the following functionalities:

- *initialize* allows to load a default configuration. This function should be called only once at the application start-up
- *terminate* allows to save the default configuration to a stream. This function should be called only once at the application shutdown
- *getDefaultManager* gives the access to the default configuration. This function has to be called only after *initialize* has been called
- *AxConfigurationManager* allows to load configurations from a given stream. This public constructor allows to create configuration manager other than the default one
- *operator[], getModule* allow to access defined modules by name
- *containsModule* tests whether a module with the given name exists or not
- *addModule, deleteModule* allow to manipulate the modules in the configuration manager
- *serializeConfiguration* serializes in XML the configuration on a given stream

AxConfigurationModule provides the following methods:

- *setVisible, isVisible, setCategory, getCategory, setModuleId, getModuleId* allows to get and set the characteristics of the module
- *operator[], getParameter* allow to access defined parameter in the given module by name
- *conatinsParameter* tests whether a parameter with the given name exists or not
- *addParameter, deleteParameter* allow to manipulate parameter in the given module

AxConfigurationParameter provides the following methods:

- *setName, getName, setType, getType* allow to get and set the characteristic of the parameter
- *operator=, setValue* allow to set the value of the parameter. There are three overload for each function: one to set long values, one to set double values and the last to set string values
- *operator long, operator double, operator string* allow to treat the parameter as one of the basic type: long, double or string
- *getLongValue, getDoubleValue, getStringValue* allow to get the value of the parameter on the base of its type
- *isLong, isDouble, isString* allow to test the type of the parameter

Nowadays, the configuration manager works using the Xerces-C++ DOM parser. As soon as possible this dependency will be removed to increase the portability of the configuration manager.

10.1 Technical Details

reference to the AXFW location of the demonstrator	A path in the CVS for example: • No in this version of t he document
List of libraries used	Xerces-C++
References to other major components needed	• Common library
Problems not solved	• Dependency with Xerces-C++
Configuration and execution context	It needs an XML configuration file
Programming language	C++

11 Plug In Manager (DSI)

Technical Details

reference to the AXFW location of the demonstrator	• No in this version of t he document https://cvs.axmedis.org/repos/Framework/doc/test/axom/axom_testing/
List of libraries used	XercesC (for profile parsing)
References to other major components needed	
Problems not solved	• Missing distinction among different plug-in categories • Only Windows • Missing implementation of the ContentProcessing and CommandReporting modules
Configuration and execution context	Presence of plug-in DLLs with their own profiles
Programming language	C++

Plug In Manager: The problems

The activity regarding the Plug-in Manager has been mainly an analysis of the required feature. The present implementation is only for a proof of concepts on the mechanism of dynamic loading and function execution by name. At the present time it presents some feature which are not exactly mapped on what has been nowadays produced by the analysis:

- The plug-in manager is only able to manage plug-ins of the content processing category. Actually the content processing category is the most complex type of plug-in for it expose the most open set of functionalities. The other category are managed with specific interfaces because they have to provide a well-know set of functionalities.

- Since the experiments have been performed on windows platform only a operational plug-in manager for loading DLLs has been produced.
- The ContentProcessing have been analyzed, but not implemented as an independent module. Some of its functionalities are already implemented in the present version of the PluginManager

Plug In Manager: Work performed

The activity performed on the plug-in manager has produced several results:

- In the *AxPluginManager* class dynamic link libraries are managed. This is the only class which is actually platform dependent. In that way is possible to create a new PluginManager for other platforms with different kinds of dynamic link libraries (i.e Linux Shared Objects). The *AxPluginInstance* is the common interface which has to be realized on the different plug-ins: any platform dynamic link library technology has to expose the required interface.
- Content Processing module is responsible of managed plug-ins which contains content processing functions (i.e. resource fingerprint, adaptation…). The particular structure of this plug-in is known by the *ContentProcessing* Module which exposes the interface to invoke a function "by name".
- The *AxPluginFunction* class generalizes a pluggable functionality. A plug-in developer has to inherit from this polymorphic class in order to customize is execute behaviour. Several feature are managed by default in the base class like the type checking of the parameters by looking at the function profile. In the same way even more restrictive controls on data are performed automatically on the basis of function description in the plug-in profile: range of arguments and default value assignment.
- The *AxParameter* is the other class that allows types travelling from application code to plug-in code. The standard types are modelled as the digital resource. The hierarchy is open to definition of new types of parameters and to defines specific resource types like image, audio, video…
- The *CommandReporting* module has been defined with base functionalities which are used in both direction of Workflow messages. This module expose interface to perform actions on the application like "open a new object for editing" inside an AXMEDIS Editor. Since in the workflow manage notification which travels in both in and out flows CommandReporting acts in the middle of this information exchange. Another aspect of the workflow is the application dependent behaviour, the workflow provides different feature if it is plugged in a AXMEDIS Editor (of objects) or in a Rule Editor or in a Rule Scheduler. An additional generalization has been created in order to model an Application Dependent Workflow set of functionalities. In this way the plug-in have the knowledge of which is specific interface will use inside the hosting application (i.e. an AXMEDIS Editor Workflow plug-in is developed on the basis of *AxObjectEditorWorkflow*.
- The plug-in profile have been analyzed and has been restructured in order to provide different information for each plug-in category (e.g., content processing plug-ins have to expose function descriptions, protection plug-ins have to declare the ProtectionToolId implemented in his encoding/decoding functions.

12 Internal Audio Player (DSI)

The Internal Audio Player allows to play audio files on a Windows PC.

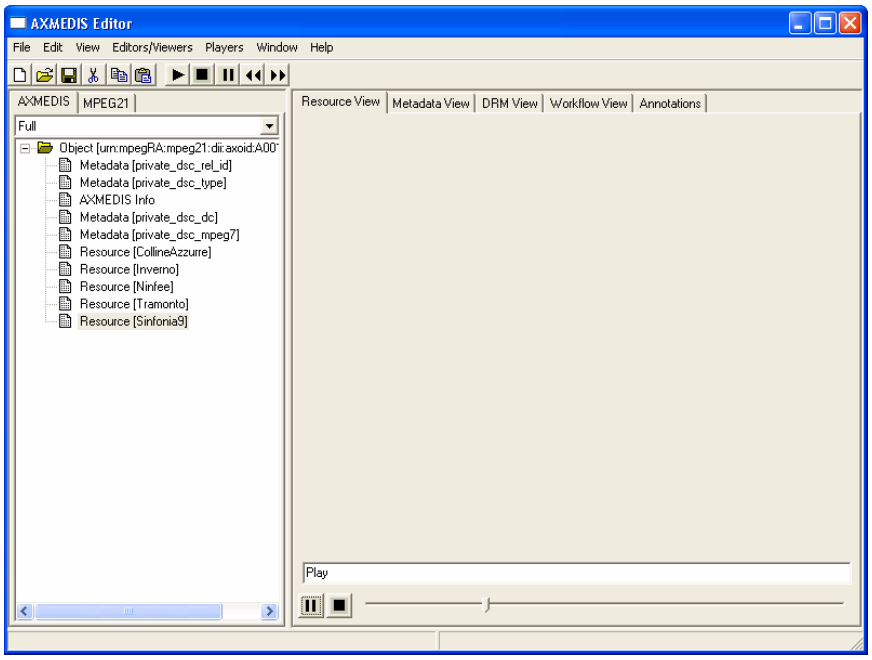
12.1 Technical Details

Reference to the AXFW location of the demonstrator	• No in this version of t he document
List of libraries used	wxWidgets, DirectShow
References to other major components needed	AXOM
Problems not solved	play an audio obtained from a stream
Configuration and execution context	
Programming language	C++

12.2 Internal Audio Player: Work performed

The Internal Audio Player prototype, in the current version uses the MS DirectShow to play audio files, it allows to:

- load an audio file from File System
- play/stop/pause an audio file
- seek the playing position
- show the current position during playing



13 Internal Image Viewer (DSI)

The Internal Image Viewer allows to view images.

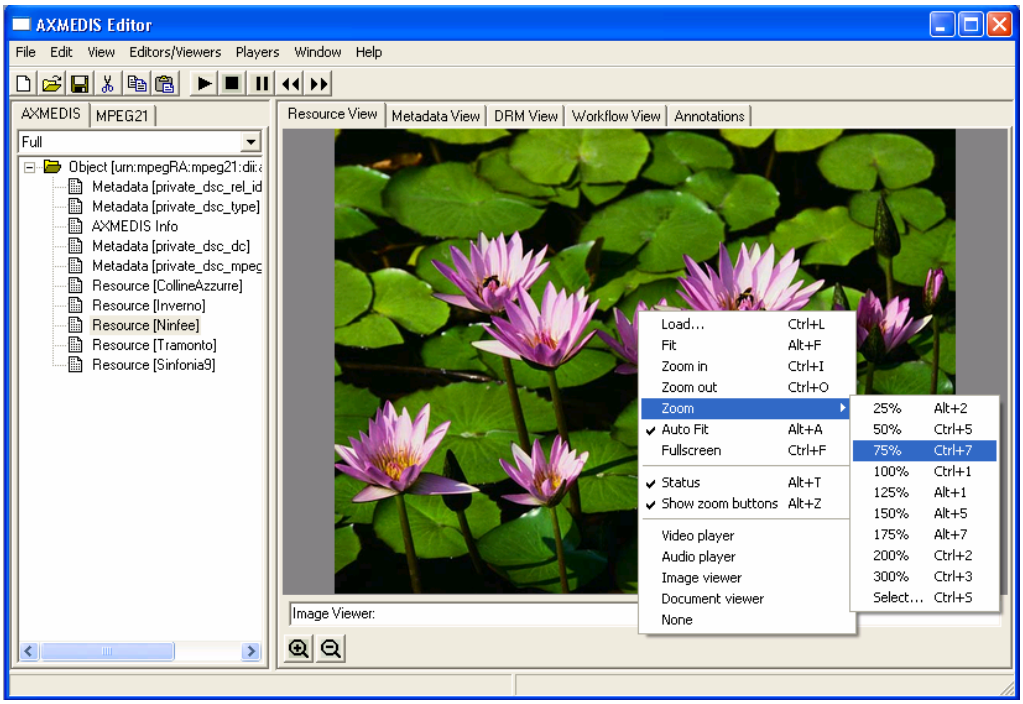
13.1 Technical Details

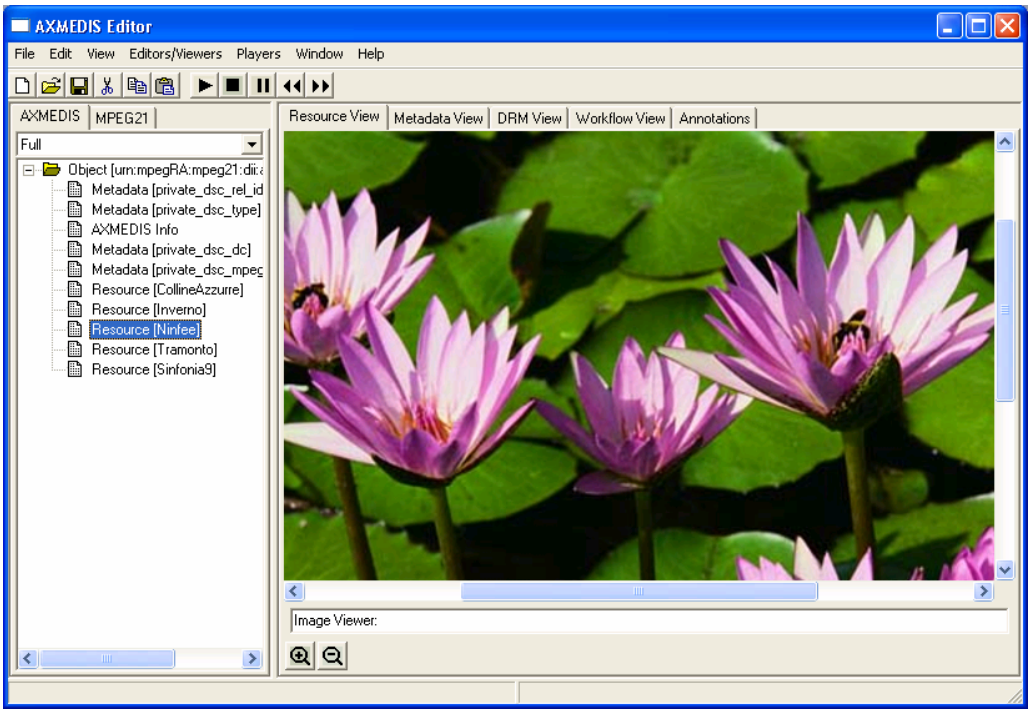
Reference to the AXFW location of the demonstrator	• No in this version of t he document
List of libraries used	wxWidgets, wxImagick, Image Magick
References to other major components needed	AXOM
Problems not solved	it does not allow to view multiple images files, show an image obtained from a stream
Configuration and execution context	
Programming language	C++

13.2 Internal Image Viewer: Work performed

The Internal Image Viewer allows to:

- load an image from file
- zoom the image in and out
- pan the image
- fit the image to the window size
- show the image full screen





14 Internal Video Player (DSI)

The Internal Video Player allows to view videos.

14.1 Technical Details

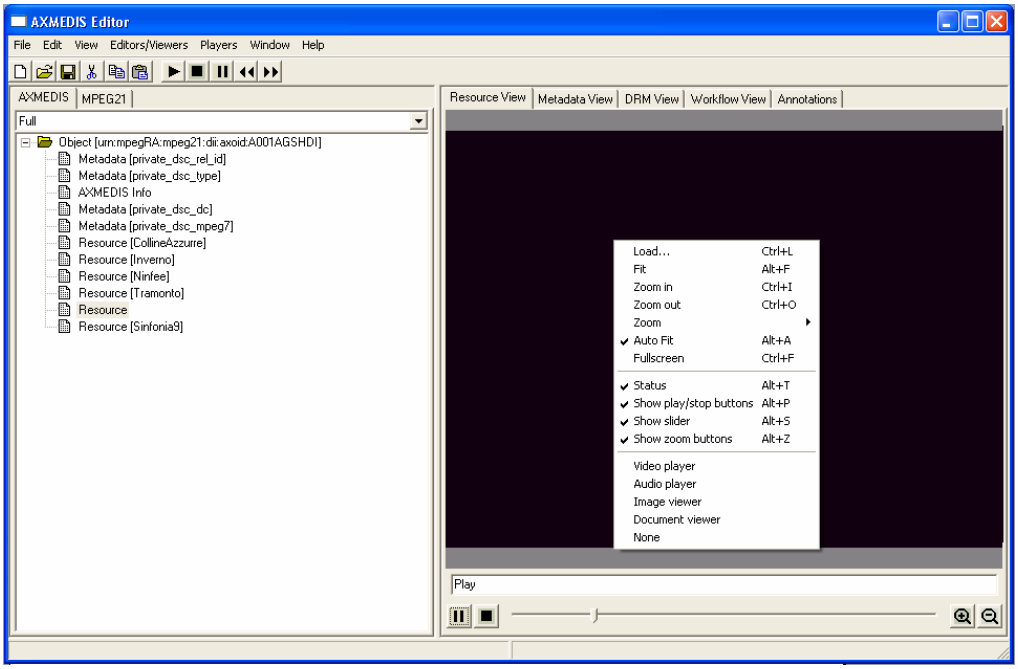
Reference to the AXFW location of the	• No in this version of t he document

demonstrator	
List of libraries used	wxWidgets, DirectShow
References to other major components needed	AXOM
Problems not solved	show a video obtained from a stream
Configuration and execution context	
Programming language	C++

14.2 Internal Video Player: Work performed

The Internal Image Viewer allows to:

- load an video from file
- play/pause/stop the execution of the video
- zoom the video in and out
- fit the video to the window size
- show the video full screen



note: the video content is missing since simple screen grabbing does not get the video content

15 Internal Document Viewer (DSI)

The Internal Document Viewer allows to view documents using the Internet Explorer ActiveX.

15.1 Technical Details

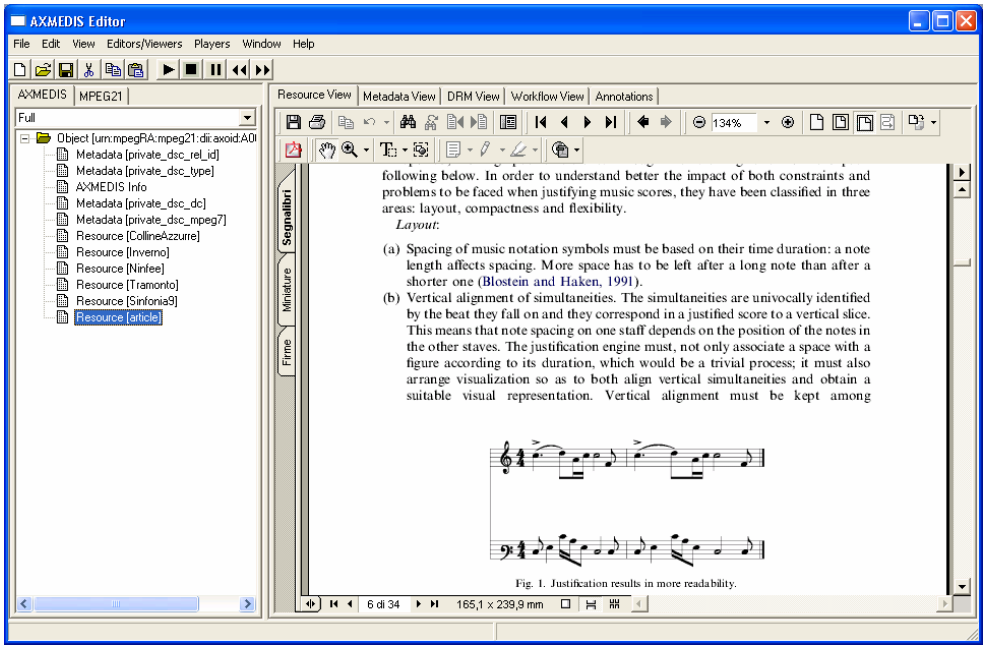
Reference to the AXFW location of the demonstrator	• No in this version of t he document
List of libraries used	wxWidgets, wxactivex
References to other major components needed	AXOM
Problems not solved	show a document obtained from a stream
Configuration and execution context	
Programming language	C++

15.2 Internal Document Viewer: Work performed

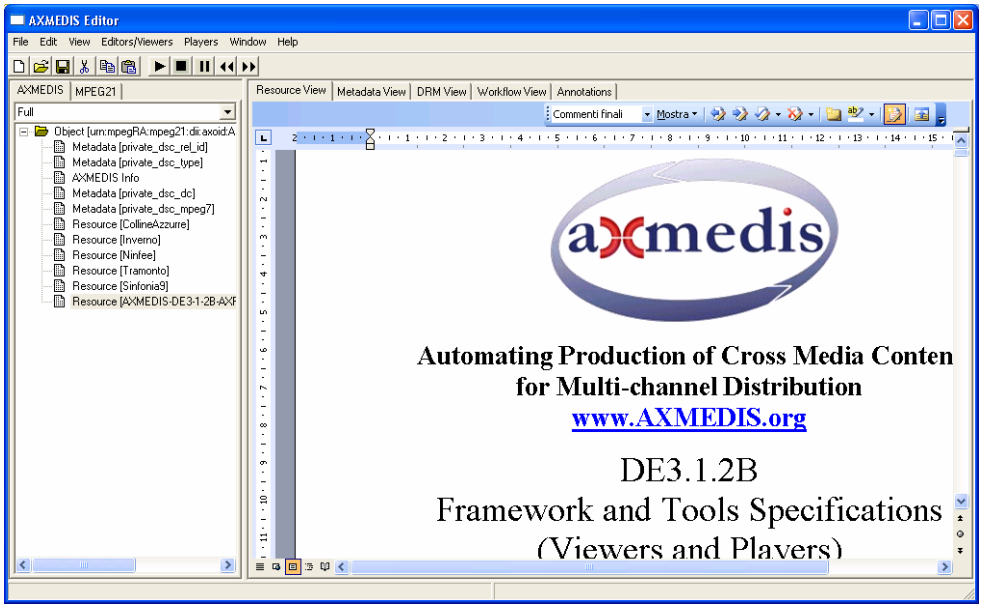
The Internal Document Viewer allows to:

- load a document from file
- show the document full screen

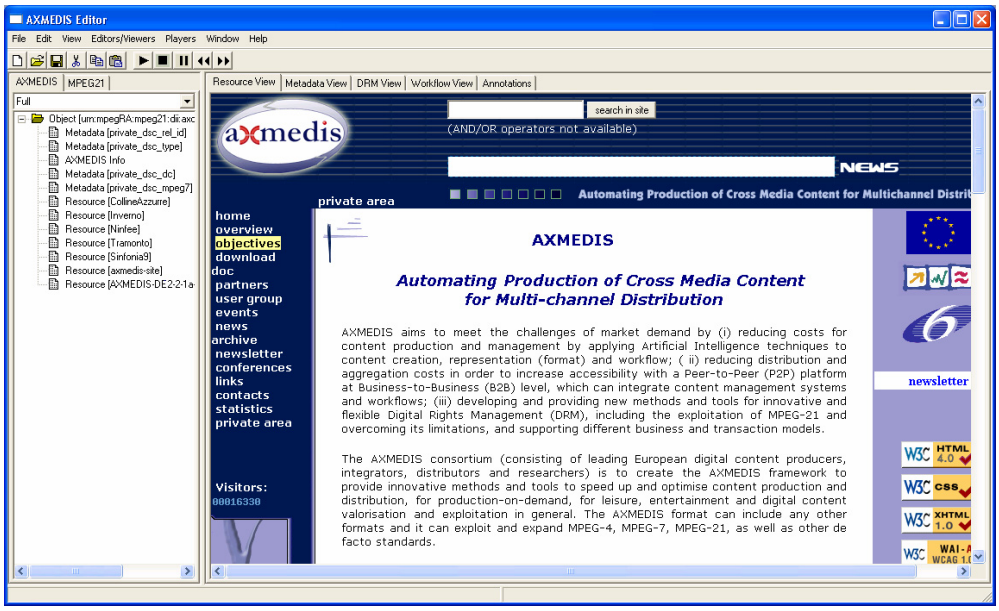
In the following picture the Internal Document Viewer shows a pdf file:



In the following picture the Internal Document Viewer shows a doc file:



In the following picture the Internal Document Viewer shows link to an internet site:



16 Internal MPEG-4 Player (EPFL)

16.1 Analyis of MPEG-4 players from EPFL

The work done at EPFL for the Internal MPEG-4 Player mainly consists of a review of the Splay Player available at the EPFL and of an evaluation of integration into the AXMEDIS Framework.

Splay is an MPEG-4 Player whose core was originally developed by bSoft srl, Italy, for which the complete C++ source code is available at EPFL; Splay was first used in IST-CARROUSO (5th framework) when EPFL included in the Player support for audio and advanced audio functionality; it is currently being used in IST-ENTHRONE, where bSoft and EPFL are still collaborating for the IPMP-X integration and porting on mobile platforms.
The possible use of Splay within AXMEDIS has not been considered yet, but it may be a good candidate. However, big question marks still exist concerning the agreement with other partners inside AXMEDIS, licensing terms with bSoft and coordination with ENTHRONE.
Nevertheless, some information about Splay is presented in the following.

Splay features

The current implementation of the player supports audio and video formats in conformance with the ISO/IEC 13818 (MPEG-2) and 14496 (MPEG-4) standards. In particular, the tools and functionalities currently available for a possible integration in the AXMEDIS terminal platform might be:

- MPEG-2 Video: Main Profile at Main Level decoding.
- MPEG-2 Audio: support of Layers 1, 2, 3.
- MPEG-2 Systems: a full MPEG-2 Systems implementation is not available. Depending on the application scenario, implementing Transport Stream and/or Program Stream management should be achievable; management of the whole DVB stack is more complicated and can be implemented in enhanced players.
- MPEG-4 Video: support of Simple Profile, and handling of Advanced Simple Profile by skipping B frames.
- MPEG-4 Audio: support of Main Profile and below (Low Complexity, Long Term Prediction).
- NB: MPEG-4 on MPEG-2 Transport Stream is also available.

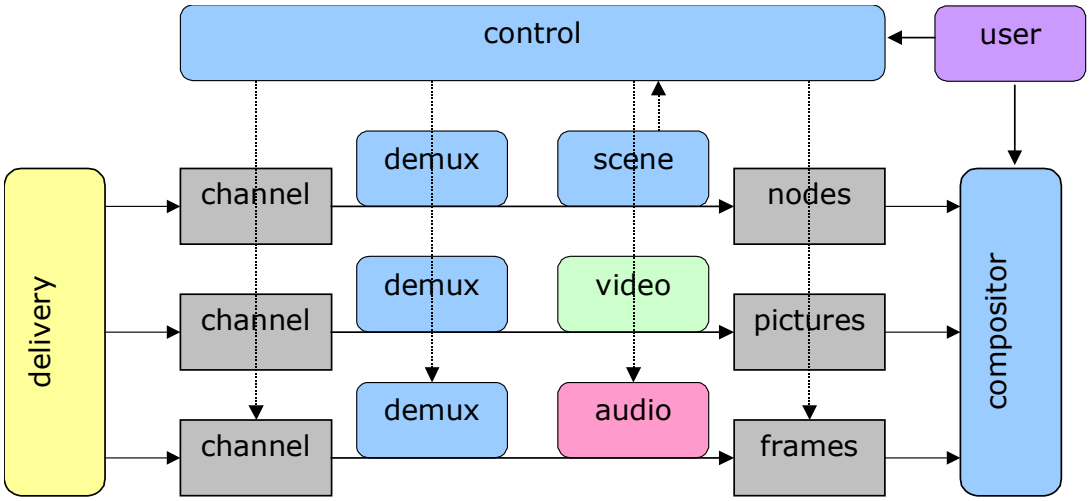
- MPEG-4 Systems: support of the complete Object Descriptors and Nodes framework; IPMP(X) functionality are integrated.

Splay implementation

Splay is a modular software system, with elementary decoders implemented as separate static or dynamic libraries (MPEG-4 Audio, MPEG-4 Video, other audio-video decoders), another module for managing the MPEG-4 Systems components (Initial Object Descriptor, Object Descriptors, Scene Descriptions), and additional modules specific to the target platform (e.g. audio and video playback), or specific to the delivery scheme (e.g. MP4 file reader, RTP receiver).

Going to a closer view of the structure of the Splay MPEG-4 Media Player, the figure below shows more detail of the internal parts and provides also a reference for the software architecture of the implementation.

Besides the essential Audio Decoder (light red) and Video Decoder (light green), which support the core functionality of audio-video playback, the full featured MPEG-4 player shall manage an arbitrary number of elementary stream decoders (Audio, Video, and possibly other Media), plus the infrastructure for the description of the rich media scene, and timing plus synchronization of the multiple elementary streams.



16.2 Analysis of MPEG-4 players from DSI

The work done at DSI for the Internal MPEG-4 Player mainly consists of a review of the IM1 and OSMO y Players. Both of them are open source players and could be integrated into the AXMEDIS Framework while licensing problems have to be addressed.

A document has been produced on the analysis of IM1 and OSMO in Italian language.

IM1-OSMO-Relazione-v13.doc

17 Internal SMIL player (EPFL)

The Internal SMIL Player allows viewing SMIL multimedia scenes composing different media types with precise space and time synchronization. SMIL does not define any particular type of media (such as vector or raster images, videos, text, or audio data). Instead of media content, SMIL describes media composition that is the layout of the different elements on the screen, as well as their time attributes. SMIL describes temporal and spatial organization of media while not defining the content itself. Every type of media – even Flash content – can be part of a SMIL animation. The SMIL language is based on XML, and thus is text based, making it very easy to generate, even from a database and a middleware.
The internal SMIL player is based on the open source AMBULANT player. Some basic modifications are nevertheless necessary before having a first prototype available.

17.1 Technical Details

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

17.2 Internal SMIL Player: work performed

The work done at EPFL for the Internal SMIL Player mainly consists of a review of the AMBULANT Player available as source code and of an evaluation of integration into the AXMEDIS Framework with initial modifications of the code in this direction.

The AMBULANT Open SMIL Player is an open-source, full SMIL 2.0 media player. It may also be used as a complete, multi-platform media player for applications that do not need support for closed, proprietary media formats. The AMBULANT player written in C++, is distributed under a modified GPL license. Currently the AMBULANT player delegates the rendering of images, video, or audio to third-party specialized libraries.

A more detailed description of integration issues concerning AMBULANT inside the AXMEDIS Framework has been presented in deliverable 3.1.2 Part B (Framework and Tools Specification (Viewers and Players)). In terms of practical work, the AMBULANT player interface is normally based (for its Windows version) on MFC.
An analysis of the wxWidget main features allowed to start a substitution of MFC by wxWidget. As a result a first draft version of the AMBULANT SMIL player running embedded in a wxWidgets application has been produced. The user can start the most standard operations of a player through this interface:

- play,
- stop,
- pause,
- close,
- open

Further modifications to the interface and more inside the Player structure have not been possible yet.

Above is a snapshot of the AMBULANT player built with wxWidgets. It runs the sample SMIL file of the NewYorGeo Demo, which is a Synchronized Multimedia file integrated with text and images.

18 External Editor/Viewer Activation Manager (EXEVAM) (DSI)

Demanded to the next period of work.

18.1 External Editor/Viewer Activation Manager: The problems

No effort has been spent to implement External Editor/Viewer Activation Manager. No relevant problems arise from this lack a part from the fact that an Axmedis Object (with its resources) can only be manipulated with the AXMEDIS Editor.

18.2 External Editor/Viewer Activation Manager: Work performed

The External Editor/Viewer Activation Manager has been considered a low priority task, thus none of its functionalities have been implemented.