Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE12.1.4.1.
AXMEDIS-4HOME Component Integration, Prototypes and Documentation

Version: 1.0
Date:10-03-2008
Responsible: BBC

Project Number: IST-2-511299
Project Title: AXMEDIS
Deliverable Type: report
Visible to User Groups: yes
Visible to Affiliated: yes
Visible to the Public: yes

Deliverable Number: DE12.1.4.
Contractual Date of Delivery: M42
Actual Date of Delivery:10-03-2008
Title of Deliverable: DE12.1.4.1. AXMEDIS-4HOME AXMEDIS Component Integration, Prototypes and Documentation
Work-Package contributing to the Deliverable: WP12
Task contributing to the Deliverable: WP12
Nature of the Deliverable: report
Author(s): BBC,sDae,PKU,ETRI,TI

Abstract This document describes the prototype components developed by the 4HOME partners and contains the documentation of these components including where appropriate, their integration details.

Keyword List: DVB-T, OMA, Domains, Tool Servers, Rights Ontology

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 Content

1 Executive Summary and Report Scope

This report describes the development of the prototypes produced by the AX4HOME Take up action. These prototypes together form the experimental and prototype concepts that demonstrate the use of the AXMEDIS framework tools as implemented by the AXMEDIS project, to address the requirements and technical specifications described in the AX4HOME documents DE12.1.2 and DE 12.1.3. This report addresses the activities of the individual prototype development. These are;

1. The AX4HOME DVB-T recorder and data enhancer for the home.
2. The DRM Protection Tool server development
3. The integration of the User Domain based on devices into the AXMEDIS PMS
4. The development of the Rights Ontology Demonstrator
5. The development of the 4HOME content factory and OMA gateway.

2 The 4home Client Recorder, Description of Software, Scripts and Use of AXMEDIS Tools (BBC)

2.1 Overview

The BBC contribution to the demonstrator is based around the capture of off-air content and the subsequent creation of an AXMEDIS object on the client from a number of sources, including the captured file.

The work is comprised of the following components:

- The broadcast content capture from a consumer DVB-T PC card.
    - This involves building a means to schedule a recording from a web site and subsequently capturing the correct file. The interface between the DVB PC card and the 4HOME application is through the Microsoft DirectShow interface and is written in C++
    - The recorder also interacts with the AXMEDIS environment to initiate the creation of an AXMEDIS object and present the object on completion to the user.

- The design of a suitable user interface to interact with the broadcast recorder application, the recording schedule and the AXMEDIS environment for playback

- The design of a SMIL based presentation of the AXMEDIS object content created using the AXMEDIS tools and presenting the recorded file, TV-Anytime data and the broadcast 'enhancements' to the user of the system as part of the AXMEDIS object playback.

- Developing the AXMEDIS scripts, on the AXMEDIS Rule Editor, to automatically acquire the components and generate the AXMEDIS objects, with the associated dynamic SMIL presentation, on the home client adaptively, based upon the resources found within the enhancement and the TV-Anytime metadata associated with the requested programme.

- Installation of the AXMEDIS Rule Executor on the home client to execute the script, triggered from the DVB-T recorder. This needs to communicate with the PMS , AXMEDIA download and have access to the transcoder plug-in.

2.2 Broadcast Content Capture

2.2.1 Core Application Development

The core Ax4Home Recorder application is a native Microsoft Windows C++ application, consisting of an Microsoft Foundation Classes (MFC) based front-end, statically linked to a library containing the core functionality. It requires a Hauppauge WinTV DVB-T capture device. Microsoft DirectShow components are used to control off-air reception, receive, demultiplex and capture the broadcast video.

A separate executable is registered in the Microsoft Windows Registry as a "protocol handler" for the "crid://" protocol. This provides the bridge, from a user clicking on a programme listing on the Electronic Programme Guide (EPG) web page, to a recording request being scheduled in the Ax4Home Recorder application. The executable communicates the request via an intermediate DLL.

The EPG web page identifies programmes by Content Reference Ids (CRIDs) – which are unique content identifiers, defined in the TV-Anytime standard. These Ids are mapped to Service IDs and Event IDs that will identify the programme in the broadcast stream. The mapping is performed by querying a TV-Anytime http server.

The recorder application continuously runs a DirectShow filter-graph containing Broadcast Driver Architecture components to control the Hauppauge receiver device; receive the MPEG transport stream and demultiplex an audio and video stream. It it also used to extract basic “now and next” programme schedule information, allowing the recorder to determine what is currently being broadcast on each channel.

When the “now and next” information changes, this indicates that a programme has started. The recorder determines if it needs to be recorded by matching the DVB Service ID and Event ID of the scheduled requests to the Service ID and Event ID of the newly commenced programme.

To start a recording, the recorder reconfigures the demultiplexer to demultiplex the channel that needs to be recorded. The filter graph also contains a custom component, designed to intercept the demultiplexed audio and video stream and save them to disk. When not recording this component is simply configured to ignore the data. When recording, it is ordered to write it to a file.

When a recording completes, it is added to an internal job queue. A separate thread processes jobs from the queue one at a time. It invokes the AxMedis Rule Executor, to post-process the video, creating the final AxMedis object that combines the recorded video and enhancements. The rule script is customised (a simple textual substitution step) for the specific details of the video it (such as the location of the source video file, and the CRID identifying the programme).

The recorder also embeds the ActiveX version of the AxMedis player to allow the resulting objects to be played within the application.

Current limitations:

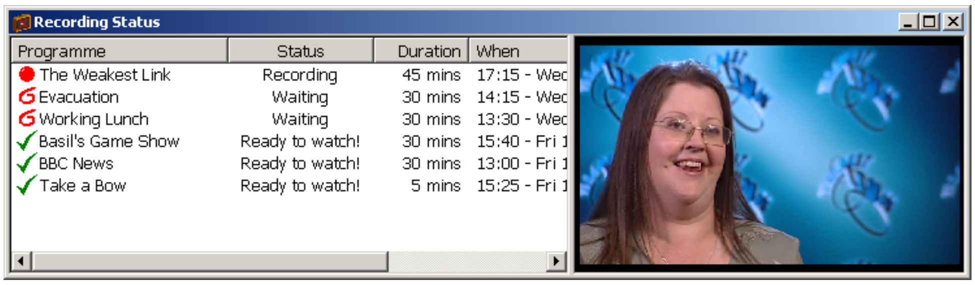
- The recorder does not support re-tuning to a different broadcast multiplex frequency at runtime. It therefore can only record programmes on channels contained in the multiplex to which it is tuned.
- The recorder can only record one programme at a time. Any conflicts are only detected when the recorder tries to start recording a second programme. They are resolved by continuing the existing recording and ignoring the second, or any other conflicting request.
- The recorder is tied to specific Hauppauge DirectShow filter components. It will therefore is currently unlikely to work with other manufacturers' DVB capture devices.

2.2.2 Recorder User Interface

The user interface for the recorder prototype is based around a demonstration of the recording capture and playback.

During development, work has been done to improve the initial base prototype to give access to the features of the recorder in a more user-friendly manner.

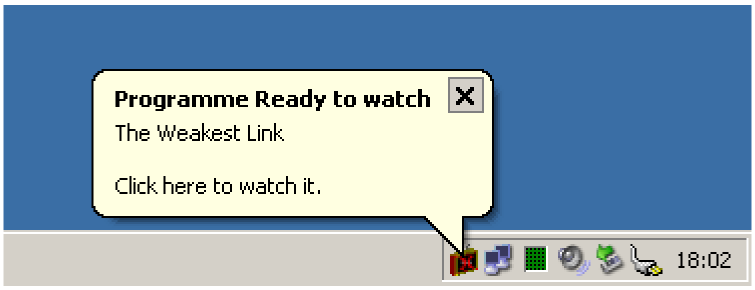
Built using the MFC components and C++, the interface comprises the ACTIVE-X AXMEDIS player in a frame that also encapsulates the live TV preview window, a directory listing of the AXMEDIS Objects available and a general purpose ‘status output’ text box, to allow monitoring of the recorder activity and status.



Programme	Status	Duration	When
● The Weakest Link	Recording	45 mins	17:15 - Wec
Evacuation	Waiting	30 mins	14:15 - Wec
Working Lunch	Waiting	30 mins	13:30 - Wec
✓ Basil's Game Show	Ready to watch!	30 mins	15:40 - Fri 1
✓ BBC News	Ready to watch!	30 mins	13:00 - Fri 1
✓ Take a Bow	Ready to watch!	5 mins	15:25 - Fri 1

Figure 1:The recorder application showing a list of objects and their status.

General recorder status is also reported to the user through a status information column in the list of scheduled recordings and also by pop up notifications.


Programme Ready to watch

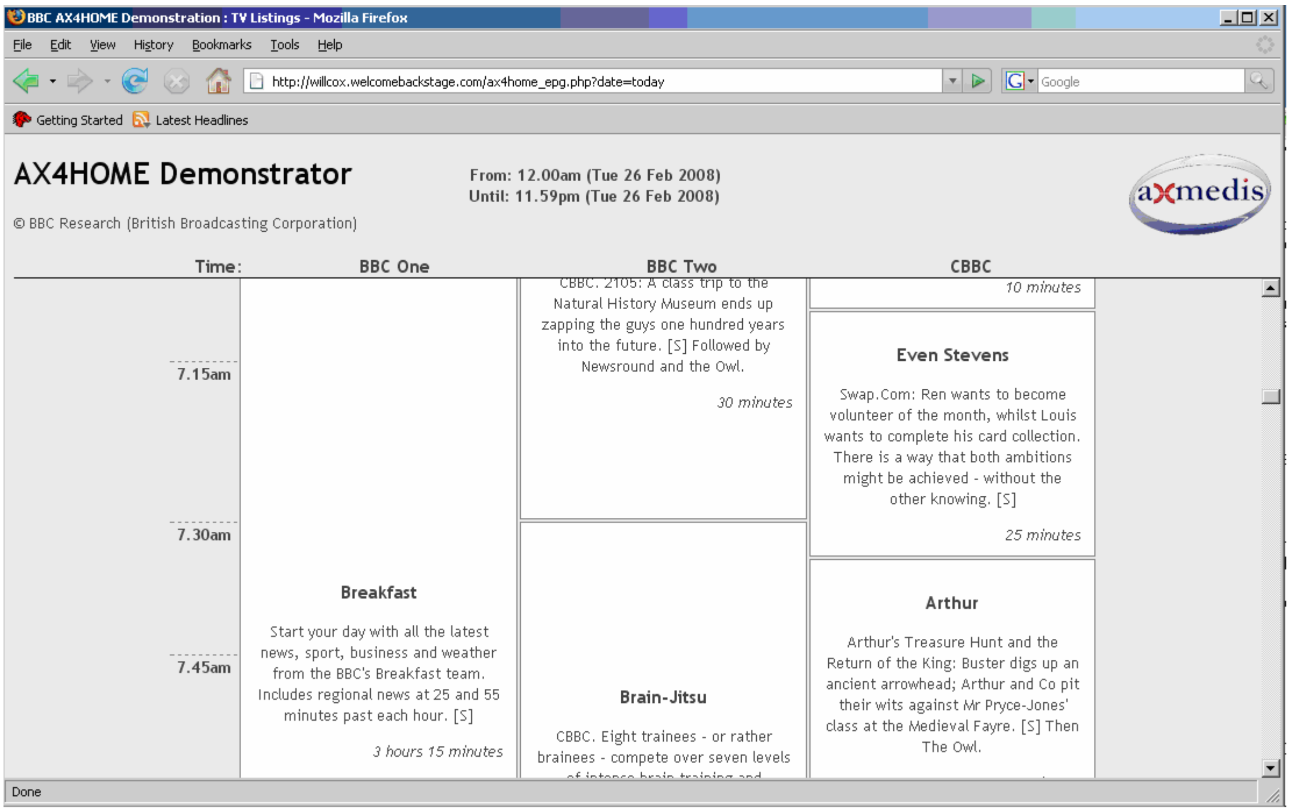
The Weakest Link

Click here to watch it.

Figure 2: Pop-up notifications alert the home viewer when recordings begin or when a programme is ready to watch.

In addition to the listing of AXMEDIS objects, the active component allows automatic loading of the AXMEDIS objects into the ACTIVE X player by double-clicking on the selected text item.

User selection of a programme to be recorded from the broadcast schedules is made though a web browser EPG, hosted by the BBC. The user sees a grid of programmes organized by service and time. Each entry gives the title and duration of the programme. Once selection is made by clicking on the chosen programme square the recorder automatically scheduled to record.


BBC AX4HOME Demonstration : TV Listings - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://willcox.welcomebackstage.com/ax4home_epg.php?date=today
Google
Getting Started Latest Headlines
Done

AX4HOME Demonstrator

© BBC Research (British Broadcasting Corporation)

From: 12.00am (Tue 26 Feb 2008)
Until: 11.59pm (Tue 26 Feb 2008)

Time:	BBC One	BBC Two	CBBC
7.15am	Breakfast Start your day with all the latest news, sport, business and weather from the BBC's Breakfast team. Includes regional news at 25 and 55 minutes past each hour. [S] 3 hours 15 minutes	CBBC. 2105: A class trip to the Natural History Museum ends up zapping the guys one hundred years into the future. [S] Followed by Newsround and the Owl. 30 minutes	10 minutes
7.30am		Brain-Jitsu CBBC. Eight trainees - or rather brainees - compete over seven levels	Even Stevens Swap.Com: Ren wants to become volunteer of the month, whilst Louis wants to complete his card collection. There is a way that both ambitions might be achieved - without the other knowing. [S] 25 minutes
7.45am			Arthur Arthur's Treasure Hunt and the Return of the King: Buster digs up an ancient arrowhead; Arthur and Co pit their wits against Mr Pryce-Jones' class at the Medieval Fayre. [S] Then The Owl.

Figure 3: Web service used to select programme to be recorded

The recorder automatically instantiates the AXMEDIS Object building process once the programme file has been captured off air by invoking the AXMEDIS RuleExecutor and loading the pre-formed script.

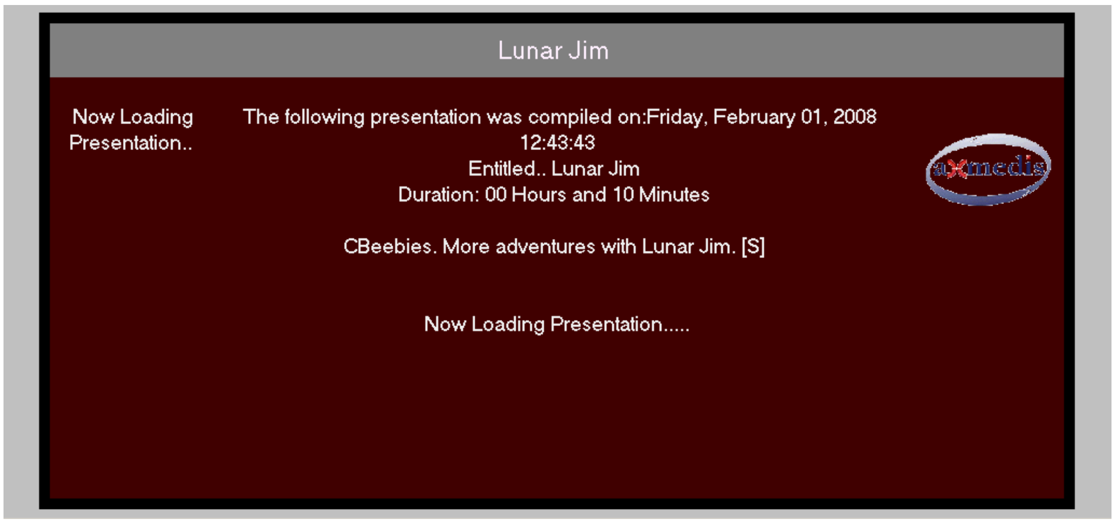
2.3 Creation of the AXMEDIS Presentation.

The creation of the AXMEDIS object on the home client is performed through execution of the AXMEDIS RuleExecutor. The RuleExecutor executes a script stored on the home client and developed with the AXMEDIS RuleEditor. The development of this script is closely related to the design of the presentation of the off-air recording and the associated resources to be added from other sources, such as P2P downloads and the HTTP TV-Anytime programme data server

2.3.1 Design of the Broadcast Presentation

The presentation of the programme recorded off air takes the form of a sequence of presentations. Beginning with the textual presentation of time of recording, programme title and other details from the online metadata service, the side panels show the AXMEDIS logo and a message that the main presentation is loading. This page is presentated whilst the player loads the video presentations.

The video presentations are in 16:9 format and occupy the central panel, leaving the two side panels for logos and programme summaries. Typically the first presentation would be a seasonal promotion, related to service the main off-air programme was taken from. Then a channel ident can follow, and finally the recorded programme. Other areas include the header banner displaying the name of the recorded programme.


Lunar Jim

Now Loading Presentation..

The following presentation was compiled on:Friday, February 01, 2008 12:43:43
Entitled.. Lunar Jim
Duration: 00 Hours and 10 Minutes

CBeebies. More adventures with Lunar Jim. [S]

Now Loading Presentation.....

Figure 4:View of presentation during opening of presentation

The presentation is contained in a W3C SMIL presentation, generated on the client after the programme is captured and inserted into an AXMEDIS object along with the associated resources. The SMIL generation is based upon a stored presentation 'template' – a valid smil script that contains the basic layout, colours and font settings, but does not contain the references to any audio-video resources or textual description of the resources, since these have to be generated in the context of the recorded programme by the home client, and the basic template does not have any prior knowledge of the number or nature of these. The home client extends the template into a full script by executing an AXMEDIS Rule script.

2.4 Functionality Contained in the AXMEDIS Rule Script

The script is generic in that the enhancements are displayed first and then the off-air programme. The nature of the enhancements, eg, news or weather reports, is not determined by the script, but by the author of the enhancements, independent of the user selection. The determination of the selection of the enhancement to use to wrap the off-air recording is set by the script author, and in the case of the prototype depends on a simple algorithm that matches the broadcast service channel (eg, BBC one) to the latest enhancement in the download directory (eg, BBCOne_enhancement).

Using the AXMEDIS Rule Editor, the script that executes on the home client to create the AXMEDIS object implements the following operations.

1. Find the file. The filename of the off-air recording is inserted into the script ‘on the fly’ by the recorder broadcast capture application. It is encoded with the DVB service ID and Event ID, used by the recorder to capture the file from DVB-CPT. Additionally, the TV-Anytime Content Reference ID (CRID) of the programme has been retained from the EPG selection, and this is also inserted into the input script on the fly, so acting as an input parameter to the AXMEDIS RuleExecutor.

2. Retrieve TV-Anytime Data. Using the programme CRID, the AXMEDIS Script enables an HTTP: connection to the BBC TV-Anytime metadata server to be established and programme information to be returned as an XML document.

3. XML4J. Using the AXMEDIS Script XML4J functionality, the incoming TV-Anytime metadata is parsed and textual objects formed containing the main items for presentation, including Title, Service name, Synopsis, related media URL and so on.

4. Building the SMIL. As the textual objects are created, so the SMIL template is expanded to produce an output .smi file. This process is extended further as the related enhancement.axm is located and opened. The number of objects within the enhancement is not predetermined but discovered during script execution. The corresponding output smil entries are authored according to the number of discovered entries in the enhancement. The text displayed during the rendering of the promo videos is currently generic.

5. Building the AXMEDIS object. The script then assembles the resources into an AXMEDIS object, placing the newly generated SMIL presentation first in the order. The AXMEDIS object can be written out as ,mp21 or .axm, depending on the internal state of a Boolean flag.

6. Optionally, the videoadapter plugin, circulated with the tools can be invoked on the recorded file.

7. Optionally, the protection algorithm can be invoked on the object. This also creates an issues and user license which is stored remotely (posted) on the PMS server.

2.5 General Observations on the AXMEDIS tools

The development of the demonstrator requires the application of a succession of processes involving AXMEDIS tools. For proper development, consistent behaviour between all these tools – Player, ActiveX Player, Editor, Rule editor and Rule Executor is essential

The demonstrator development also requires the use of some specific AXMEDIS functionality, such as protection of large file, the presentation of large files, the un-protection of Axmedis objects. As a result, the speed of presentation is an issue the AXMEDIS partners are still addressing. Also, any bugs in the PMS system have a significant effect on the development as each AXMEDIS tool requires registering and the process has to be repeated on every machine used. Failure to perform these operations when the tools are updated, doing it inconsistently or violating the PMS checking can prevent the completion of a chain of events and cause end user frustration.

General comments on the tools themselves now follow.

Rule Editor

The use of the AXMEDIS Rule Editor is quite intuitive, especially if time is taken to manually create and object using the AXMEDIS editor first to familiarize oneself with the correct actions and their intended outcome. Added to this is the provision of rule examples covering the basic operations in javascript which allow the easy assembly of the script. This approach is preferable to a low level documentation of the API description which although available, lags the development of the AXMEDIS tools. The posting of script fragments was a very important factor.

There have been problems with saving files as .mp21 which the AXMEDIS consortium has addressed. There have also been problems with PMS operation and alignment of the tools as upgrades have been issued. Finally there are now some environmental issues to do with the residual files caused by execution of the rule editor and other tools. These were noted in the Project Review of February 2008.

The Rule Executor.

The RuleExecutor works as intended on scripts tested on the Rule Editor. Where there is common code deployed, the RuleExecutor shows the same issues, though no additional issues have been found. Until February 2008 it was not necessary to register the Rule Executor with the PMS in order to post licences, though this has now been rectified.

Editor

There are a number of ongoing issues regarding the evolution of the Editor; the rectification of the .mp21 creation, the protection of the resources within .mp21, the time to play large files and increasing the basic functionality of the editor to allow rules to be included in the AXMEDIS objects. Much of this is still ongoing within the project at the time of writing and issues are being resolved (and new ones found) on a continuous basis, though of course these are converging.

So far the 4HOME activity has not been able to exploit the new behavioural editor functionality, though it provides the potential to separate out some of the AX4HOME recorder functionality to the player and may ultimately support a better partition of functions between the AX4HOME application and the AXMEDIS execution, giving a more securely robust system with less complex user set up.

The editor also contain a SMIL editing facility. Unfortunately this is not able to interpret all the SMIL functionality used in the AX4HOME demonstrator and so was only used in the initial stages of building the template. The capabilities of the SMIL player were not (at the time we checked) fully synchronized with the editor.

Another aspect of the SMIL player capabilities is the independence of the SMIL player from the AXMEDIS tools. This has meant that important developments within the AXOM, AXPLAYER etc, particularly in regard to the fast opening of large files or the synchronization of audio and video and playback, are not automatically reflected in the SMIL player. This is a disappointment as the 4HOME demonstrator uses the SMIL player extensively for its 'novel' approach and does not benefit automatically from the maintenance of the AXMEDIS tools by the partners. Problems with the SMIL player are currently being assessed by the project technical authority.

Player

As the player is intimately related to the editor for its technical implementation, the comments above apply directly. Of most concern to a home user only using the player is the time to unprotect content or to open large files. These issues are already noted by the project technical manager at the time of writing, as is the poor performance of the SMIL player compared to the raw AXMEDIS tools.

3 PKU - Description of Integration Prototype of domains and Documentation

3.1 Overview

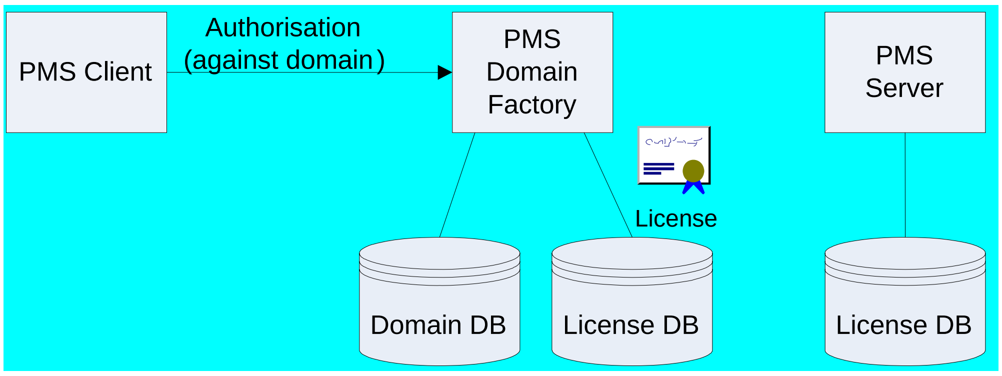
The contribution from PKU to the demonstration is to help establishing the domain-based management for usage of protected content for a group of people and/or devices in home or company environments. In order to make use of domains, a domain license must be created in the first instance which binds certain content to a particular domain. People who wish to enjoy the content must be registered as a member of the domain and use a device that must also be registered in the domain in advance. With the domain management, it is possible to use the domains as a way to group users and define a license for the whole group of users, which eases the management of licenses for multiple users to use multiple objects, as a single license is needed for the whole domain of users (plus the association of the user into the domain). For example, it is feasible to devise a subscription model where members of a family or a company can share some common content without having to purchase a license for each people.

The demonstration is set up to mimic a home environment, in which a laptop are used to form a home domain to share the protected broadcast contents created with AXMEDIS enhancements in conjunction with BBC demonstration. In the demonstration, the protected content created by recording broadcast content in BBC demonstration is associated with a domain license. The content is distributed using the AXMEIDS P2P tool, axeptool and shared by two laptop computers, which are already registered as members of the domain.

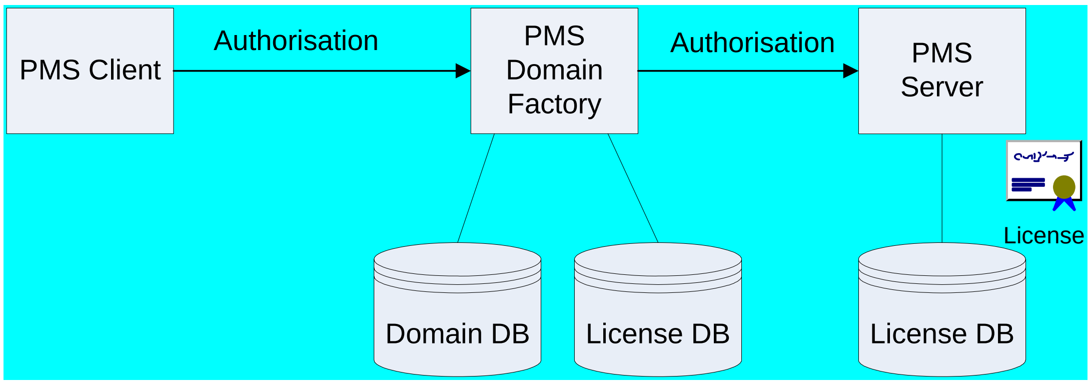
Note: PKU describes here the specification and suggestion to realise the domain management, whereas UPC implements the concepts in AXFW.

3.2 Component architecture

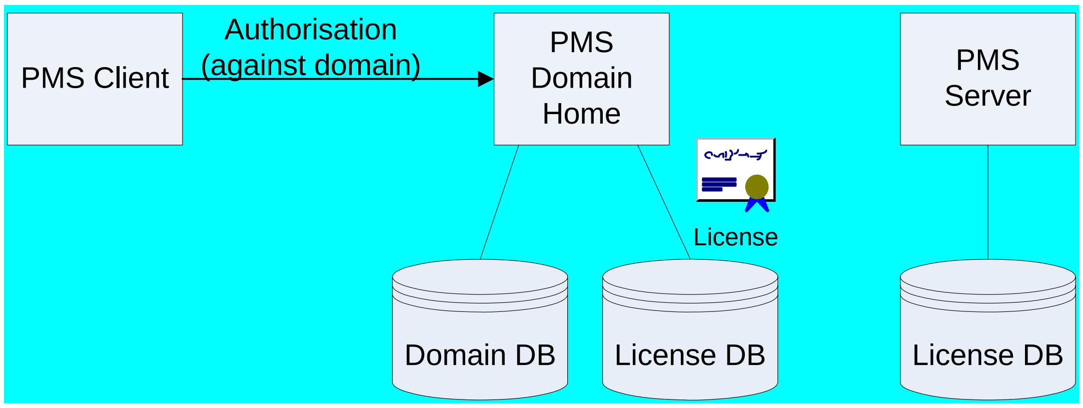
The preliminary components used to realise the domain management are PMS domain factory/home, PMS server and PMS client. The PMS domain factory and PMS domain home are respectively used for a company environment and home environment. The former is devised to manage more than one domains and the latter is devised to manage a single domain.
In the case of PMS Client connected to PMS Domain Factory, when the user requests authorisation of an action that is associated to the Domain, PMS Domain Factory will perform the authorisation using its local database. The generated action log (not shown in the figure) is sent to AXCS through PMS Server.



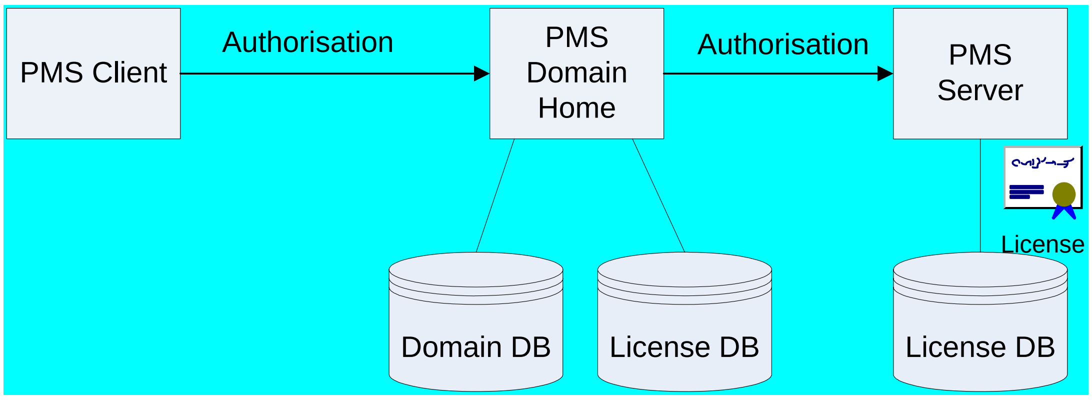
When the user requests authorisation of an action that is not associated with the Domain, PMS Domain Factory will bypass the authorisation to PMS Server. In this case the authorisation will be performed with the licenses present in PMS Server database.



In the case of PMS Client connected to PMS Domain Home, when the user requests authorisation of an action that is associated with the Domain, the Domain Home will perform the authorisation using the licenses present in its local database. The generated action log (not shown in the figure) is sent to AXCS through PMS Server.



When the user requests authorisation of an action that is not associated with the Domain, PMS Domain Home will bypass the authorisation to PMS Server. In this case the authorisation will be performed with the licenses present in PMS Server database.



3.3 Components in detail

The following components are applied to realise domain management in AXFW:

- PMS domain factory, the component for creation and registration of domains, users and devices in home or company environments, as well as for issuing domain license for a domain, this component is used for management of more than one group of domains;
- PMS domain home, the component for creation of a single domain in a home environment, and registration of users and devices in the domain, this component cannot be used to issue the domain license, however it stores a copy of the domain license generated by PMS server;
- Domainmanager, the command-line application as interface to add or delete a domain, and also the user interface to register and un-register users and devices in the domain;
- DUDmanager, a GUI application that interface with domain database, this component provides the same functionalities as the domainmanager;
- DRM editor/viewer, the component for user to request the creation of a domain license associated with a particular domain, the component enables the step by step creation of domain license;
- AXCP tools, the component used for creation of domain license in one step.

3.4 Integration and interaction with AXMEDIS components

All the components for domain management are implement in the AXFW.

3.5 Set up/compilation issues

- Update jspmsclient so that domain functionality can be used from AXCP tools (FHGIGD)
- Recompile AXCP tools to include the new functionality (DSI)

4 TI - Description of Integration Prototype of OMA Gateway and Documentation

4.1 Overview

The TI demonstration shows how a service provider can distribute premium content over fixed and mobile networks in an integrated way. The end user can buy certain Premium content on the service provider portal and then download and play the content (in different formats) on his mobile phone, his PC or both. The goal is to show that the same content (with different quality) may be used on the two clients seamlessly, enabling a business model where the customer can buy the content once, and use it independently of the delivery network (and client device/DRM technology). In this context the AXMEDIS platform plays the role of the unified back-office solution for the distribution of Premium Content over the internet and the mobile network. On the client side, PCs and mobile phones are used respectively: the PC runs the AXMEDIS player while the mobile phone is equipped with standard 3gpp/OMA technology.

The goal of the demo is to show that is easy:

- For content providers, to package and distribute content on the demo platform;
- For end user, to access and use the content.

In order to show a simple use case, the demo implements a “music shop” where the user subscribes to the service (e.g. paying a monthly fee) and receives in return unlimited access to the music library for one month. Access to content is granted month by month as long as the end user continues to pay a monthly fee (the payment is not required in the demo platform). This business model is sometimes called “all you can eat” or “subscription based” model.

4.2 Component architecture

In the prototype platform, the AXMEDIS Framework is used in connection with an OMA DRM v2 server provided by Telecom Italia for distribution of content to OMA DRM v2 mobile devices. Content adaptation takes place in the AXMEDIS Content Factory in order to better suit the characteristics of mobile phones. Also, automatic ingestion of content into the OMA DRM 2.0 server is performed by the AXCP tools

(Rule Editor, Rule Executor). In this configuration, the OMA DRM 2.0 Server (which is pre-existing know-how of TI), plays the role of an “OMA Gateway” between the AXMEDIS platform and the mobile devices.

On the Internet the following configuration is used for content purchase and usage:
- A PC Client equipped with the AXMEDIS Player and the AXOM
- AXMEDIS specific Content Format (based on MPEG-21)
- MPEG-21 licenses
- PMS acting as DRM server
- AXMEDIS certificates provided by AXCS

On the mobile network the configuration is the following:
- A Mobile phone equipped with a native multimedia player and an OMA DRM v2 Agent;
- Content format: OMA DCF v2 (based on ISO Base Media File Format)
- OMA DRM REL v2 licenses
- OMA DRM Rights Issuer provided by TI Multimedia as DRM server
- OMA and CMLA compliant certificates provided by TI Multimedia root CA

The demo platform is composed by several components. On the client side we have:
1. a PC Client equipped with the AXMEDIS Player and the AXOM;
2. a Mobile phone equipped with a native multimedia player and an OMA DRM v2 Agent.

While on the server side we have:
3. the AXMEDIS Editor– to author new content
4. The AXMEDIS Database – to store content and associated metadata.
5. the AXMEDIS Content Processing tools (AXCP) – to adapt content to different formats
6. the AXEPTool – to connect in peer-to-peer the AXMEDIS-4HOME Content Factory to all other AXMEDIS Factories (optional)
7. the OMA Gateway (OMA DRM v2 server) – translating simple AXMEDIS Objects into OMA DCF v2 objects and producing the corresponding licenses

4.3 Components in detail

All components used in the demo platform are standard AXMEDIS components (with no modifications), apart from two elements: the OMA Gateway (OMA DRM v2 server) and the mobile client device. Since the AXMEDIS components are described in their respective specifications, in the following only the latter two elements are described:

The OMA Gateway components are:
- Content Server (CS)
    - The web portal where Premium content can be obtained: this has been implemented as a simple PHP page which refers to content stored on the OMA DRM v2 server.
- OCSP: online certificate status protocol
    - The OCSP server provides certificate validation and DRM time synchronization;
- DB: relational database containing information and data pertaining to:
    - content and associated rights;
    - registered devices and domains (keys, certificates, etc.)
- DRM Server (Rights Issuer Server)
    - Implements the OMA DRM v2 specification;
    - executes the ROAP protocol;
    - deliver Rights Objects (licenses) to OMA clients;
- SMSC: optional network component sending messages via wap-push from the DRM server to the mobile device (not used in the demo)

The mobile client devices used in the demo are Nokia N91 commercial phones equipped with TI own certificates. The N91 is an OMA DRM v2 compliant device. The technical specification of the N91 is available at http://forum.nokia.com/devices/N91

4.4 Deployment architecture

The demo platform is composed mainly by one server (http://ax4home.axmedis.org) which hosts the AXMEDIS Content Factory (AXDB, AXCP, AXEPTool) and the OMA Gateway (CS, DB, DRM Server) with the exception of the "OCSP responder" and "SMSC" components.

The OCSP responder is hosted on another TI server which is reachable online at the following address: https://atlantis.tilab.com/

The SMSC is not used in the demo (it is only needed for compatibility with OMA 1.0).

The ax4home server is located in Turin and is available online since April 2007. This server is maintained operational at full time.

- Hostname: ax4home.axmedis.org
- IP address: 82.90.96.218
- PC model: HP xw4100 2.7 GHz dual Pentium

The atlantis server is located in Turin as well and is running an OCSP responder service since 2006. Also this server is maintained operational at full time.

- hostname: atlantis.tilab.com
- IP address: 163.162.91.85
- PC model: HP PROLIANT DL 380 G4

4.5 Integration and interaction with AXMEDIS components

The integration between the AXMEDIS environment and the OMA Gateway is performed by a script executed by the AXCP Rule Executor. The script (which is itself named "OMAgateway.axr") is fired at fixed time intervals (e.g. 1 hour, or 1 day) and executes the following steps:

- selects from the ax4home DB all AXMEDIS Objects which have a Dublin Core element named "subject" whose value is " music";
- checks that the selected Objects contain the following Dublin Core elements: title, creator, description (these fields are needed to publish the Object on the Content Server);
- checks if the Object has been already exported; in this case, stops processing the Object;
- converts the audio resource contained in the Object into a 64kbit, mono, compressed AAC format;
- produces a clone Object containing the compressed version of the audio resource;
- gives the cloned Object as input to the OMA DRM server (using the httpconnection class);
- the OMA DRM Server converts the Object to OMA DCF v2, produces a time-limited license expiring in one month (to support the subscription model), and imports the metadata in the OMA DRM server DB.
- Once in the DB, the new content is immediately visible on the Content Server, and can be downloaded from a mobile phone.

4.6 Set up issues

AX4Home Factory installation and configuration:

1. The FTP server running on AX4home supports passive mode in a limited way (only a few ports are available). Therefore if used by multiple users, it may refuse to connect.

2. (*solved in v. 1.2*) When running the SQL script that creates the AXMEDIS DB on MySQL 4.1.22 or MySQL 5.0.41, I get the following *mysql Error: 1071 (Specified key was too long; max key length*

is 1000 bytes) which is explained here. To solve the problem I used the workaround suggested in the mysql bug documentation.

3. Saving PAR information on the AX4HOME server from the AXMEDIS Editor is currently disabled. If you do need this functionality please contact me.

Open issues regarding the AXMEDIS tools:

1. (*solved in v. 1.2*) The Editor certification does'n work from behind a corporate firewall. When performing the initial certification the Editor tries to connect to the PMS server using the SSL protocol on port 8502, which is not the default SSL port (443). Most firewalls block the SSL exchange when it is executed on a port different from 443. I suppose the same applies to all other tools that need to be certified.

2. (*solved*) Protecting a resource using the AXMEDIS Editor using AES with a 128 bit key results in an error ("key is too short").
   - **Solution**: *the key length must be indicated in BYTES, not BITS!*

3. (blocking) access to the REL translation functions from JavaScript is disabled;

4. (*solved in v. 1.2*) the aac audio format is not supported (in any tool - the Editor, the Rule Editor, the Player)

5. (*solved in v. 1.2*) the mime-type currently assigned to the mpeg-1 audio layer III (i.e. mp3) format is audio/mpeg. Instead, the AudioAdaptation.FFAudioTranscoding call of the Rule Editor requires such resources to be tagged as audio/x-mpeg (experimental). This was the old mime assignment to the mp3 format and is now obsolete.

6. (new feature request) the audio fingerprinting extraction function is defined as follows: STRING AxAFPExtract (RESOURCE InputResource, RESOURCE OutputResource, RESOURCE OutputResource2, INT32 nFeatures, INT32 frameSize, INT32 frameShift, INT32 offset);
   The last parameter indicates the offset for the fingerprint calculation, however there is no way to limit the length of the fingerprint calculation (the calculation is always performed up to the end of file). Therefore we suggest to modify the call by adding another parameter named "length" which indicates the number of audio samples over which the calculation is performed. The resulting signature would then be:
   STRING AxAFPExtract (RESOURCE InputResource, RESOURCE OutputResource, RESOURCE OutputResource2, INT32 nFeatures, INT32 frameSize, INT32 frameShift, INT32 offset, INT32 length);

7. (blocking) using the Rule Editor to extract a fingerprint from large files (wmv having size 300-400 MB), we get the following errors: "not enough free memory in system! ran out of memory!" or " bad allocation error".

8. (annoying) the user registration procedure doesn't check for duplicated nicknames inserted by end users in the registration web page. Instead, the procedure accepts the data, then sends an email to the end user containing an hyperlink to the registration portal; however when the user clicks on the link, if the nickname was a duplicated one, the result is a misleading error message: "server error, please try again later". Trying multiple times always results in the same error message being displayed.

  - o **Solution**: the user registration procedure should check for duplicated nicknames in advance, and inform the end user immediately (possibly proposing a new nickname).

9. (annoying) the user registration procedure does not allow the end user to set its own password. A random password is generated, which is very difficult to remember.

10. (annoying) the user registration procedure sends a total of three email messages to the end user to complete, when one would be sufficient.

11. (annoying) the user registration procedure requires all fields to be filled (even the fax number! not quite common for end users).

12. (annoying) there is no way for the end user to unregister himself from the DB, or to change his/her own data (preferred nickname, email address, etc.)

4.7 General feedback

Here is a list of suggestions to AXMEDIS developers in order to improve the usability and functionality of the AXMEDIS Tools. It is provided just as an informative feedback to the project, as the result of our experience using the AXMEDIS Tools to build some real-case scenarios.

1. **Tool configuration**: currently each tool must be individually configured, and different tools expose the configuration editor under a different menu item: in the AXMEDIS Editor it is under file–>configuration–>advanced, in the AXCP Rule Editor it is under View–>Preferences, and so on. It would be simpler if an installation wizard performed the configuration once and for all. Alternatively, a “configuration plugin” could manage the configuration for all tools in the same way.

2. **Firewall support:** currently the AXMEDIS tools (Editor, Rule Editor, etc.) communicate via Web Services with the Content Factory on TCP port 8080. Although there is a “proxy setting” paramenter in the tool configuration, a direct communication is currently required to perform any operation involving such Web Services, e.g. to save or load an Axmedis Object to/from a Content Factory.

3. **Input parameter check**: the “Edit Protection” functionality of the AXMEDIS Editor requires input parameters from the end user at a detailed level (algorithm, key, key length) but does not inform the user about the format of such parameters, therefore the user has to “guess” the correct ones. It would be more convenient for the end user to be informed about the appropriate input parameter when he/she is required to provide one, and then check that the value actually provided is correct.

5 ETRI - Description and documentation for Tool server and interface

5.1 Overview

The ETRI demonstration shows not only how to register, modify and delete tools but also how a client can download tool from the DRM Protection Tool Server. The DRM Protection Tool Server basically provides functionalities of registering new tools, modifying existing tools and their parameter and deleting unnecessary tools. In addition, it provides a function to transmit any tool requested by the AXMEDIS Client.

The goal of the demonstration is to show following functionalities:

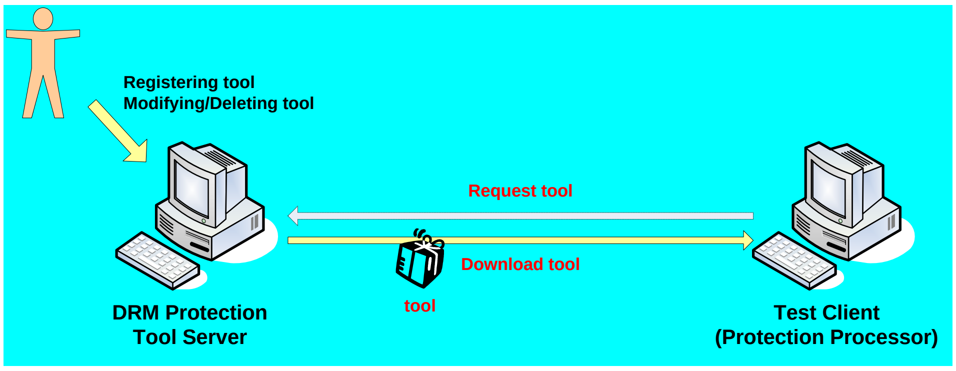
- Registering, modifying and deleting DRM tool
- Downloading DRM tool

To show a simple use case, the demonstration involves a server manager who operates the DRM Protection Tool Server can register/modify/delete tools and an end-user who would like to playback a protected content at the Test Client can request and download any tool to decrypt it.

5.2 Component architecture

The demonstration consists of following component:

- The DRM Protection Tool Server
- The Test Client
    - In order to show a tool downloading this Test Client is provided, which is an internal module of the AXMEDIS Client to communicate with the DRM Protection Tool Server. Note that it is named the Protection Processor in 3.5.2.2 chapter of the 12.1.3.1 specification is provided.

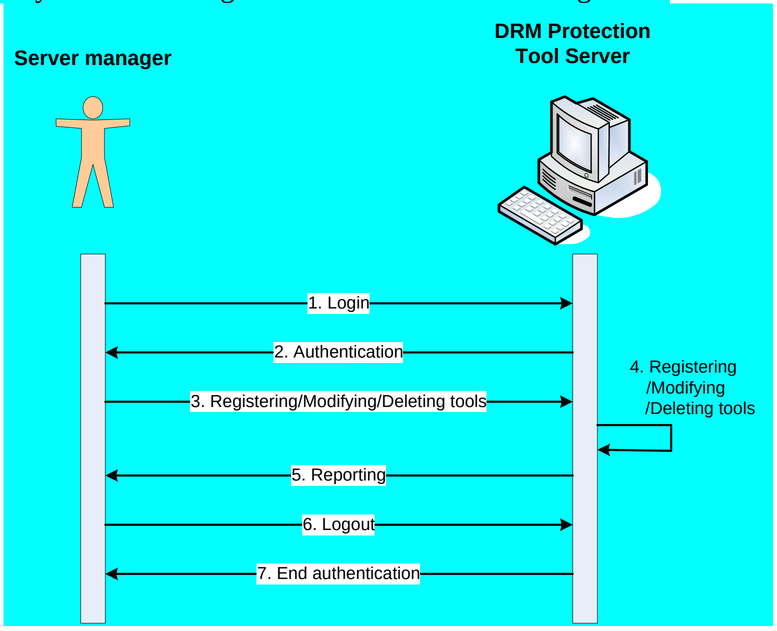


Above figure shows a concept of the relation between the DRM Protection Tool Server and the Test Client.

5.3 Components in detail

5.3.1 Instruction flow

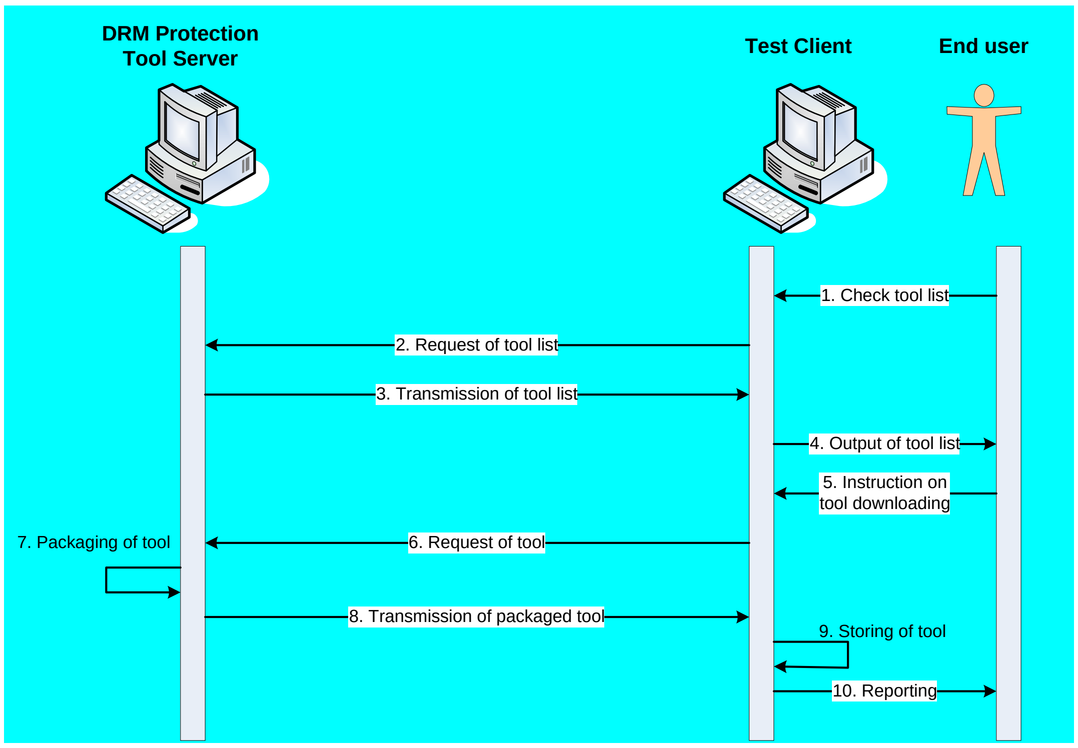
The first use case for registering, modifying and deleting the DRM tool in the DRM Protection Tool Server is shown in the following figure. This sequence diagram presents procedural operation of the DRM Protection Tool Server by a server manager's instruction for tool management.


Server manager
DRM Protection Tool Server
1. Login
2. Authentication
3. Registering/Modifying/Deleting tools
4. Registering /Modifying /Deleting tools
5. Reporting
6. Logout
7. End authentication

The sequence is described below.

1. A server manager login on the DRM Protection Tool Server using identifier and password.
2. The DRM Protection Tool Server confirms that a server manager has an authentication to operate it.
3. A server manager instructs registering, modifying or deleting tools using user's interface supported by the DRM Protection Tool Server.
4. The DRM Protection Tool Server registers, modifies or deletes tools by an instruction of a server manager.
5. And the DRM Protection Tool Server reports its all operations including registering, modifying and deleting tools.
6. A server manager logout.
7. The DRM Protection Tool Server ends an authentication on operating, and keeps on stand-by status until any manager's login.

The second use case for downloading DRM tool at end user's side is shown in the following figure. This sequence diagram shows how a DRM tool is transmitted from the DRM Protection Tool Server to the Test Client. All instructions between two components are the XML based messages and their physical interface is Ethernet using TCP/IP protocol. The messages are defined in 5.3.4 section.



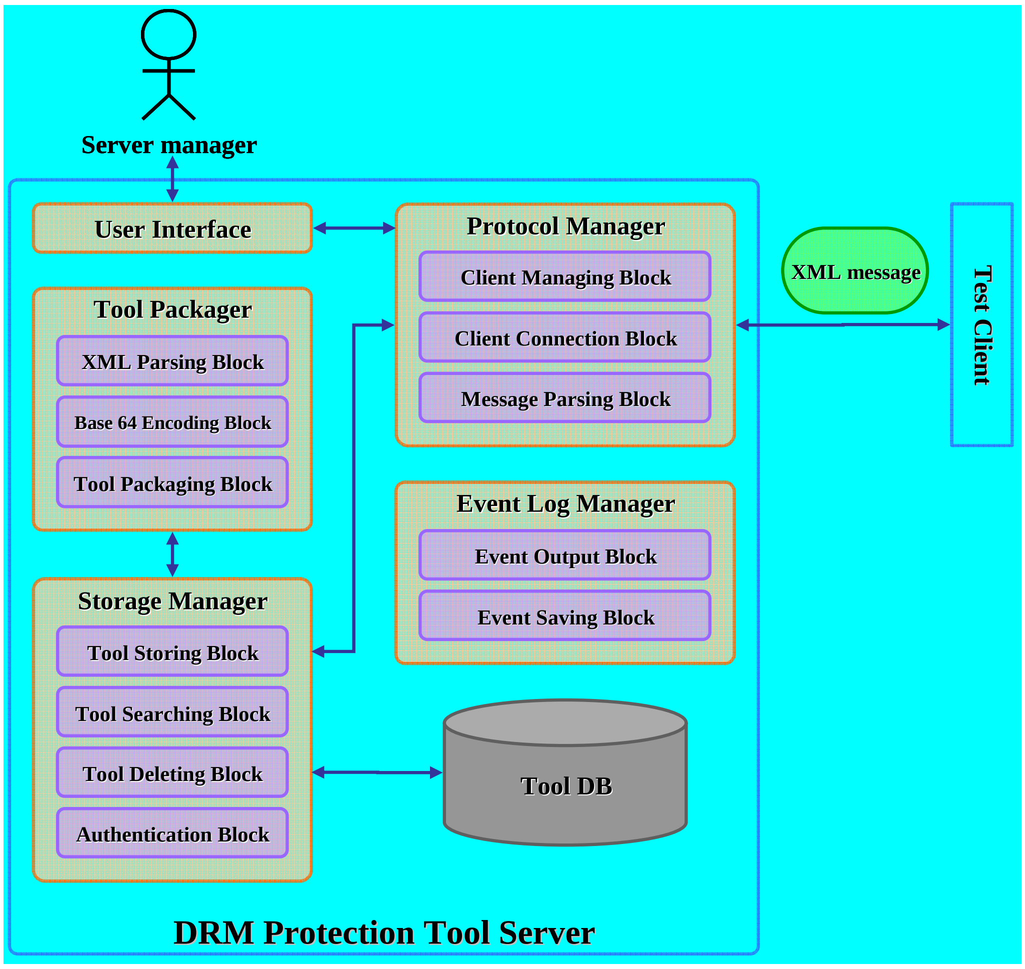
The working sequence is described below.

1. An end user checks a tool list of the DRM Protection Tool Server.
2. The Test Client requests a tool list to the DRM Protection Tool Server by an end user's checking.
3. The DRM Protection Tool Server transmits a tool list to the Test Client.

4. The Test Client shows a tool list to an end user.
5. An end user instructs on downloading of any tool which is required to decrypt content.
6. The Test Client requests tool.
7. The DRM Protection Tool Server packages a requested tool as a form of the XML.
8. And the DRM Protection Tool Server transmits the packaged tool to the Test Client.
9. The Test Client stores the tool.
10. And the Test Client reports a completion of downloading process to an end user.

5.3.2 Functional structure

The DRM Protection Tool Server's functional structure is shown in following figure. It consists of Tool DB, Storage Manager, Tool Packager, Protocol Manager and Event Log Manager. All modules have two or more functional blocks which support their modules respectively, and all blocks have implemented as a function in C++ language. Especially Event Log Manager, did not described in 12.1.3.1 specification, is included in the DRM Protection Tool Server so that provides functionality of event reporting for a server manager.



The main role of each module is described below.

1. Tool DB: is the storage module for tools and their additional information. The additional information includes tool ID, metadata, binary form of tool, etc. This additional information as a form of the XML is architected as shown following table.

	Data type	Not NULL	Auto Inc	Flags	Default value
index	INTEGER	√	√	UNSIGNED	NULL
type	VARCHAR(10)	√			
ToolID	VARCHAR(100)	√			
ToolName	VARCHAR(100)	√			
ToolDescription	VARCHAR(1000)	√			
ToolFileName	VARCHAR(100)	√			NULL
ToolFileSize	INTEGER	√		UNSIGNED	NULL
ToolFileBinary	LONGTEXT	√			

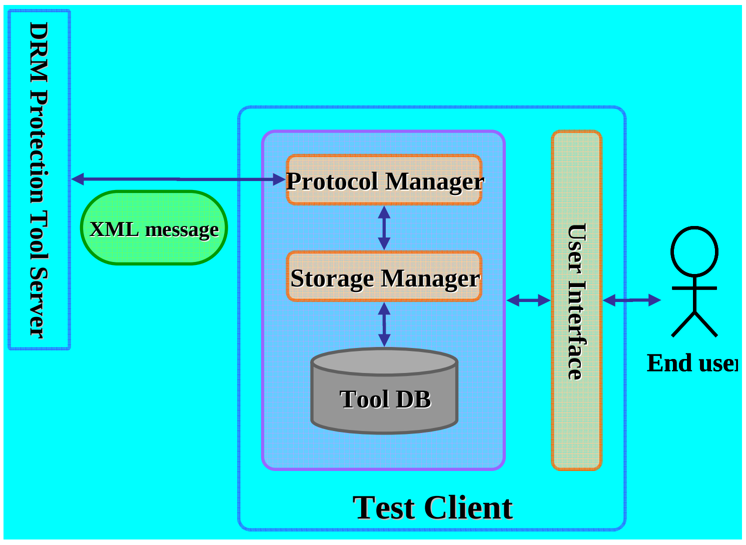
2. Storage Manager: is the module for management of Tool DB as a middleware with Tool Packager. The Storage Manager's management includes authentication of a server manager as well as storing, searching and deleting DRM tools in the Tool DB

3. Tool Packager: is the module for generation of tool message using tool related information which is stored from the Tool DB. The tool message includes a coded DRM tool using BASE64 encoding algorithm.

4. Protocol Manager: is the module for management of the outer client. The Message Parsing Block parses all messages received from clients, and by means of this parsed information, the Protocol Manager instructs all operations to modules in the DRM Protection Tool Server.

5. Event Log Manager: is the module for storing of all events generated in the DRM Protection Server. The events enable a server manager to comprehend all status of the DRM Protection Server.

The Test Client's functional structure is shown in following figure. It consists of Tool DB, Storage Manager and Protocol Manager. Functionalities of three modules are same with theirs in the DRM Protection Tool Server.



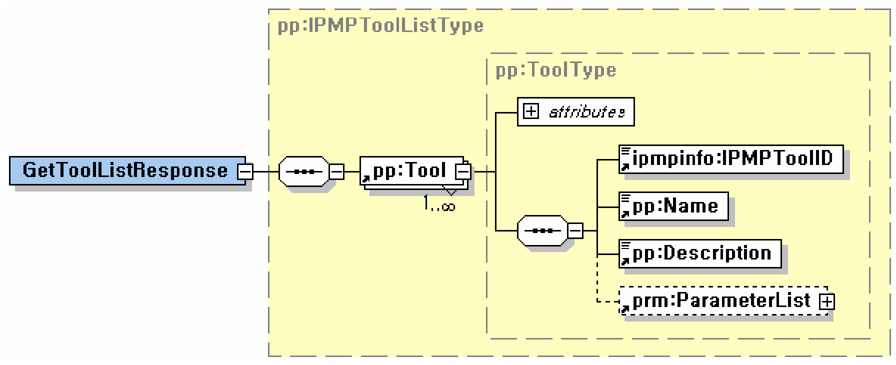
5.3.3 Communication messages

The DRM Protection Tool Server and the Test Client exchange the XML messages to request and response DRM tools. The XML messages are composed of GetToolList, GetToolListResponse, GetTools and GetToolsResponse. The followings show the structure and the description of the XML messages.

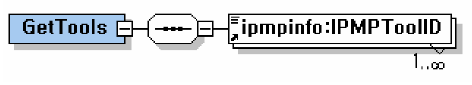
1. GetToolList

Diagram	
Description	This message is sent by the Test Client to the DRM Protection Tool Server to request tool list. Receiving this message, The DRM Protection Tool Server sends response back to the Test Client with GetToolListResponse message.

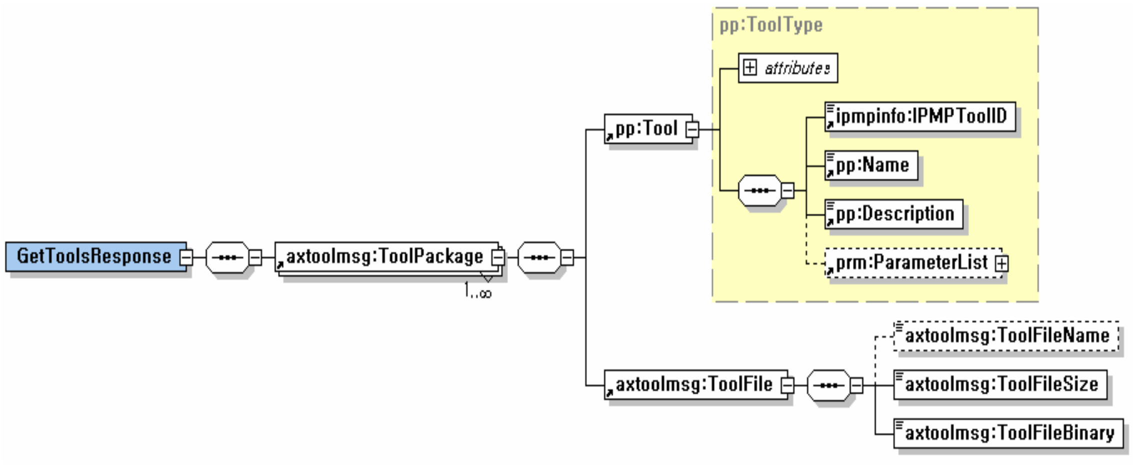
2. GetToolListResponse

Diagram	
Description	This message is sent by the DRM Protection Tool Server to the Test Client to response. This message conveys the list of tools stored in the DRM Protection Tool Server. The Tool ID of the schema is compliant with MPEG-21 DII (Digital Item Identification). Receiving this message, the Test Client can select any tool to use for decryption and encryption of contents.

3. GetTools

Diagram	
Description	This message is sent by the Test Client to the DRM Protection Tool Server to request necessary tools. This message conveys the list of the requested tools. Receiving this message, the DRM Protection Tool Server sends response back to the Test Client with GetToolsResponse message.

4. GetToolsResponse

Diagram	 pp:ToolType attributes pp:Tool ipmpinfo:IPMPToolID pp:Name pp:Description prm:ParameterList GetToolsResponse axtoolmsg:ToolPackage 1..∞ axtoolmsg:ToolFileName axtoolmsg:ToolFile axtoolmsg:ToolFileSize axtoolmsg:ToolFileBinary
Description	This message is for embodiment of raw DRM tools into structured and self-explained DRM tools. When the DRM Protection Tool Server receives the tool request messages from the Test Client, the DRM Protection Tool Server create a Tool package with Tool XML Profiles and Tool binaries for transmission.

5.4 Integration and interaction with AXMEDIS components

The DRM Protection Tool Server and the Test Client is demonstrated without any AXMEDIS components. However, as the Test Client shall be embedded into AXOM of the AXMEDIS Client, it needs for the Protocol Manager of the Test Client to interface with PMS Client of the AXOM. Therefore the Test Client depends on the following AXMEDIS components:

- AXOM
- PMS Client

5.5 Set up/compilation issues

In order to test, the following tools need to be installed.

- MySQL Essential 5.0.45
  - http://dev.mysql.com/get/Downloads/MySQL-5.0/mysql-essential-5.0.45-win32.msi/from/pick
- MySQL GUI Tools
  - http://dev.mysql.com/get/Downloads/Connector-ODBC/3.51/mysql-connector-odbc-3.51.21-win32.msi/from/pick
- MySQL Connector/ODBBC 3.51.20

- o http://dev.mysql.com/get/Downloads/MySQLGUITools/mysql-gui-tools-5.0-r12-win32.msi/from/pick

5.6 General feedback

To operate in the AXMEDIS framework, the DRM Protection Tool Server should be interoperated with the AXMEDIS Client. For this, it is need for the DRM Protection Tool Server to test an interface.

6 sDae - Documentation of the API and Basis of Formulation

6.1 Overview

This section describes the AxIPOntology (Axmedis Intellectual Property Ontology) and related software.
The ontology is a repository of knowledge, where an Intellectual Property model is made explicit in the form of an W3C standard OWL Ontology.
Key operations realized over Intellectual Property can be confirmed to be Axmedis-model conformant when an Ontology server is queried. This server is called the Axmedis IP Ontology Server.
The standard way of accessing the Ontology would be querying this AxIPOntology Server through WebServices, but direct queries can be made through a standard API. The Server itself is based on the API.
This section of the document will describe the Ontology, the API, the Server, a monitor Tool and a Command Line interface.

6.2 Component Architecture

The functional archictecture is shown in the next figure.

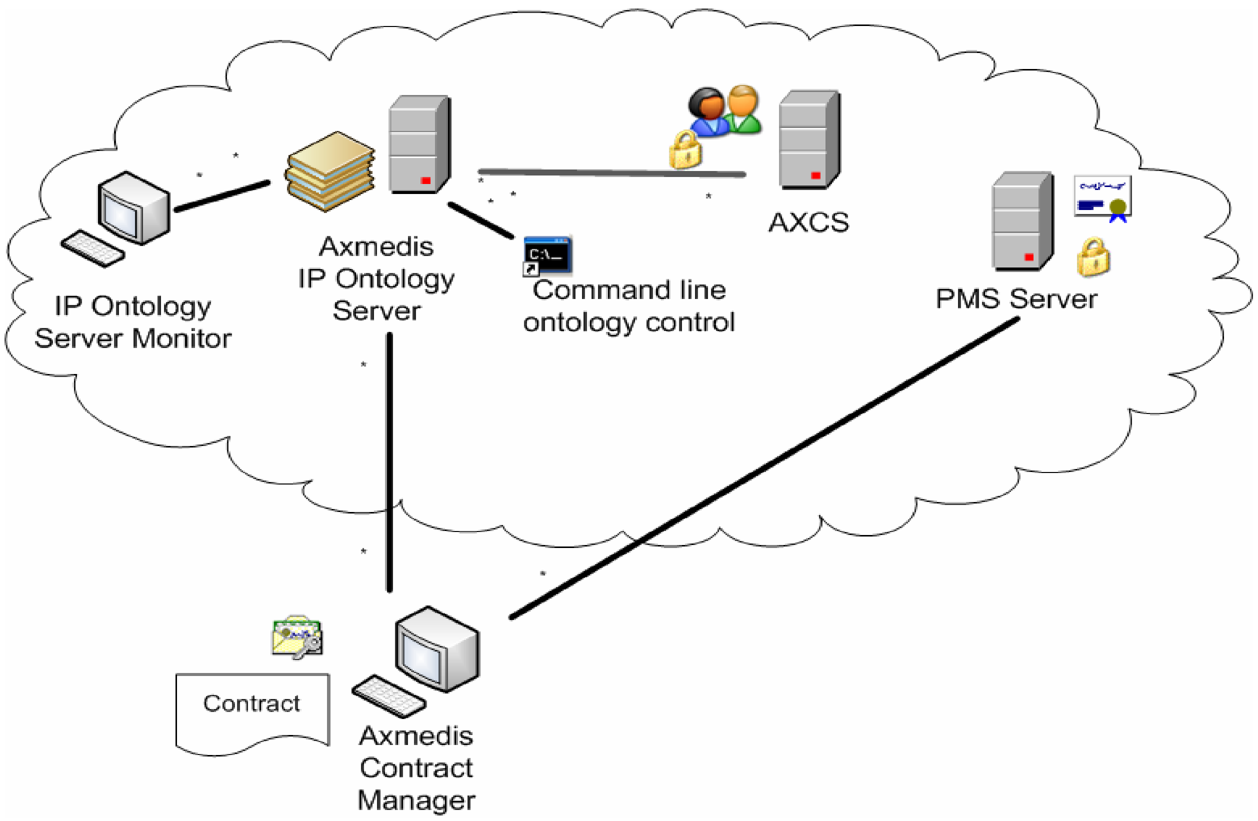

IP Ontology Server Monitor
Axmedis IP Ontology Server
Command line ontology control
AXCS
PMS Server
Contract
Axmedis Contract Manager

Figure 5. Architectura of the Ontology components

The stack of libraries can be seen in the next figure:

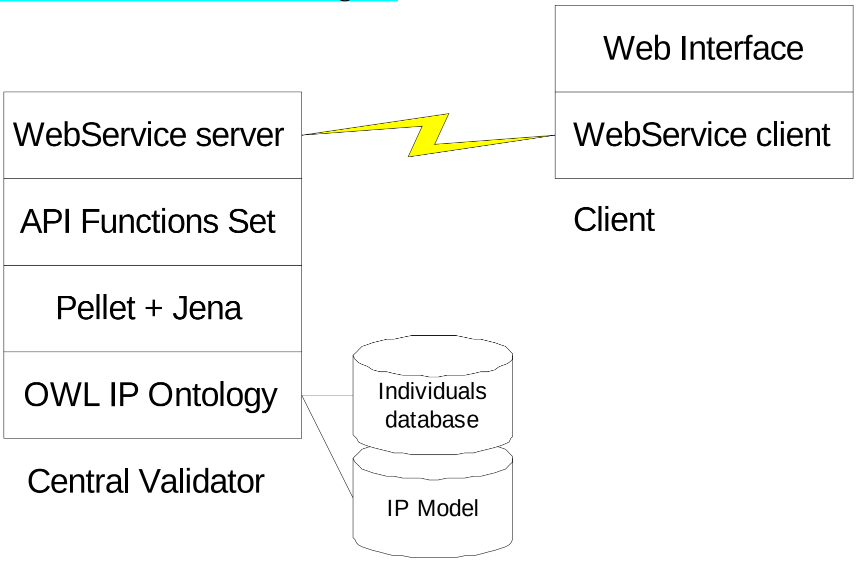


Figure 6. Libraries stack

- The Ontology is an OWL file, that is to say, a XML file accesible by everybody.
- Jena and Pellet are two Java libraries that provide an easier access to ontology functions, the offer an API which can be viewed in: http://jena.sourceforge.net/documentation.html, and http://pellet.owldl.com/docs/ respectively.
- A Java API is provided on top of Jena and Pellet, for dealing with the particular AxIPOntology. This API offers an easy access from Java applications to the main functionalities to be done on the ontology.
- WebServices are given on top of the aformentiioned, both to further simplify access as well as to make it machine-independent.

Any client will be able to connect on top of the web service client.

6.3 Components in Detail

6.3.1 The Ontology

The Ontology in an OWL file, called AxIPOntology and counting 25,6kB in its present version (4.0). Each ontology must have a unique URL, in this case, the given URL is http://axmedis.org/AxIPOntology.owl. Note that a PURL could be given upon request, in order to keep a permanent reference if axmedis changes its domain name.

The ontology is available in the Axmedis repository here:

https://cvs.axmedis.org/newrepos/trunk /Framework/doc/specification/axipontology/axipontology.owl

6.3.2 The Axmedis IP Ontology API

The API interface is given in the repository under the following path:

https://cvs.axmedis.org/newrepos/trunk /Framework\source\axipontology

Three files define the API, and this can be described through the following figure:

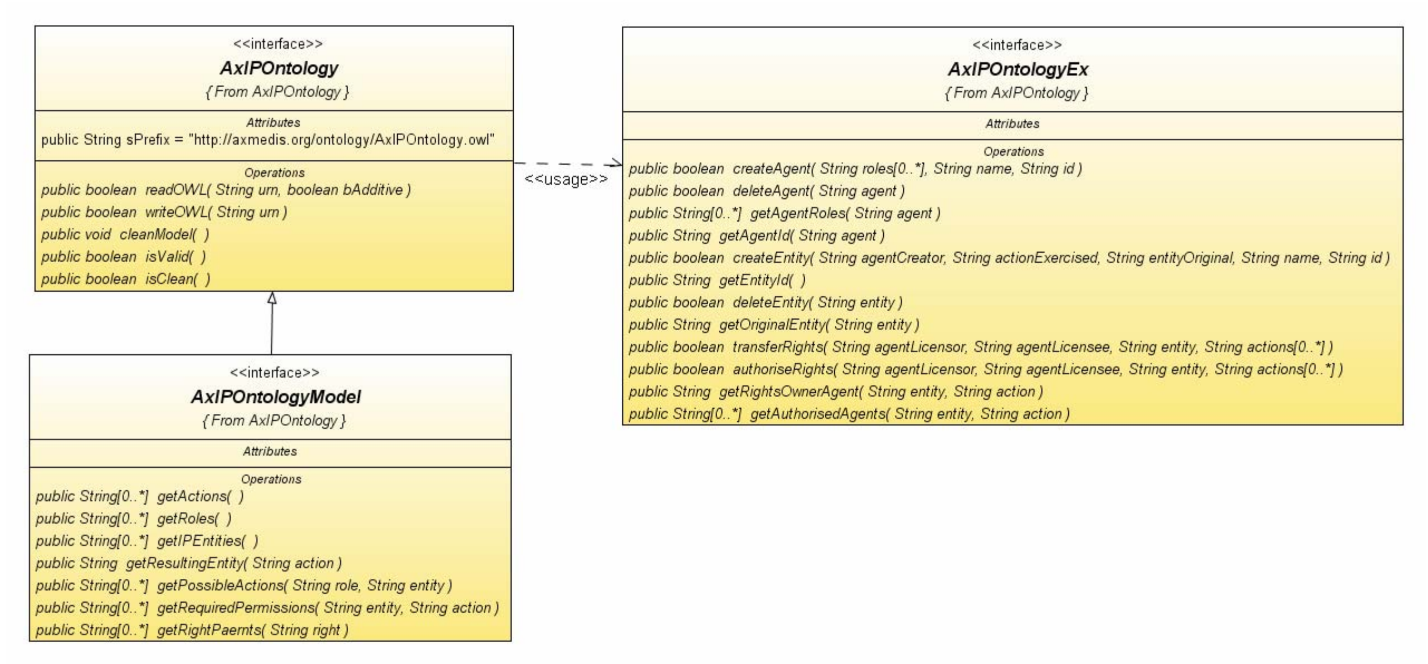

<<interface>>
AxIPOntology
{ From AxIPOntology }
Attributes
public String sPrefix = "http://axmedis.org/ontology/AxIPOntology.owl"
Operations
public boolean readOWL(String urn, boolean bAdditive)
public boolean writeOWL(String urn)
public void cleanModel( )
public boolean isValid( )
public boolean isClean( )
<<usage>>
<<interface>>
AxIPOntologyModel
{ From AxIPOntology }
Attributes
Operations
public String[0..*] getActions( )
public String[0..*] getRoles( )
public String[0..*] getIPEntities( )
public String getResultingEntity(String action)
public String[0..*] getPossibleActions(String role, String entity)
public String[0..*] getRequiredPermissions(String entity, String action)
public String[0..*] getRightPaernts(String right)
<<interface>>
AxIPOntologyEx
{ From AxIPOntology }
Attributes
Operations
public boolean createAgent(String roles[0..*], String name, String id)
public boolean deleteAgent(String agent)
public String[0..*] getAgentRoles(String agent)
public String getAgentId(String agent)
public boolean createEntity(String agentCreator, String actionExercised, String entityOriginal, String name, String id)
public String getEntityId( )
public boolean deleteEntity(String entity)
public String getOriginalEntity(String entity)
public boolean transferRights(String agentLicensor, String agentLicensee, String entity, String actions[0..*])
public boolean authoriseRights(String agentLicensor, String agentLicensee, String entity, String actions[0..*])
public String getRightsOwnerAgent(String entity, String action)
public String[0..*] getAuthorisedAgents(String entity, String action)

Figure 7. Axmedis IPOntology API

The Java API consist of the following classes:

- **AxIPOntology**. It defines the I/O methods for accessing the ontology. Also, it can validate the ontology.
- **AxIPOntologyModel**. Derived from the former, it provides access to the structural features of the ontology, i.e., the function calls are invariant for the same version of the ontology. They describe the ontology statically (i.e., which actions can be executed by which roles)
- **AxIPOntologyEx**. Uses AXIPOntology, and permits a dynamic use of the ontology. Individuals of the classes can be created, and dynamic operations can be performed, like giving authorisations or validating operations.

The implementation is available in the repository under the following URL:

https://cvs.axmedis.org/newrepos/trunk /WebServices\AXIPOntologyWS\src\java\AxIPOntologyAPI

The implementation consist of this five classes:

- AxIPOntologyBasic.java
- AxIPOntologyBasicEx.java
- AxIPOntologyManager.java
- AxIPOntologyModel.java
- AxIPOntologyUtil.java

6.3.3 The AxOntologyConsole control application

The source code is available under:

https://cvs.axmedis.org/newrepos/trunk /WebServices\AXIPOntologyWS\src\java\AxOntologyConsole

This is a console application allowing basic consults and changes in the Axmedis IP Ontology. In particular, it allows:

- Loading / Storing data in an ontology
- Listing the users and their roles

- Listing the IP Entities
- Listing the actions for a user over a role
- Creating users and giving them the roles
- Deleteing the users
- Creating named IP Entities

To access these functionalities, the command lines parameters are given as follows:

```
            Usage: java -jar AxOntology [-h] [-lu] [-cu Name Role] [-owl OWL_URL]");
---------------------
                -owl OWL_URL          Specifies the URL address of the OWL
                -h                              Shows help
                -lu                             List users
                -lip                            List IPEntities
                -lao A O                        Lists the actions available for an Agent A over an Object O
                -s                              Stores the changes made on the ontology
                -cu    Name Role          Creates a user named Name with role Role
                -du    Name                     Deletes a user named Name
                -dip Name                       Deletes an IPEntity named Name
                -cip A O N Or             Creates an IP named N of kind O based on Or belonging to A
```

6.3.4 The Axmedis IP Ontology Server

It is a web services server, running on Apache Tomcat 5.5 and making use of the *axis* library.
It does not require any special library apart from those of the ontology, and has been tested with Sun Application Server and axis+Tomcat 5.5. Several clients have been tested too, using either of axis or gsoap.
The WSDL is available in the Axmedis repository here:

https://cvs.axmedis.org/newrepos/trunk /WebServices\AXIPOntologyWS\doc/specification/AxIPOntologyWS.wsdl

The web services are offered in the following url:

http://*ipserveraddress*:8080/axis/services/AxIPOntology

The following public services are defined:

- `public String testOntologyServiceWS(String dummy) throws RemoteException`
  Tests the presence of the Ontology Service. A dummy string is passed to make web services compatible with gsoap.

- `public String[] getAgentRolesWS(String agent) throws RemoteException`
  Gets the roles of a given agent

- `public String transferRightsWS(String agentLicensor, String agentLicensee, String entity, String[] actions) throws RemoteException`
  Transfers a certain set of rights over one entity from agent Licensor to agent Licensee

- `public String getRightsOwnerAgentWS(String entity, String action) throws RemoteException`
  Gets the right owner of a entity for doing certain action

- `public String[] getActionsPerAgentAndObjectWS(String agent, String object) throws RemoteException`
  Gets the available actions per agent and object

6.3.5 The IP Ontology Server Monitor

The IP Ontology Server Monitor is an application to visually monitor the activity of the ontology.
It is a multiplatform remote C++ application, and communicates with the OntologyServer through Sockets. This makes the monitor aplication portable and able to be executed remotely in any connected computer.

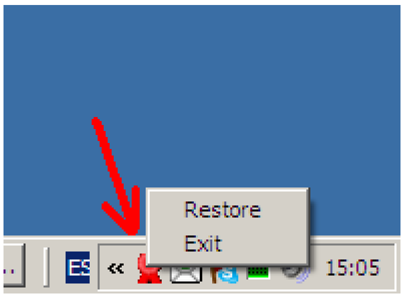
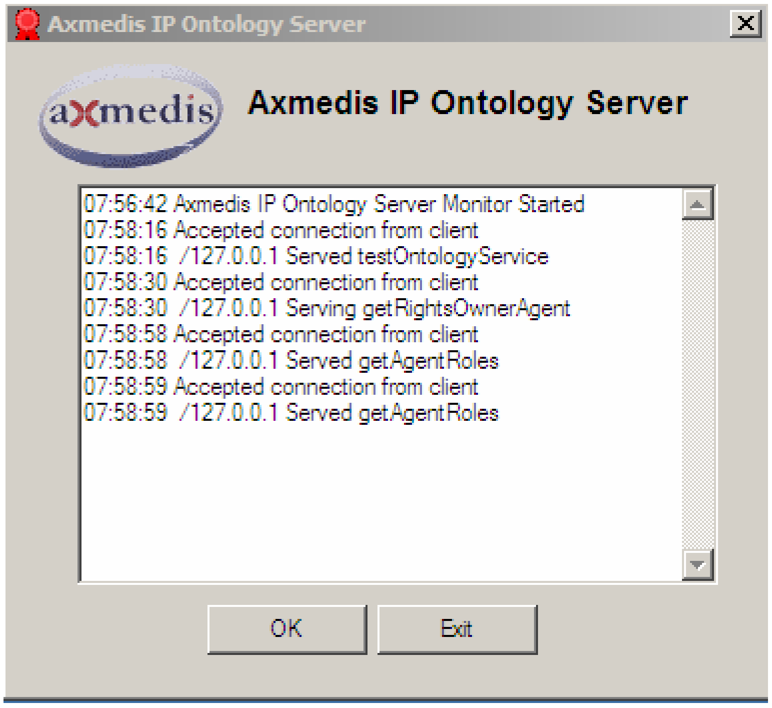


Figure 8. The Monitor application minimized in the Windows tray

It is can be minimised and maximised upon user request. Additionally, it leaves a clear events log



```
07:56:42 Axmedis IP Ontology Server Monitor Started
07:58:16 Accepted connection from client
07:58:16  /127.0.0.1 Served testOntologyService
07:58:30 Accepted connection from client
07:58:30  /127.0.0.1 Serving getRightsOwnerAgent
07:58:58 Accepted connection from client
07:58:58  /127.0.0.1 Served getAgentRoles
07:58:59 Accepted connection from client
07:58:59  /127.0.0.1 Served getAgentRoles
```

Figure 9. Appearence of the Axmedis IP Ontology Server Monitor

6.4 Integration and interaction with AXMEDIS components

The application is demonstrated through the ContractManager application.
This is a rather complex C++ application which depends on the following Axmedis Framework components:

- ContractGen
- AuthorisationSupport
- EncDecSup
- KeyGen
- LicenseManager

- LicenseModel
- Axom
- PluginManager
- PMS Client
- ProtectionInfoManager
- RDDServer
- SecureCache
- Common

6.5 Set up/compilation issues

To test all functionalities, the following tools need to be installed:

In the server side

- The OWL with the ontology
- The WebService server, a web application prepared to be deployed in an Apache Tomcat Web Server. We shall call this tool Semantic Server, all together with the ontology.
- A Monitor Tool, able to view the ontology behaviour (that is to say, on the server)
- A Command Line Control tool, to modify the individuals in the ontology.

In the client side

- The contract manager tool, enabled to interoperate with the ontology.

To install the WebService server, the following prerequisites are needed:

- An Apache Tomcat web server (tested on version 5.5)
- No restrictions for the ports. Although the webservice is served through port 8080, which very seldom is blocked, the Semantic Server and the monitor tool interact via TCP/IP, through non-standard ports. This provides the advantageous feature of allowing the monitor tool to be run in any machine with an internet connection. Protection is given because the monitor tool has to identify itself with the ontology, that is, no arbitrary users or hackers can have access to monitor the ontology.

6.6 General feedback

To boost the utility of the ontology, it should be able to interoperate with Axmedis Users´s databases and Axmedis Content Databases.

This is not a trivial endeavour which could be addressed in a further extension of the application but that undoubtedly will involve modifications in several other AXEMDIS parts. .