



Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE3.1.2I

Framework and Tools Specifications (Distribution and Portal)

Version: 3.1

Date: 17/03/2005

Responsible: DSI

Project Number: IST-2-511299

Project Title: AXMEDIS

Deliverable Type: report

Visible to User Groups: NO

Visible to Affiliated: NO

Visible to the Public: NO.

Deliverable Number: DE3.1.2 Part I

Contractual Date of Delivery: January 2005

Actual Date of Delivery: 17 March 2005

Title of Deliverable: Document

Work-Package contributing to the Deliverable: WP3.1

Task contributing to the Deliverable: WP3, WP2

Nature of the Deliverable: report

Author(s): DSI, EUTELSAT, CRS4, SEJER, ILABS,

Abstract: This document report the early specification of the distribution channels demonstrators integrated in the AXMEDIS architecture. The channels considered are towards: I-TV, PC, PDA, KIOSKS, AXPETools, cellular phones, tablet PC, etc.

Keyword List: distribution channel, protection, packaging, integration demonstration..

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 Content

1	EXECUTIVE SUMMARY AND REPORT SCOPE	7
2	DISTRIBUTION VIA INTERNET TOWARDS PC, TISCALI CHANNEL (WP4.6, WP9.4: TISCALI)	8
2.1	TISCALI DISTRIBUTION CHANNEL (TISCALI)	8
2.2	MEDIA CLUB BACK-OFFICE.....	9
2.3	MEDIA CLUB FINAL USER	10
2.4	TISCALI CMS PLATFORM.....	11
2.4.1	Xaura2 Architecture	11
2.4.2	Xaura2 Tree.....	11
2.4.3	Xaura2 Actors	12
2.4.4	Xaura2 - MediaClub – resources	13
2.4.5	Xaura2 - MediaClub - web site & Content Lib.....	13
2.4.6	Managing and publishing content.....	13
2.5	MEDIA CLUB – E-COMMERCE SERVER (TISCALI PAYMENT GATEWAY)	14
2.5.1	How does it work	14
2.5.2	How to integrate it	14
2.5.2.1	Order creation.....	14
2.5.2.2	Order editing.....	15
2.5.2.3	The callback url.....	16
2.5.2.4	Order refunding.....	16
2.5.3	Success and failure url	17
2.5.4	TPG webservices.....	17
2.5.5	Backoffice area	17
2.5.5.1	Order detail	17
2.5.5.2	Files management.....	17
2.5.5.3	Payment mode management	18
2.6	QUERY SUPPORT FOR CLIENTS (FHGIGD).....	18
2.7	P2P FOR C2C BASED ON AXEPTOOL TECHNOLOGY (CRS4).....	18
3	DISTRIBUTION VIA INTERNET TOWARDS PC, SEJER CHANNEL (SEJER).....	19
3.1	SEJER DISTRIBUTION CHANNEL (SEJER)	19
3.1.1	General Architecture (Production/gathering and distribution) (SEJER)	20
3.1.2	Production and protection (SEJER).....	21
3.1.3	WEB Portal of SEJER	22
3.1.3.1	The publisher's access.....	22
3.1.3.2	The client's access.....	24
3.1.3.3	The end user's access	24
3.2	CONTENT FACTORY OF SEJER (SEJER)	27
3.3	SEJER VS AXMEDIS SCENARIOS (SEJER)	28
3.4	AXMEDIS TOOLS AND THE INTEGRATION WITH SEJER DISTRIBUTION (SEJER).....	29
3.4.1	Ordering the content	29
3.4.2	Consulting the content	30
3.5	SEJER PLAYER WITH AXMEDIS SUPPORT (SEJER).....	30
4	DISTRIBUTION TOWARDS MOBILE (WP4.7, WP9.5: COMVERSE).....	34
4.1	ARCHITECTURE TERRITORIES:	34
4.2	COMMON SYSTEM GUIDELINES	34
4.2.1	System Interfacing Protocol	34
4.2.2	System Implementation and Deployment Platforms.....	36
4.3	DISTRIBUTION TERRITORY.....	36
4.3.1	General Architecture	36
4.3.2	Sub Systems Incorporated	36
4.3.3	Actors	37
4.3.4	Main Scenarios.....	37
4.3.4.1	Get Catalogue (Category Tree) Scenario.....	37
4.3.4.2	Get Category Content Offering, Personalized and Filtered by Device-Capability Scenario.....	38

4.3.4.3	Sample Content Scenario	40
4.3.4.4	Buy and Push Content Scenario	41
4.3.5	Data Structure	42
4.3.5.1	Content Domains	42
4.3.5.2	Primary Objects	43
4.3.5.3	Data Origins, Adaptation and Transformation	43
4.3.6	Sub Systems – Detailed Specifications	46
4.3.6.1	COMVERSE APS	46
4.3.6.2	Handset Management Service	47
4.3.6.3	Personalization Service	54
4.3.6.4	AXMEDIS Plugin	60
4.3.6.5	P2P AXEPTool	65
4.4	TRANSCODING TERRITORY	66
4.4.1	General Architecture	66
4.4.2	Modules	66
4.4.2.1	COMVERSE Transcoding Server	66
4.4.2.2	Transcoding Platform	66
4.4.2.3	P2P AXEPTool	68
4.4.2.4	System Common Data – Detailed Structure	69
4.4.2.5	Configuration – Detailed Structure	70
4.4.2.6	HMS/HME Configuration – Detailed Structure	71
4.4.2.7	Device Capabilities Data – Detailed Structure	72
4.4.2.8	HMS/HME Protocol – Detailed Structure	75
4.4.2.9	PS/PE Protocol – Detailed Structure	79
4.5	SUPPORTING PROTECTION AND SECURITY WITH AXMEDIS SUPPORT (COMVERSE)	81
5	DISTRIBUTION TOWARDS I-TV VIA SATELLITE (WP4.8, WP9.3: EUTELSAT)	82
5.1	PASSING FROM AXMEDIS AXINFO TO OPENSKEY PACKAGE ON THE SERVER SIDE (EUTELSAT)	82
5.1.1	Basic Features	85
5.1.2	Internal Layers	86
5.1.3	System Structure	87
5.1.4	Modules	87
5.1.5	Actors	89
5.1.6	Content Packaging	90
5.1.7	Content Sending	91
5.1.8	Content Transmission	92
5.1.9	Content Addressing	92
5.1.10	From AXInfo to OPENSKEY Package	93
5.1.11	API Server	94
5.1.12	Web Site	96
5.1.13	Broadcast Service Guide	97
5.2	RECEIVING OPENSKEY PACKAGES ON THE CLIENT SIDE (EUTELSAT)	99
5.2.1	OPENSKEY B2C Application for the Content Consumer (GUI)	101
5.2.2	OPENSKEY B2B Application for the Content Provider (API)	105
5.2.3	OPENSKEY towards AXMEDIS	106
5.3	FILTERING OF AXMEDIS OBJECTS ON THE CLIENT SIDE (CRS4)	109
5.3.1	The Virtual Channels composition[CRS4]	111
5.3.2	The Player [CRS4]	112
5.3.3	User Profiling in the client side [CRS4]	112
5.3.4	Graphic User Interface for the Application setting[CRS4]	112
5.4	REFERENCES	113
6	DISTRIBUTION TOWARDS PDA VIA KIOSKS (WP9.6: ILABS, DSI, EXITECH)	115
6.1	OVERALL USE CASE	116
6.2	OVERALL ARCHITECTURE	117
6.3	TEMPORAL DIAGRAMS & RELATED GUI ASPECTS	117
6.3.1.1	User interface language selection	118
6.3.1.2	User registration	118
6.3.1.3	User login	119
6.3.1.4	Content load, selection & preview	120
6.3.1.5	Content adding to chart	122
6.3.1.6	Content purchase	122

6.3.1.7	Content delivery	123
6.3.1.8	System maintenance	125
6.4	KIOSK SERVER ARCHITECTURE	128
6.4.1	Kiosk Server Architecture: Application Front-end Management	128
6.4.1.1	Kiosk Application Front-end API	129
6.4.2	Kiosk Server Architecture: Data Management	130
6.4.2.1	Data Management API	131
6.4.3	Kiosk Server Architecture: User Management	133
6.4.3.1	User Management API	134
6.4.3.2	User Management external connection	135
6.4.4	Kiosk Server Architecture: Delivery Module	135
6.4.4.1	Delivery Module API	136
6.4.5	Kiosk Server Architecture: e-Commerce Module	137
6.4.5.1	e-Commerce Module API	138
6.4.6	Kiosk Server Architecture: Satellite Reception	138
6.4.6.1	Satellite Reception API	140
6.4.7	Kiosk Server Architecture: Catalogue Management	140
6.4.7.1	Catalogue Management interfaces description	141
6.4.7.2	Catalogue Management API	141
6.4.8	Kiosk Server Architecture: System Management	141
6.4.8.1	System Management API	143
6.5	DATABASE ENTITY RELATIONSHIP	144
6.6	SESSION OBJECT AS VALUE OF THE ATTRIBUTE PARAMETER	144
6.6.1	User object	144
6.6.2	User card object	145
6.6.3	User billing address object	145
6.6.4	User list object	145
6.6.5	Catalogue object	145
6.6.5.1	XML Representation of a catalogue	146
6.6.6	Catalogue list object	147
6.6.7	Kiosk object	147
6.6.8	Loading schedule object	149
6.6.9	Billing object	149
6.6.10	Chart object	149
6.6.11	Billing log object	151
7	AXMEDIS PORTAL (EXITECH)	153
7.1	OVERVIEW	153
7.2	CONTRACTORS AREA, (PRIVATE ACCESS):	156
7.3	AFFILIATED AREA (PRIVATE):	158
7.4	USERGROUP AREA (PRIVATE):	159
7.5	PUBLIC AREA:	160
7.6	PORTAL ADMINISTRATION AREA:	161
7.7	MAIN SERVICE CHARACTERISTICS	166
7.7.1	Documents Organisation	166
7.7.2	Documents Download	167
7.7.3	Documents Upload	167
7.7.4	Web search engine	169
7.7.5	Mailing lists	169
7.7.6	News	169
7.7.7	Newsletter	171
7.7.8	User Profile	171
7.7.9	Contractors and Affiliated web pages	172
7.7.10	Events	174
7.7.11	CVS	175
7.7.12	Conference management	175

1 Executive Summary and Report Scope

The full AXMEDIS specification document has been decomposed in the following parts:

- A. general aspects up to the description of the content model
- B. Viewers and players, including plug ins, etc.
- C. Content Production tools and algorithms
- D. Fingerprint and descriptors algorithms and tools
- E. Database area, query support and Content Crawling from CMS
- F. AXEPTool area, for B2B distribution and Programme and Publication for B2C distribution
- G. Workflow aspects and tools
- H. Protection tools and support, Certification and Supervision and Accounting tools
- I. Distribution tools and AXMEDIS Portal
- J. Definitions, tables, terminology, acronyms, lists, references, links and Appendixes

This document contains Part B only.

This document report the early specification of the distribution channels demonstrators integrated in the AXMEDIS architecture.

The channels considered are:

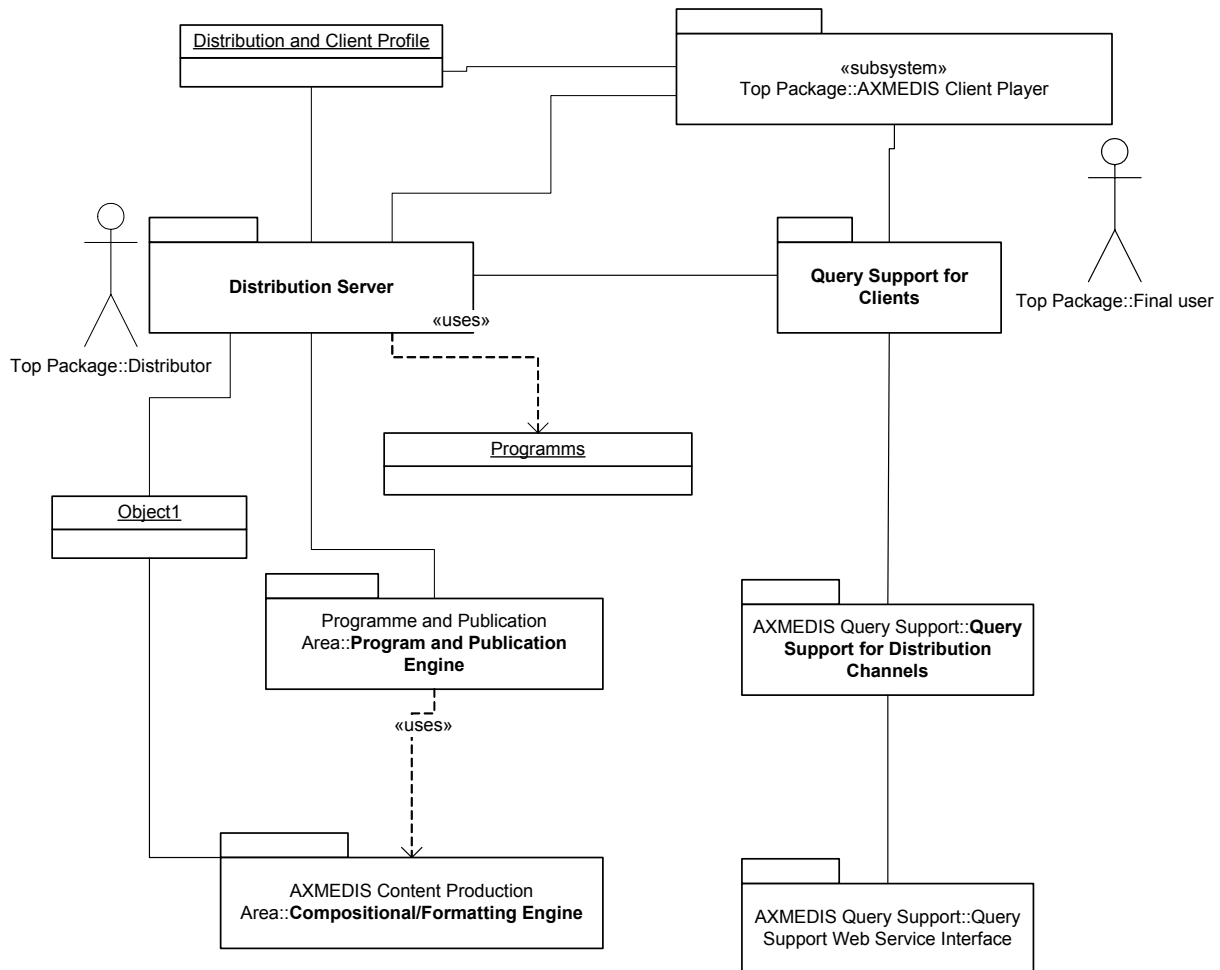
- Internet towards PC of TISCALI: mainly for audio and video distribution
- Internet towards PC of SEJER: mainly for educational content distribution
- Satellite data broadcast towards PC of EUTELSAT: mainly for video and audio content distribution, when is for B2C distribution
- Satellite data broadcast towards AXMEDIS Factories of EUTELSAT: for AXMEDIS objects distribution, when is for B2B distribution
- Satellite data broadcast towards AXMEDIS Kiosks of EUTELSAT: for AXMEDIS objects distribution, when is for B2B distribution
- Towards Cellular Phones of COMVERSE: mainly for audio
- Towards PDA of ILABS: distribution of content towards Kiosks, and from these to clients with PDAs.

COMVERSE has not contributed as expected to this deliverable. In their distribution channel, presently, there is not evidence or report to the content owner about how many objects have been sold or at least this number is only estimated counting transactions in a NON AXMEDIS compliant environment. The number provided has to be trusted by the content owner while the objects are distributed without monitoring any transaction or distribution with AXMEDIS tools. In addition, the so called Transcoding tool has not been described neither integrated on paper with the AXMEDIS tools.

This means that the COMVERSE distribution solution cannot get in some sense the “AXMEDIS certification”, that means that their solution presently is not AXMEDIS oriented since it does not guarantee the safeness and trace ability of the activities performed. This problem has to be solved to provide the right demonstrator of AXMEDIS in WP9.

2 Distribution via Internet towards PC, TISCALI channel (WP4.6, WP9.4: TISCALI)

Distribution Area



The Compositional/Formatting Engine is reached by means of the WF Manager.
See for instance the Compositional/Formatting Engine Page and Design

2.1 TISCALI distribution channel (TISCALI)

The functionalities of the system for content distribution toward Internet could be mainly grouped in two groups, the first describes all back office functionalities provided by the system, the second one describes the end user functionalities.

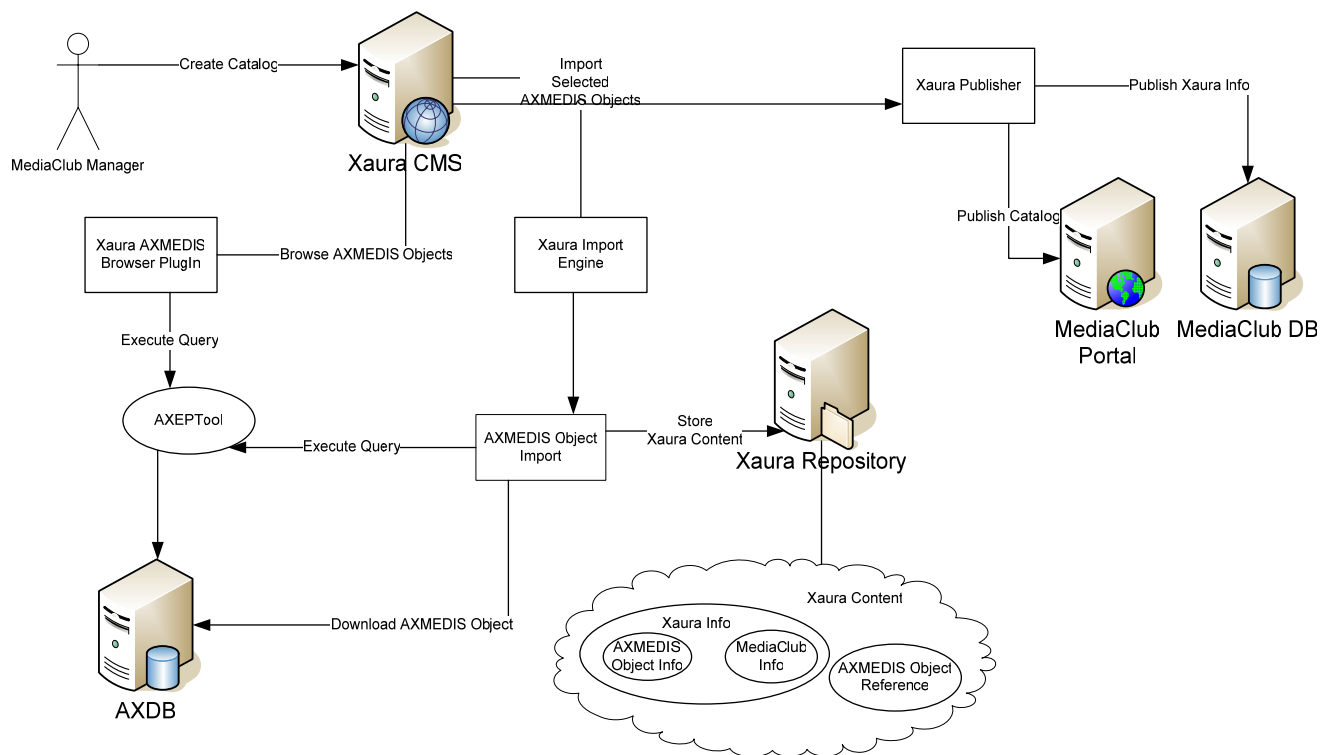
The internet distribution system core is based on a Content Management System (CMS) called Xaura2 that allow the “Media Content Manager” to organize contents in it’s own catalogue (media file, description, price,etc) and give it available for internet distribution (Media Portals).

Over the Xaura2 system the others component that combined to create the distribution-to-internet system are:
AXMEDIS Project

1. **AX Plugin** – A Xaura2 Component that shall interfaces the AXMEDIS platform, enabling and encapsulating AXMEDIS operations; retrieve catalogs, search for content, download media, permissions, etc
2. **AXEPTTool** – An AXMEDIS system that is required for the database (AXDB), search and storage services it offers
3. **Xaura Media Repository** – The media content repository enhanced to support AXMEDIS content too.
4. **MediaClub Portal** – The web site portal offering media contents, managing media contents, end user subscription and profile, user portfolio, etc.
5. **MediaClub E-Commerce** – Payment gateway system, integrated with the MediaClub Portal to allow content payment to the end user
6. **MediaClub Downloads** – Allows the and user to access to the media content offered in the MediaClb Portal.
7. **AXMEDIS PMS** – Is the “AXMEDIS DRM” to release to the end user licenses for AXMEDIS contents

2.2 MediaClub back-office

Back office architecture schema for AXMEDIS content distribution via Internet is summarized in the sequent figure.



Mediaclub Manager implements the Xaura2 platform for organizing and publishing Multimedia content catalogues (AXMEDIS objects, video, etc).

The MediaClub Manager can search and retrieve AXMEDIS objects from the “AXMEDIS universe” via the AXMEDIS Browser Plug-In which implements the AXEPTTool Interface

Once one AXMEDIS object is selected for importation the MediaClub system extracts all related AXINFO and, by combining these with the MediaClub Info (price, category, license duration etc.), creates a Xaura Content

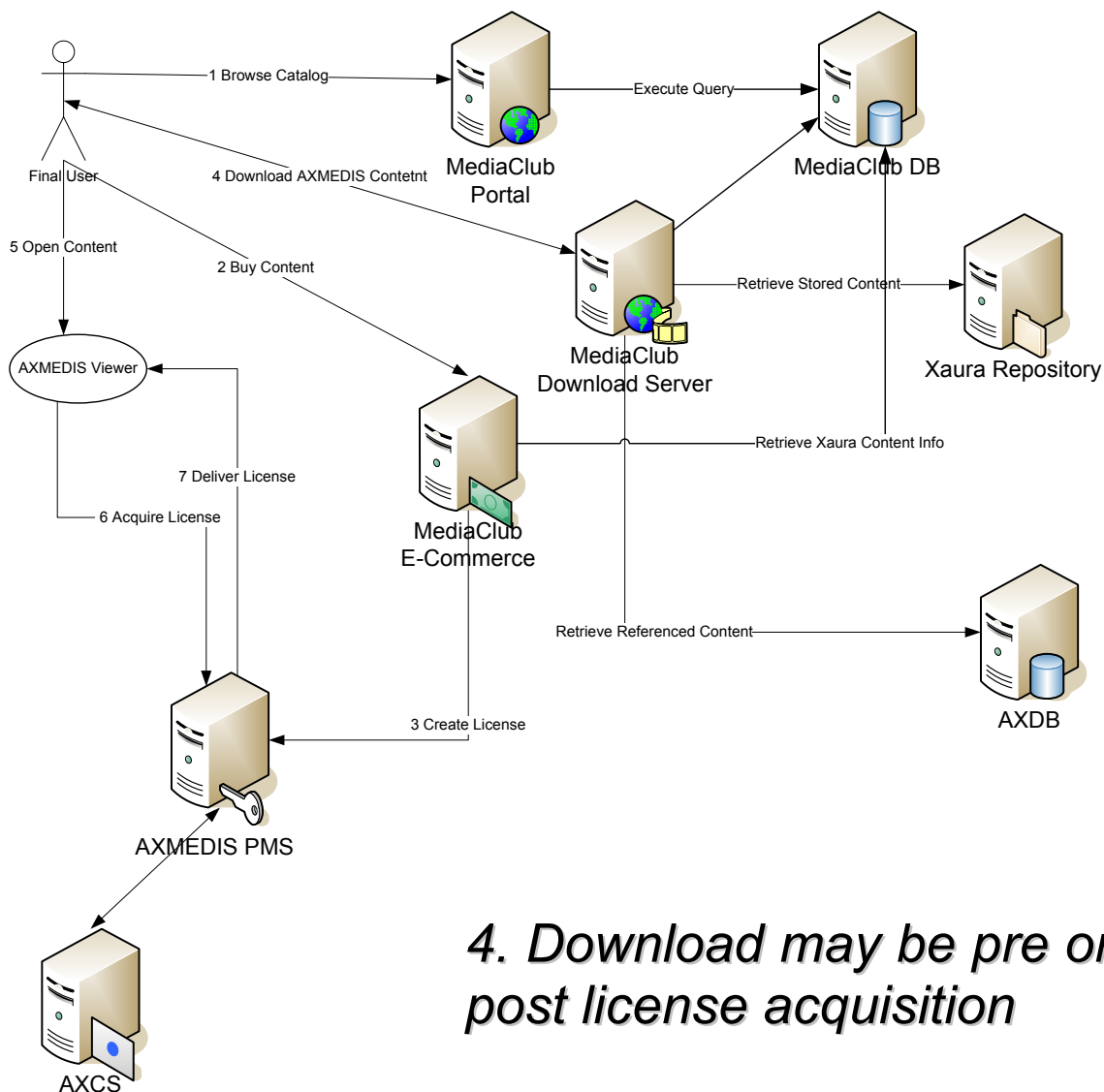
Once the Xaura Content is selected, the MediaClub Manager can start operating the Catalogue management. He can select the content to be published and where such content should be published (which instance of the MediaClub).

Now the content could be browsed and purchased by the end users surfing the medioclub portal.

Xaura2 CMS Platform functionalities are shown on chapter 2.3

2.3 MediaClub Final User

Front End architecture schema for AXMEDIS content distribution via Internet is summarized in the sequent figure. The schema describes the system components that interfaces the end user during AXMEDIS content acquisition.



The end user can surf the MediaClub without giving it's own credential and it can access to all the free content (i.e. trailers, ...) (our first assumption).

To purchase AXMEDIS content the MediaClub subscription is needed. The subscription and login data are stored in the Mediaclub DB as the user media content portfolio, the user credits and the AXMEDIS contents license status (enabled, expired, activation date, etc)

All media contents are provided by the MediaClub Download server.

Depending on the AXMEDIS content license configuration and the way how the Mediaclub Manager has structured his Portal, the end user can download or not content without having already bought the AXMEDIS content license.

Mediaclub E-Commerce server allow mediaclub users to pay for content acquisition. Once payment succeeded the E-commerce server request a license creation for that AXMEDIS content and for the specific end users.

The end user interfaces the AXMEDIS PMS server, toward the AXMEDIS Viewer, to request the license for the content. Furthermore the license acquisition the end user can view the AXMEDIS content.

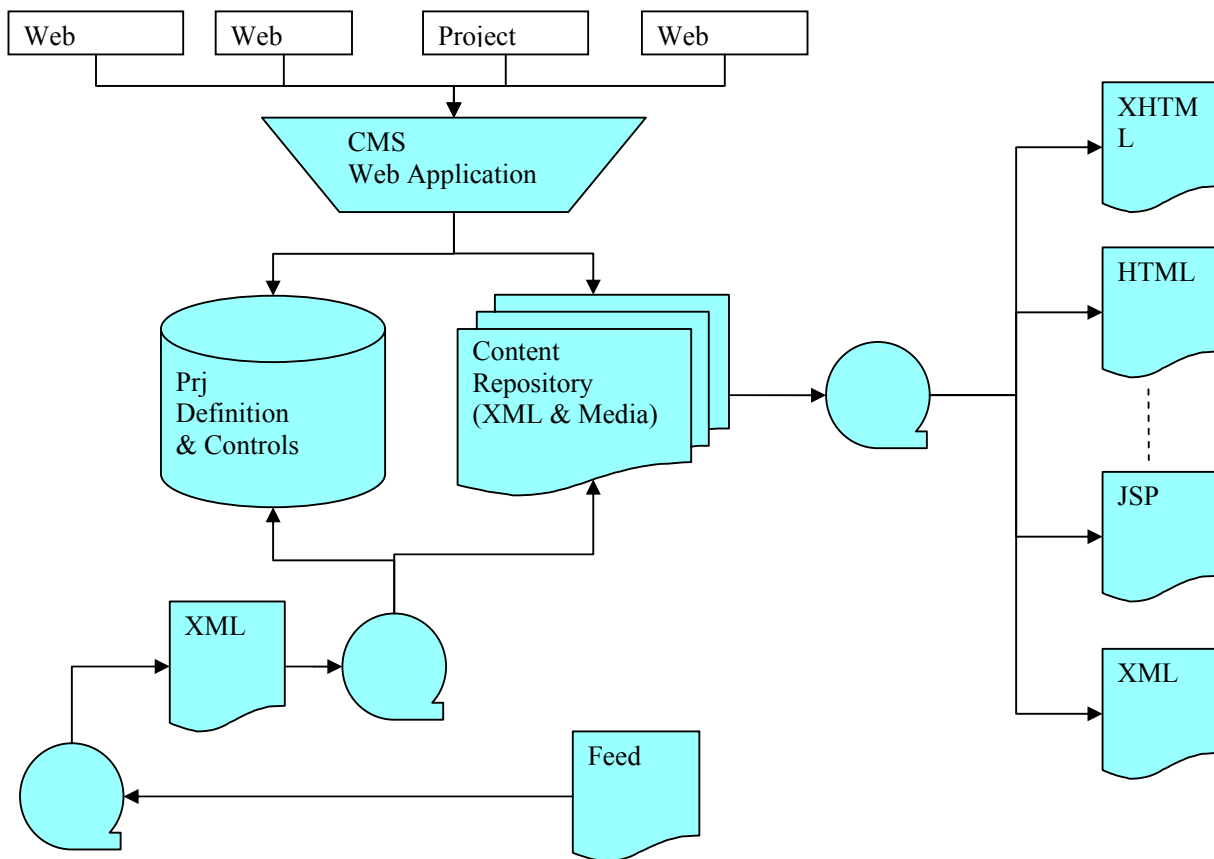
2.4 Tiscali CMS Platform

Xaura2 is a platform for multi-site and multi-channel publishing

Xaura2 is a Java application which uses XML/XSL for content transformation and rendering

Xaura2 uses a static content publishing process in order to optimize http content delivery

2.4.1 Xaura2 Achitecture



2.4.2 Xaura2 Tree

- Resources

- **Contains the MediaClub project definition structure (schema and layout of any part of the MediaClub project)**
- **Web Site**
 - **Contains the MediaClub web site tree**
- **Content Library**
 - **Contains the content (AXMEDIS Objects) and the category in witch the contents are organized**
- **Media Library**
 - **Contains media content (image) and the category in witch this contents are organized**

2.4.3 Xaura2 Actors

- **Xaura2 Administrator**
 - Create new projects defining:
 - Project name and project quota (Mbyte)
 - Project administrator account
 - Project components:
 - One or more web sites (project targets)
 - Content library
 - Media library
 - Project Administrator
 - Gestione degli utenti (web developer; web editor; web publisher) e le loro permission sul progetto o parti di esso
 - Specialmente per I ruoli di web editor e web publisher il project administrator può assegnare i diritti di accesso a livello di web site, di sezioni, di categorie di contenuti
 - Report sul quota utilizzato dal progetto
- **Web developer**
 - specifies addresses, structure and deployment modes of the project (ftp; local; http; other) both for websites and media
 - Specifies web site structure
 - Creates contenuti structure
 - Creates content structure an corresponding relations with component/target (web sites) of the project
 - specifies content schema (xsl)
 - Specifies lay-out specification and associates corresponding target
- **Web Publisher**
 - Authorizes content publishing on the website target
- **Web editor**
 - Content item authoring
 - Triggers publishing process on the target

2.4.4 Xaura2 - MediaClub – resources

- Contents
 - Contains the MediaClub project definition structure (schema and layout) for contents
 - File named as s_XXXXXX are schema file
 - File named as l_XXXXXX are layout
 - Schema define the content schema
 - Layout define the rendering format of the page
- Feed
 - Contains the xsl layout to import data from an external source (AXEPTool)
- Img
 - Contains images used to render content in the Xaura2 Back Office
- Inc/style
 - Contains the stylesheets used to render content in the X2 B.O.
- Pages
 - Contains schema and layout of pages published in the site
 - x_home_page.xsl
 - x_category.xsl define the content category pages
 - x_faq_home.xsl define the faq main page
 - x_sx.xsl define the sx column of videoclub pages
 - x_dx_info.xsl define the rx column of videoclub pages
 - x_html.xsl define the tools content shema and layout (test page/ help pages etc)
 - x_jsp.xsl define the integrationwith the MediaClub dynamic side part

2.4.5 Xaura2 - MediaClub - web site & Content Lib.

- **WEB SITE:** Contains all file definition published in the MediaClub web site witch means:
 - Html files
 - Jsp reference files
 - Images
 - Stylesheet
 - Other file referenced in the site
- **CONTENT LIB.:** Contains all content file present in the MediaClub organized in categories. witch means:
 - AXMEDIS Object details
 - Non AXMEDIS Object detail
 - Infos (How to) contents

2.4.6 Managing and publishing content

- How to change and publish a content
 - Go to the content library and select the content you want to change
 - Do the changes you need in the MediaClub Info
 - Save the content
 - Than select the section you want to publish the changes:
 - Film means that changes will be published in the MediaClub web site
 - MediaClub means that changes will be uploaded to the portfolio management application
 - Publish the content
- How to publish a section page
 - Go to the web site side and select the page you want to publish
 - Publish the page
 - This operation is needed after changes in the content library or in the media library

2.5 MediaClub – E-Commerce Server (Tiscali Payment Gateway)

Tiscali Payment Gateway (TPG) is the payment platform provided to Mediaclub Portals (from now Vendors and Shops) to allow them selling goods through the internet without having to trouble with https connections, protocols and communications with banks.

TPG exposes several payment modes into the same frontend and backend interfaces; vendors and shop administrators can choose one or more of them depending on their preferences, using the same backend interface even if each payment mode performs completely different operations.

2.5.1 How does it work

TPG works as a generic payment system, exposing a unique interface for all registered payment modes. A TPG payment operation follows these steps:

- 1) The shop creates a **Payment Order** and gets its id back (**createOrder** call)
- 2) The shop sends the user's browser to the payment page (**editOrder.shtml**).
- 3) The user selects its preferred payment mode, fills in the form and completes the operation. If everything goes fine a success notification is displayed to the user (**paySuccess.shtml**).
- 4) TPG sends a success notification to the shop with the transactionId previously communicated to the shop (**callbackUrl**).
- 5) TPG sends the user to the success page into shop's web site (**successUrl**).

If the user abandones TPG's pages by clicking on the Exit button he is sent back to the shop's **failurePage**.

2.5.2 How to integrate it

Using TPG for electronic payments is mainly a matter to allow the communication between the shop's server and TPG server.

This communication is performed in two steps: the order creation and the callback. Between these steps there's the order editing, which is performed by the user into TPG system.

In some cases an order can need to be refunded. This operation can be done by calling the order refund servlet.

Depending on the result of the payment operation the user will be sent back to the shop's success url or failure url.

Recently a new and more modern way to integrate TPG via SOAP webservice has been introduced: see details in the TPG webservice paragraph.

2.5.2.1 Order creation

To create a new order into TPG system the **createOrder** servlet must be invoked.

Before calling this servlet the shop must create a unique transactionId (a good hint is to use the current date's timestamp) which will be used together with the shop Id to refer the order into TPG pages.

The call for invoking the createOrder servlet is:

https://tpg.tiscali.com/services/createOrder?shop=<SHOP_ID>&transaction=<SHOP_TRANSACTION_ID>&amount=<AMOUNT>&email=<USER_EMAIL>

There's the parameters explanation:

shop	The shop's Id as registered into the backoffice.
transaction	The unique transaction id assigned from the shop to the order.

shop	The shop's Id as registered into the backoffice.
amount	The amount of the order. Dot (“.”) as decimal separator, no grouping separator must be used.
email	The email of the customer which is going to pay

Table 1: Order creating servlet call: mandatory parameters

In addition to these mandatory parameters, optional parameters can be provided for different purposes or to better describe the order. Below is the list of these optional parameters:

paymentMode	Id of the preferred payment mode. If a valid id is provided the order editing page will display this payment mode already selected.
Country	The country which will provide payment invoices. It must be a valid iso country code. By default is the same country in which the payment gateway resides.
Language	The language to use in the user interface. It must be a valid iso language code. By default is EN (english).
description	The goods description (useful if a payment mode with invoice is used)
Vat	The vat rate to apply. By default is 20. It must be a valid number with dot (“.”) decimal separator and no grouping separator.

Table 2: Order creating servlet: optional parameters

If the order creating call is correctly done TPG answers with a string containing numbers and letters which refers the accepted order. This is the internal order id which will sent back to the shop once the order will be correctly payed from the user.

If anything goes wrong during the createOrder call the shop will receive a **server error** with a description of the occurred error.

NB: be careful to **never** put the TPG transaction id into the pages displayed to the user. This value must be only known from TPG and the shop's site, and its secretness is a strong point for the security of the payment operations.

2.5.2.2 Order editing

Once the order has been created the user can be sent (by clicking on a link or with a redirection) to the payment page, which resides into TPG's system.

The url of the payment page is this one:

<https://tpg.tiscali.com/gateway/editOrder.shtml?shop>

https://tpg.tiscali.com/gateway/editOrder.shtml?shop=<SHOP_ID>&transaction=<SHOP_TRANSACTION_ID>

where **shop** and **transaction** parameters are the same used in the previous order creating call.

TPG will recognize the order and display to the user a payment page for the amount specified in the order create call.

NB: Any parameter sent to the editOrder.shtml page is **stored** from TPG and **added** to the success url or failure url in order to help the shop server restoring the session when the user comes back to it.

2.5.2.3 The callback url

If the payment operation ends correctly TPG will perform a call to the shop's **callback url** configured in the backoffice.

The most important parameter the callbackUrl **must** handle is TRANSACTION_ID, which is the transaction id of the order as provided from TPG at point 2). Other parameters are sent to be helpful to callback operations. Below is the complete list of parameters sent to the callback url:

TRANSACTION_ID	The id of the transaction assigned from TPG
SHOPTRANSACTION_ID	The id of the transaction assigned from the shop
AMOUNT	The amount of the transaction
EMAIL	The email of the user performing the transaction
PAYMENT_MODE	The id of the used payment mode

Table 3: Callback url: parameters sent back to the shop

*It's **strongly important** for the security of the transaction to keep in mind that the most reliable parameter to handle from the callback url is TRANSACTION_ID because it is only known just from TPG and the shop's server. All other parameters can be displayed in the pages or urls during the payment operation and so they should be used just for checking or non-authorized purposes.*

What the callback URL must do is just to return the string “OK” (plain text, no HTML or XML tags must be used) after checking all parameters. Any other response will be considered as an error and the order won't be considered exhausted.

If any error occurs during the callback call (for instance if the server was down) TPG will try contacting this url for other 10 times every 10 seconds. If the server won't give a correct answer in this time the order will be considered successfully completed as well, but it will remain in a pending status (PROCESSED). To move the order to the final status EXHAUSTED it will be needed to manually force another callback url call by clicking on the **exhaust** button in the backoffice area.

2.5.2.4 Order refunding

Order refunding can be programmatically done using the order refund servlet by calling this url:

<https://tpg.tiscali.com/services/refundOrder?order>
https://tpg.tiscali.com/services/refundOrder?order=<ORDER_ID>

The **order** parameter is the order id returned from TPG at order creation time, not the shop's order id for obvious safety reasons.

If the order is correctly refunded the servlet will return **no answer** (it corresponds to a *void* method call). Otherwise a server error will be returned with a description of the occurred error.

Typical refunding errors can be:

The order does not exist

The order is in a state which is not compatible with refunding operation. For instance orders in OPENED or PROCESSING states cannot be refunded (the reason is obviously that no money has already been given from the user in these cases).

The paymentMode used to pay does not support refunding. At the moment just order paid with CreditCard can be refunded.

2.5.3 Success and failure url

These urls stored in the TPG backoffice, represents the **returning points** for the user after performing payment operations. It is very important to understand the difference between this urls and the callback url: while these ones are public pages reachable by the user's browser, the callback url is a secret server-to-server port to allow communication between the involved servers.

This means that **no payment related operations should be performed from these urls** (for instance updating user's portfolio) because they are not safe. The parameters given back to these pages are provided just to be helpful for restoring the user session.

Of course success and failure page can be the same (a typical case is a page displaying the user account into the shop).

2.5.4 TPG webservice

TPGService is a SOAP webservice which provides the most useful features of TPG through webmethods.

The WSDL of this webservice is at the address:

<http://tpg.tiscali.com/services/TPGService?wsdl>

TPGService webservice exposes the webmethods below:

getPaymentOrder
createOrder
refundOrder
exhaustOrder

The first one, getPaymentOrder, allows the programmer to retrieve information about a PaymentOrder by providing its order id. The other three webmethods performs the same operations as their servlet “peers”, but allow to obtain more information. For instance the createOrder method returns a **PaymentOrder** instance instead of a simple order id.

In order to get the big advantage by using this webservice is highly suggested to generate client-side stubs so that you do not have to trouble about XML parsing and SOAP protocol details.

A good tool to generate webservice stubs is **WSDL2Java**, provided by **Axis** at <http://ws.apache.org/axis/>

2.5.5 Backoffice area

2.5.5.1 Order detail

This page displays detailed information about a paymentorder, including all order parameters and a log with all the order history.

Furthermore two buttons are provided to perform distinct operations:

- **EXHAUST**: forces the shop's callback url to be called. This button should be used **ONLY** for orders which have been in status PROCESSED for a long time (>1hour). Every other status will cause an error.
- **REFUND**: refunds the user the whole amount of an order. This operation is not supported from all PaymentModes, and should be used for orders which are in status PROCESSED, EXHAUSTED or ABORTED. Every other status will cause an error.

2.5.5.2 Files management

This tool is meant for shop customization and look&feel purposes. It allows you to put and delete files into a special area in the TPG server which responds to the URL

https://tpg.tiscali.com/ext/<SHOP_ID>

There are some **reserved filenames** you can use for predefined purposes:

gateway.css	Defines the css rules that will be used to display the editOrder page
btn_submit_<language> btn_exit_<language> btn_reset_<language>	Images for navigation buttons, localized for the language used in the current order. The value for <language> must be a valid ISO language code.
extra.html	Html additional code which can be included into the editOrder.shtml page.
extra_success.jsp	Jsp additional code which can be included into the paySuccess.shtml page.

2.5.5.3 Payment mode management

This tool allows you to choose which paymentmodes made available to the shop.

The complete list of available payment modes is displayed on the left, and the used ones in the right. You can add a paymentmode by selecting one from the left and clicking in the **add** button, or you can remove a paymentmode by selecting one on the right and clicking in the **[X]** button.

2.6 Query Support for Clients (FHGIGD)

The query support for distribution channels is described in detail in section 5 “Query for Production on Demand” of part E. The general idea is that the final user can enter a query in the AXMEDIS database using the Client GUI. The **Query Support for Clients** creates a query message in a simple XML query language, a client profile is added and then the query is sent to the Distribution Server.

When query results are returned to the Query Support for Clients they are presented to the final user with the Client GUI. The user can select a specific content using the Client GUI and the Query Support For Clients passes this selection on to the Distribution Server.

The **query support for PC using web interface** is integrated as a web server in the distribution server. The final user can enter a query in the AXMEDIS database on the web server of the distributor using the client browser. The Query Support for the Client on the distribution server creates a query message in a simple XML query language and passes it on to the Query Support for Distribution Channels.

When query results are returned to the Query Support for Clients on the distribution server they are presented to the final user on a web page. The user can select a specific content by clicking on the respective link and the Query Support for Clients passes this selection on to the P&P Engine via the Distribution Server.

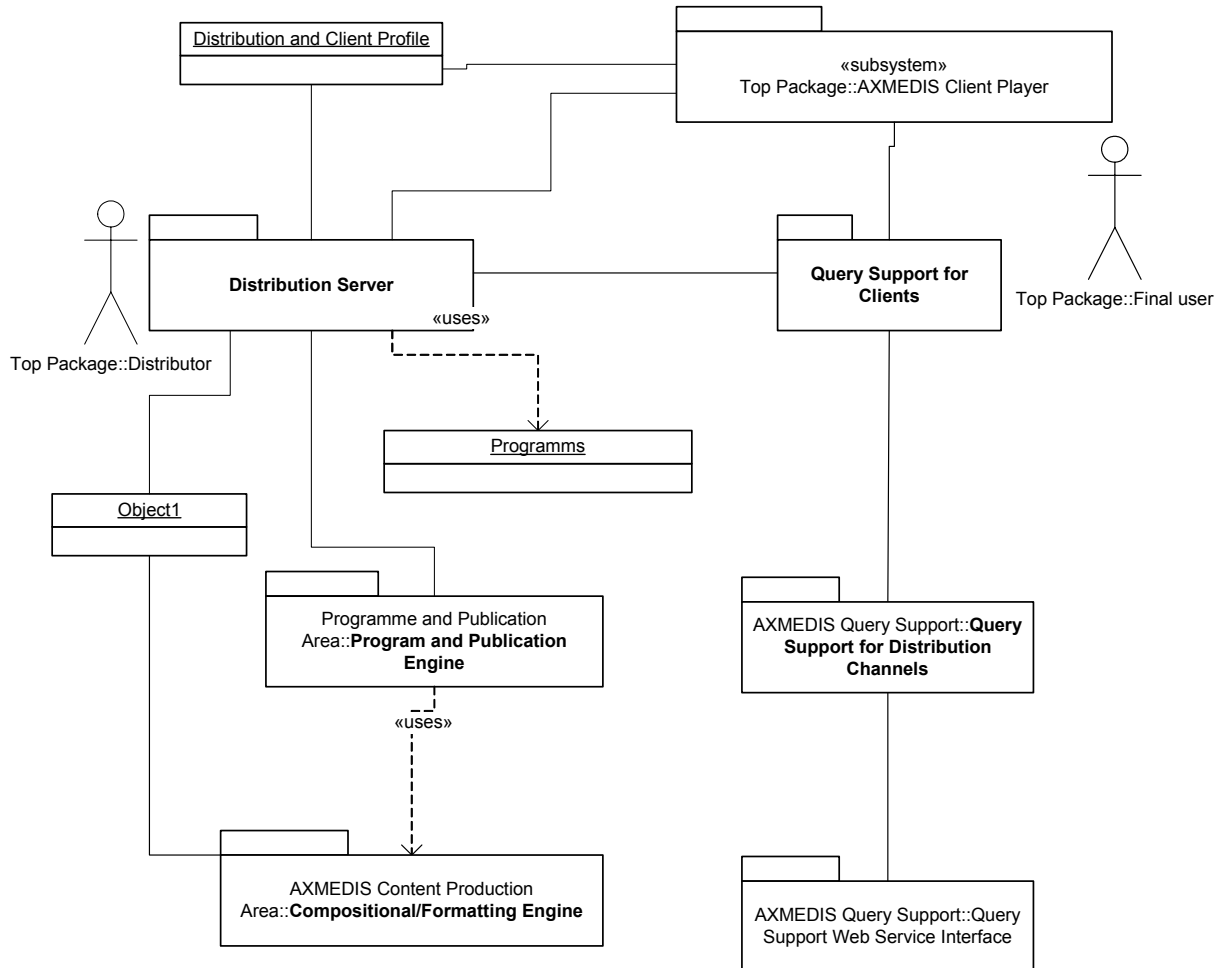
2.7 P2P for C2C based on AXEPTool technology (CRS4)

In this phase, the design of P2P for C2C is less critical then the B2B counterpart. We expect to provide a full specification for the demonstrator activity. The C2C network will use the same solutions of B2B except that in case of scalability and openness requirements where other solution off-the-shelf will be deployed.

The single peer client tool of the P2P network has been called since the beginning “AXMEDIA peers”.

3 Distribution via Internet towards PC, SEJER channel (SEJER)

Distribution Area



The Compositional/Formatting Engine is reached by means of the WF Manager.
See for instance the Compositional/Formatting Engine Page and Design

3.1 SEJER distribution channel (SEJER)

SEJER distribution channel for PC and Tablet PC is the same, only content production differs. Both use a “per user” licensing model, even though to enforce the model, it is sometimes useful to be able to restrict the product to a single device. The license model is subject to the following constraints:

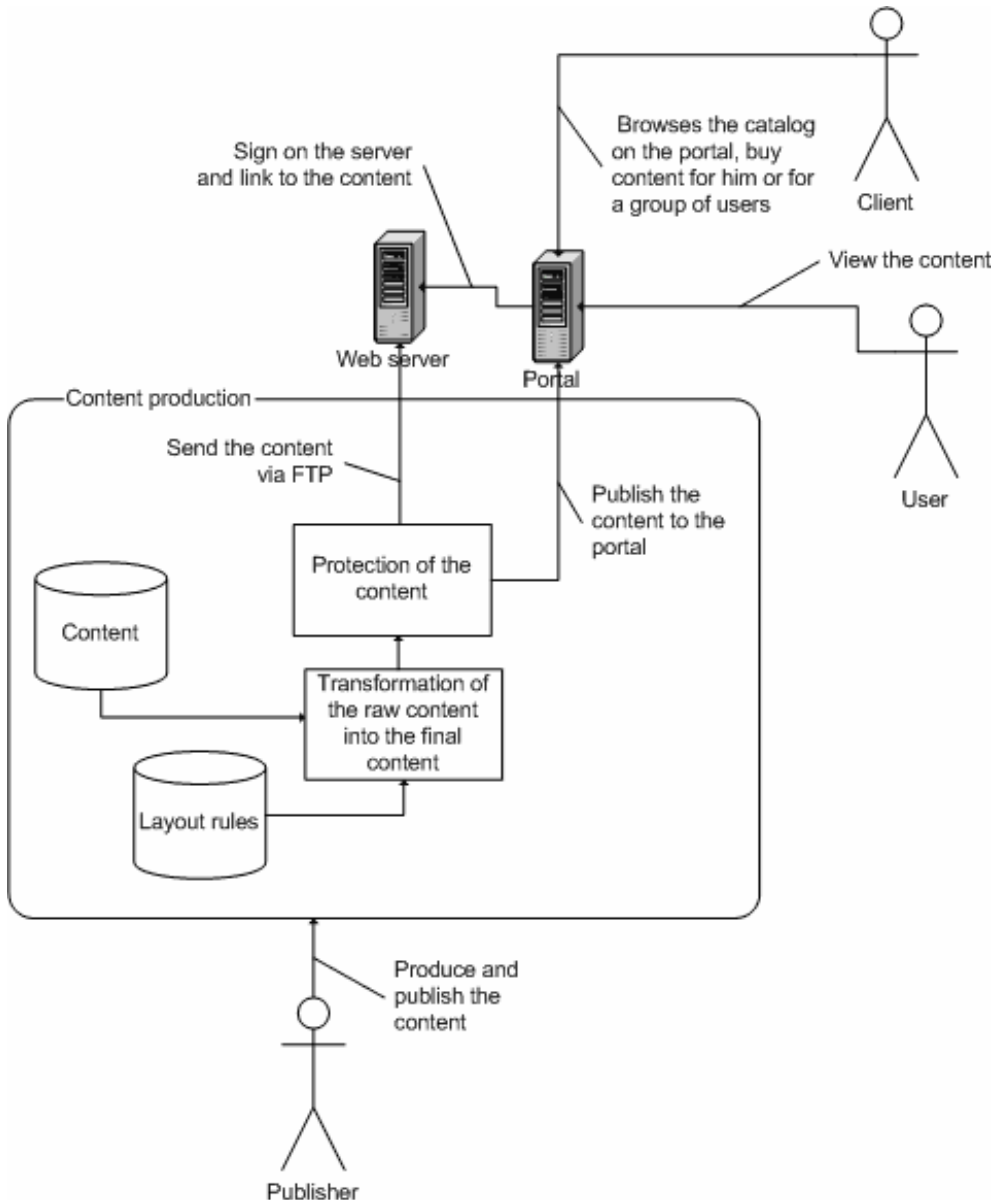
- A user should be able to use it's product everywhere
- No other user should benefit of it's license
- Client should allow personalization of content ; personalization is saved on a per user basis, thus associated to the license
- Possibility of time restrictions
- etc.

What makes SEJER distribution channel different is the ordering process. As we are distributing school products, **the teacher or the school orders the licenses for the students.**

3.1.1 General Architecture (Production/gathering and distribution) (SEJER)

In this part, the architecture is presented at a more abstract level than in the Teacher/Students scenarios, as they are a specialization of B2B scenarios, the Teacher being the client and the End User being the students.

The general architecture is quite simple:



Once the final content is generated from raw content and layout rules, the content is sent via ftp to a web server and published into the web portal. The portal knows how to authenticate its users on the publication server.

From the portal, the connected user, if he/she is a client, can browse the catalog and purchase some content.

The end user can view the content he/she has the right to.

3.1.2 Production and protection (SEJER)

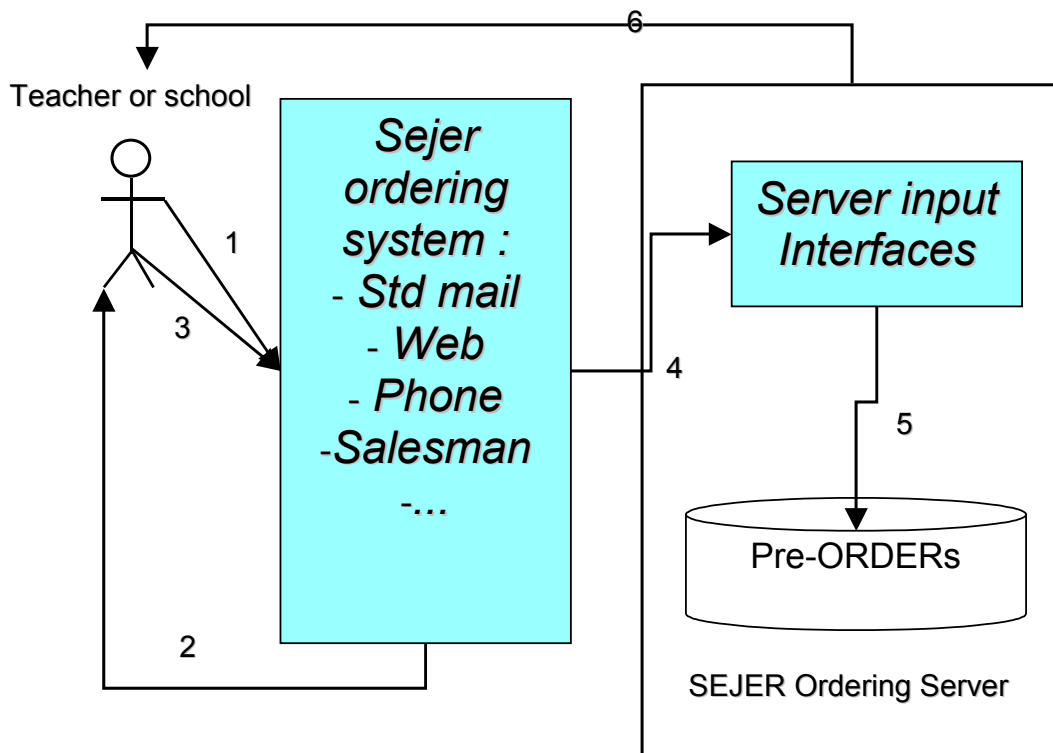
The production of content is automated. On one side, there is the raw content and on the other side, a set of formatting rules (XSLT files, script files, images conversions and aggregation tools, etc.)

To produce the content for distribution, a build process is run which interpret the formatting rules and automatically transforms the raw content into the final product, which contains both the content and the user interface (an electronic school book). Even though a single file is generated, the product can be considered as being an embedded web site.

The data of the product is encrypted and SEJER Mozilla's add-on is needed in order to open the book. The add-on checks the licence of the user before opening any pages of the school book.

The license is retrieved at the end of the ordering process: which is subject to the following constraints:

1. **There is numerous Ordering channels : standard mail, web, phone, salesman, etc.** Therefore, no AXMEDIS license can be directly returned to the buyer at ordering time
 - **The teacher buys N licenses for N students, with some specific parameters depending on Informational system present in the school**
 - **The license is associated to the student by the student itself, or by a dedicated IT personnel because the teacher don't want to manipulate N licenses, or having to perform N installation etc.**
2. **Thus a kind of key is communicated to the buyer at ordering time : the activation number used to associate a Final User to a License. It is only in this second step that the license is fully acquired**

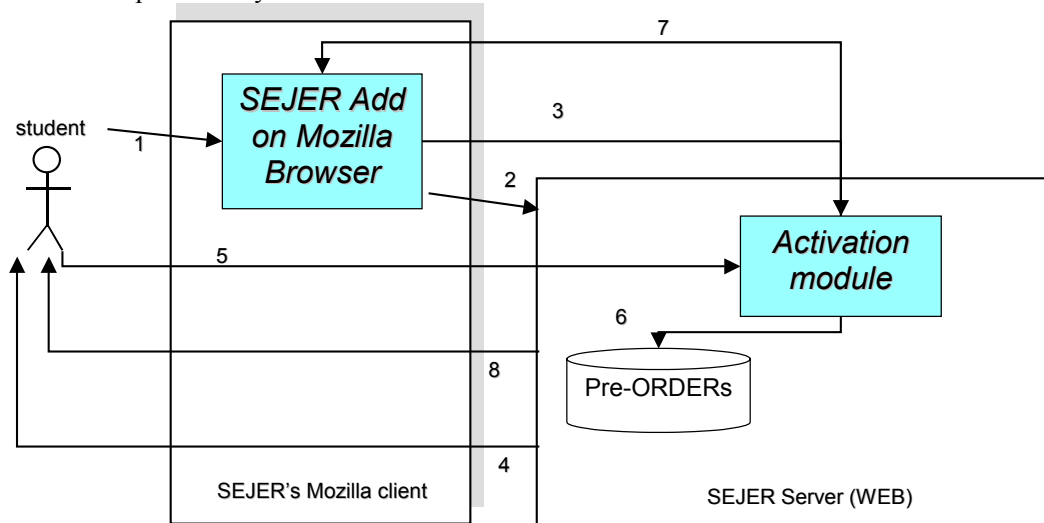


1. Order product => N licenses

2. Bill (web form, or mail etc. depending on the ordering system used)

3. Pay the bill, confirm the order. A valid e-mail must be provided
4. Enter the order in SEJER's Information System
5. Ask for the creation of pending licenses corresponding to the order. N Licenses are created in a specific state : waiting to be associated to a user, for a given Activation number (Note : these are not AXMEDIS Licenses, but some specific SEJER stuff)
6. Send back Activation Number (usually by email, but might be phone, mail etc.)

The registration of the students is done automatically by the system, by using automatic identification information provided by the SEJER Mozilla Add on.



1. Request the object
2. Try to get the resource. Assuming answer is "Permission denied/No license".
3. Request Activation Form to the SEJER Server. **The Add On computes Personal Identification data and send them with the request.**
4. Serve Activation Form
5. Enter Activation Number
6. Update SEJER DB and Get Info related to the requested Activation Number : product ID, DRM Rules associated to the license etc.
7. Notify client that activation has been performed, and serve the corresponding object
8. Display the object to the user

3.1.3 WEB Portal of SEJER

What follows is a description of the architecture of a web portal distributing content from several contributors to business and/or customer.

The distribution involves three different user roles:

- the publisher, who post the content to the portal server
- the client, who purchase the licences to consult the content
- the end user, who consult the content

3.1.3.1 The publisher's access

The publisher post resources to the portal server.

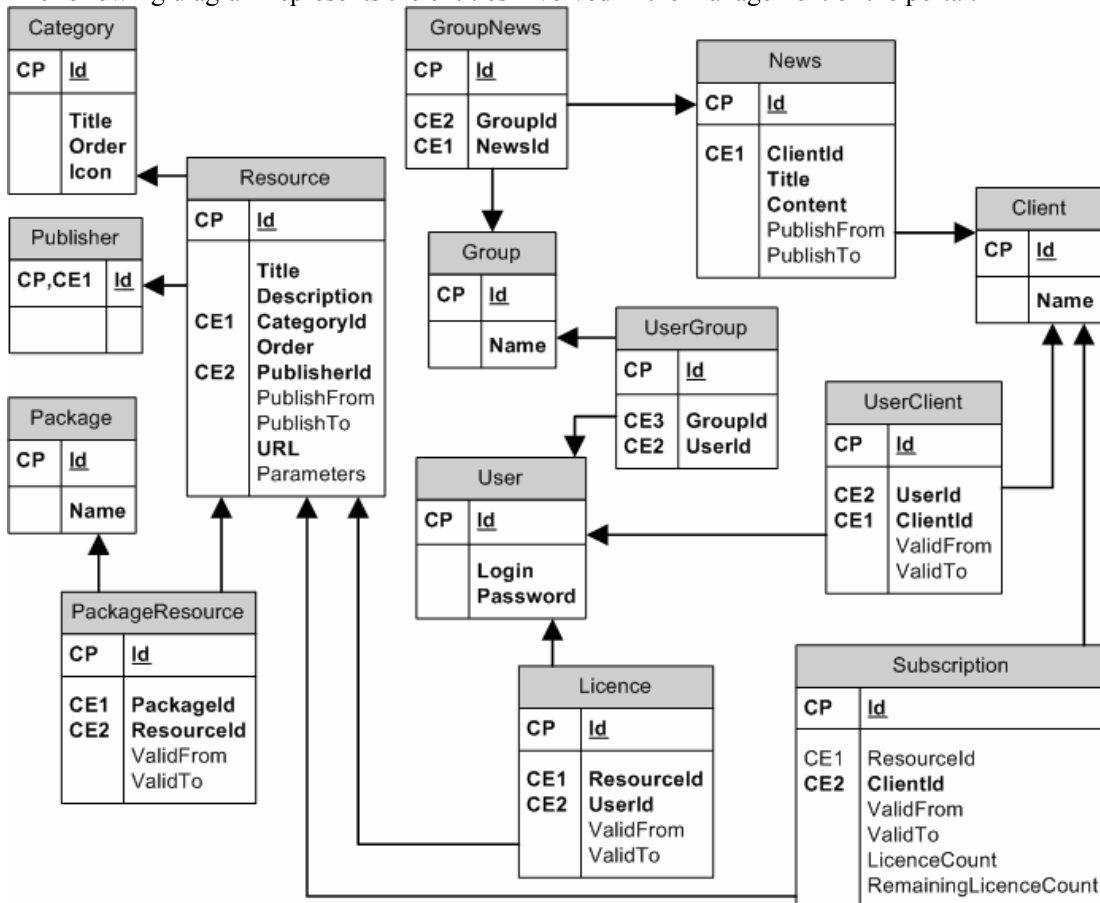
Some of the content provider might not be AXMEDIS compliant.

Regarding the nature of the resources and the access mode implemented by the content providers themselves, several ways are possible to access them:

- **Direct link via a URL without authentication.** This case happens when the resource is already and freely available for everybody.
- **Direct link via a URL with authentication.** Authentication information is embedded in the URL parameters.
- **Access via a login form.** This is the most frequent case and it often requires that the end-user type his/her login and password in, before being allowed to reach the resources.
- **Access via a user licence.** The licence, using when possible hardware information, is generated when the end-user register himself/herself.

In any case, the authentication of the user should happen only when the user log in the portal (Single Sign On).

The following diagram represents the entities involved in the management of the portal.



The publisher can perform the following actions:

- add a category, by defining its label and its icon
- remove a category
- move a category to another position, so can organize them the way he wants
- add resource, by defining its title, the identifier of its publisher, its category, its description, the URL and its parameters and, optionally, the publish from and publish to dates
- remove resource
- move a resource inside a category to another position

- create package
- delete package
- add resource to package
- remove resource to package

The package is used to group resources and sell them as the whole. The client will then be granted on the rights on every components of the package at the moment of the purchase.

- create user
- delete user
- list the groups the users is in
- create group of users
- delete group of users
- check if a user is in a group

3.1.3.2 The client's access

The client can perform the following actions:

- create user
- delete user
- list the groups the users is in
- create group of users
- delete group of users
- check if a user is in a group

This set of actions is equivalent to the one of the publisher, with the restriction that a client (respectively group) can only create user (respectively group) attached to its client account.

- add news, by defining its title, content, publication parameters
- remove news
- add a group in a news publication list
- remove a group from a news publication list

The client can also:

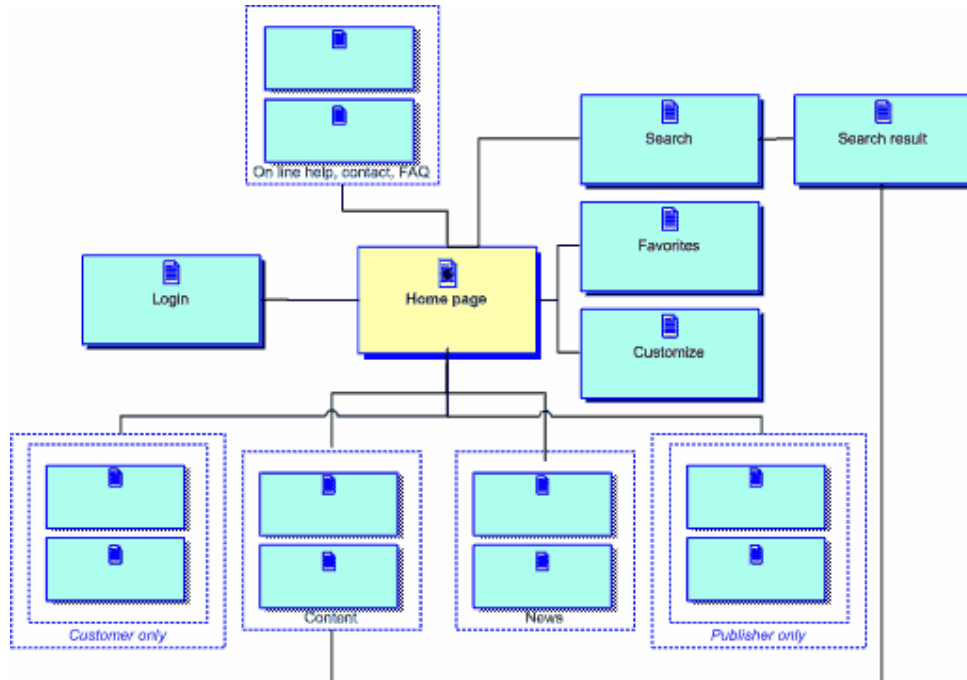
- assign a licence on a resource to a user
- remove a licence on a resource from user, if and only if that licence as not been activated yet

3.1.3.3 The end user's access

The web portal is a very simple internet web site, composed of a main page and four kind of secondary pages:

- the search of content
- the customization of the user's homepage
- the management of the user's favourites
- informational content, such as online help, FAQ, contact list

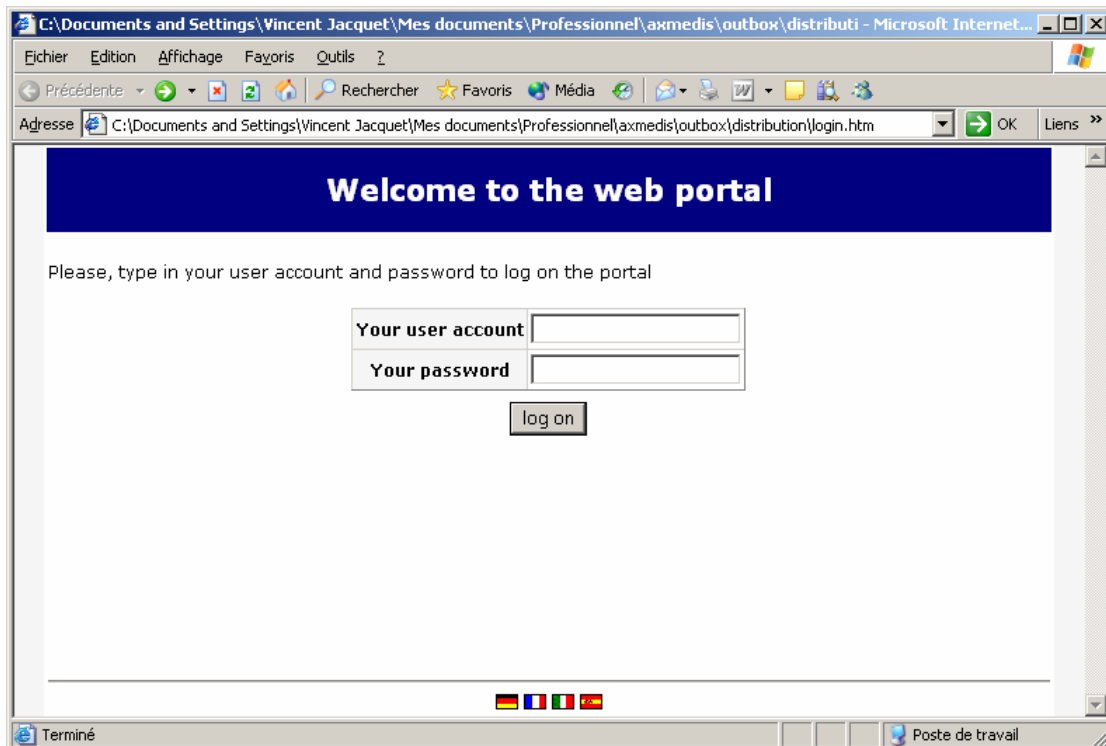
The diagram below represents the several parts of the portal mentioned above plus the client private area and the publisher private area



Its main purpose is to offer to the end-user a list of the content the client has purchased the rights for him/her.

The content is exposed as a hyperlink on the page. The portal simply redirects the user to the publisher resource, in a new target frame to keep the portal visible.

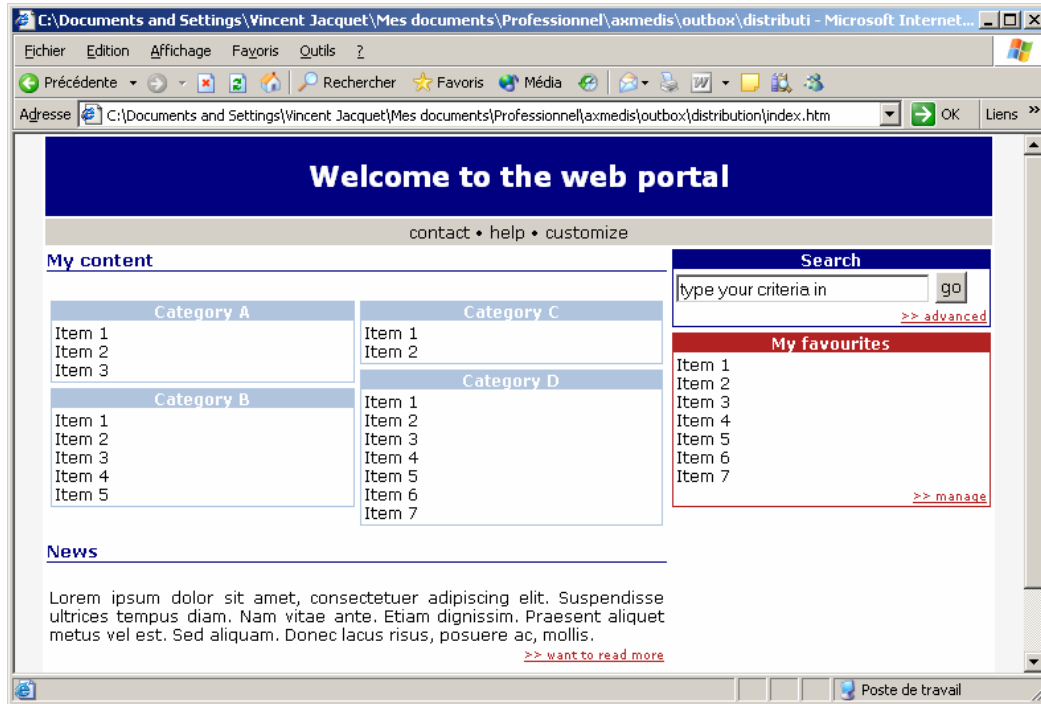
In order to access the homepage of the portal, the user must enter is login and password and, eventually, change the current language of the interface. It should happen only once.



Then, if the log on is successful, the user reach the home page of the portal.

This page is split into four main components:

- The top banner that identifies the portal and provide a tool bar, with links such as contact or help
- The content pane, that list all the content the user as access to
- The news pane, which informs the user about new content availability
- The favourites pane which contains a list of hyperlinks the end-user has selected



This page should fit on an 800x600 screen resolution. When not possible because there is too much content or news to display, a simple scrollbar should be used to scroll the whole page. The usage of frames should be avoided.

The content pane presents all the resources available for the current user. This list is dynamic and depends of the content purchased for/by him.

The resources are grouped by categories. They are all listed on that page, with their title in a hyperlink and, next to the title, a help button can be provided so the user can obtain a description of the resource.

The news pane contains news about the portal and the content distributed in it. This list is also dynamic and depends of the client and the right he/she purchased.

The top banner provides hyperlink to online help, contact list, frequently asked questions. It also contains the basic search box that allows the end-user to run queries on the content database.

The favourite's pane contains a list of hyperlinks the end-user has selected. The hyperlink can be:

- bookmark to some provider's resources
- link to query results
- link to external resources

The end user can perform the following actions:

- open content
- search content

- add to favourites
- remove from favourites

3.2 Content Factory of SEJER (SEJER)

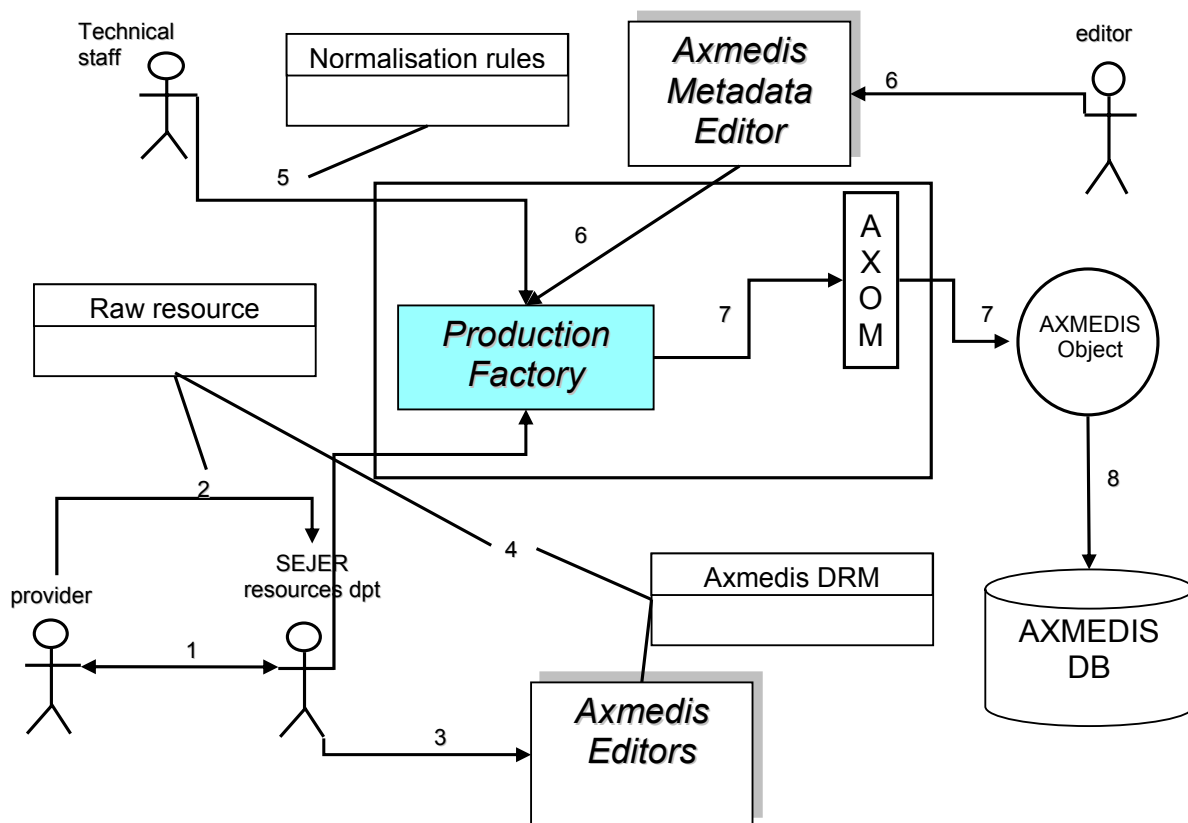
The content is always complex and heterogeneous as it is an aggregate of texts, photos, pictures, video, animations, recorded content, etc.

The several elements can be from different providers, with different Rights policies and different Business Models. For instance:

- Maps might be provided by the French “Institut Géographique National” (I.G.N.)
- Photos might be provided by press agencies or photo agencies
- Videos might be provided by the French “Institut National de l'Audiovisuel” (I.N.A.)
- Animations might be provided by other publishers
- Text excerpts might be provided by other publishers or press agencies

Most of the providers might not be AXMEDIS compliant and the right management would be manual.

Anyway, AXMEDIS TOOLS can still be used in SEJER Content Factory the following scenario presented below:



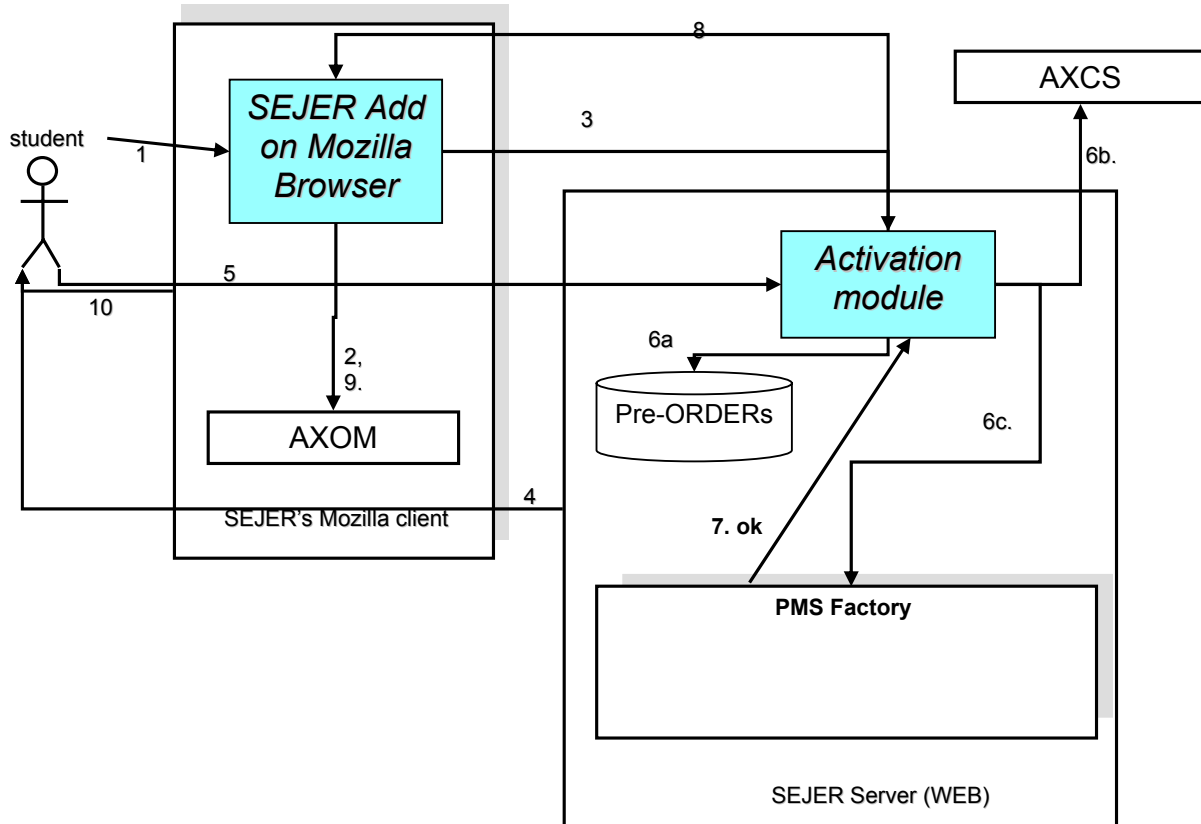
1. Negotiate Rights for one resource targeted to one product
2. Send the resource through any kind of digital channel (CD, Network etc.)
3. Transcribe, Produce the DRM Rules associated to the resource, as AXMEDIS rules (REL ?)
4. Send the Raw resource to the technical staff for further processing
5. Transform the Raw resource into a format + metadata suitable for further use in SEJER tools & products
6. Add Editorial metadata to the object if needed

7. Feed AXMEDIS editors to produce a valid AXMEDIS Object
8. Store the object in a specific DB (“external resources for integration into SEJER resources”)

The creation of the AXMEDIS through AXOM is simply a new step to add to the production automaton.

3.3 SEJER vs AXMEDIS Scenarios (SEJER)

The scenario of the protection of the content described in the Production and Protection section can be enhanced to include the AXMEDIS tools:



1. Request the object
2. Try to get the resource. Assuming answer is “Permission denied/No license”.
3. Request Activation Form to the SEJER Server. **The Add On computes Personal Identification data and send them with the request.**
4. Serve Activation Form
5. Enter Activation Number
- 6.a. Update SEJER DB and Get Info related to the requested Activation Number : product ID, DRM Rules associated to the license etc.
- 6.b. Create an AXMEDIS user corresponding to the Identification Data, get back its UID/Certificate
- 6.c. Ask for the license creation for the user registered in 6.b.
7. create the license for this object & Final User (See SC 11.1 / slide 111) And send the OK to the activation module
8. Notify client that activation has been performed, and serve the corresponding object
9. Request the license for the object just downloaded. License is now available now available
10. Display the object to the user

With the previous solution:

AXMEDIS Project

CONFIDENTIAL

- The content is not personalized to the student: license and content arrive from different paths.
- the student has taken the content online or offline, passed by the teacher or by the school.
- The student can open the content, while to use it has to be online the first time to get the license and the rights according to the chain of licenses solved by the PMS
- Once done the above point for the first time, the conquered grants and licenses can be stored in the cache into its AXMEDIS tool (PMS Client, license manager, cache manager)

It can be also chosen a different solution

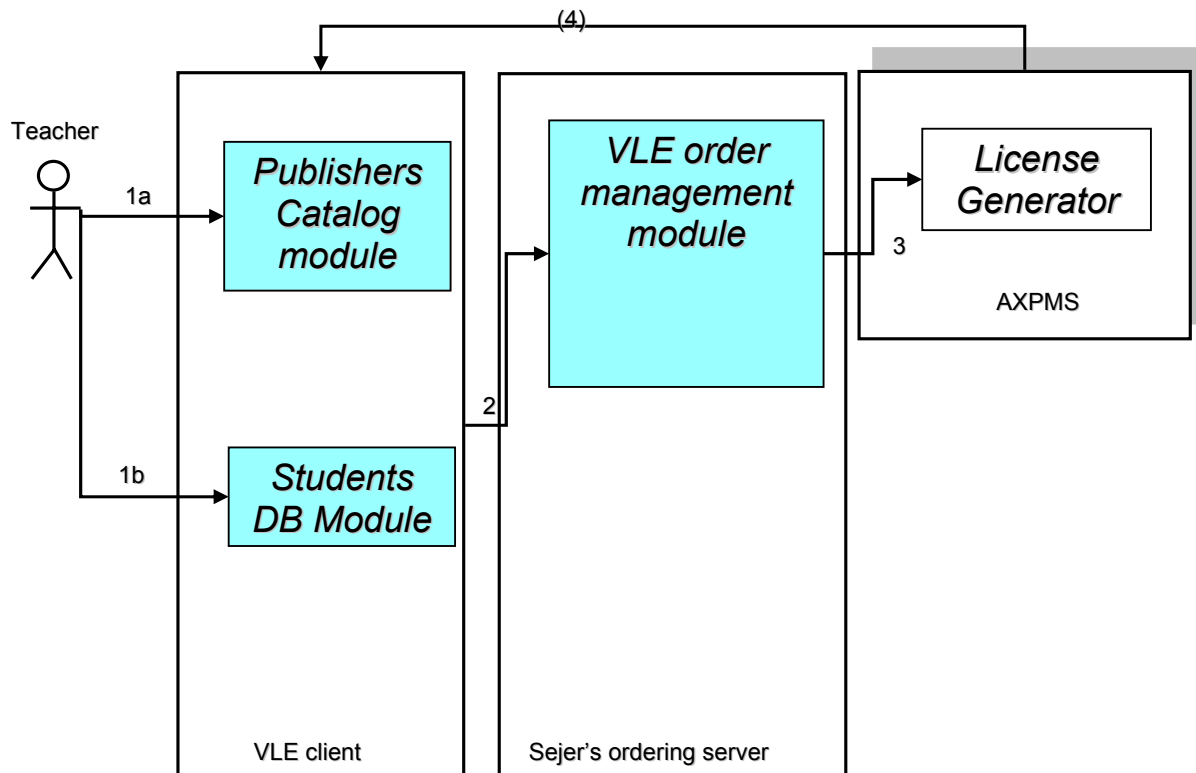
- The content is personalized to the student: license is include into the content: Governed AXMEDIS Objects.
- the user/student has taken the Governed Content online or offline, passed by the teacher/technician or by the school specifically for him.
- The user/student can open the content
 - without to be online, we can find the way to solve directly all into the Governed Object License
 - to be online the first time to solve the rights according to the chain of licenses solved by the PMS. Once done this for the first time, the conquered grants and licenses can be stored in the cache into its AXMEDIS tool (PMS Client, license manager, cache manager)

3.4 AXMEDIS tools and the integration with SEJER distribution (SEJER)

In the following scenarios, the client is the “Teacher”, the end-users are the students, and the VLE is the web portal. The user, whether the client or the end user, sign on the portal only once.

3.4.1 Ordering the content

When the teacher wants to order some content for his/hers students, he /both have to select the content and the students. All those information are combined in a request to the order management system.



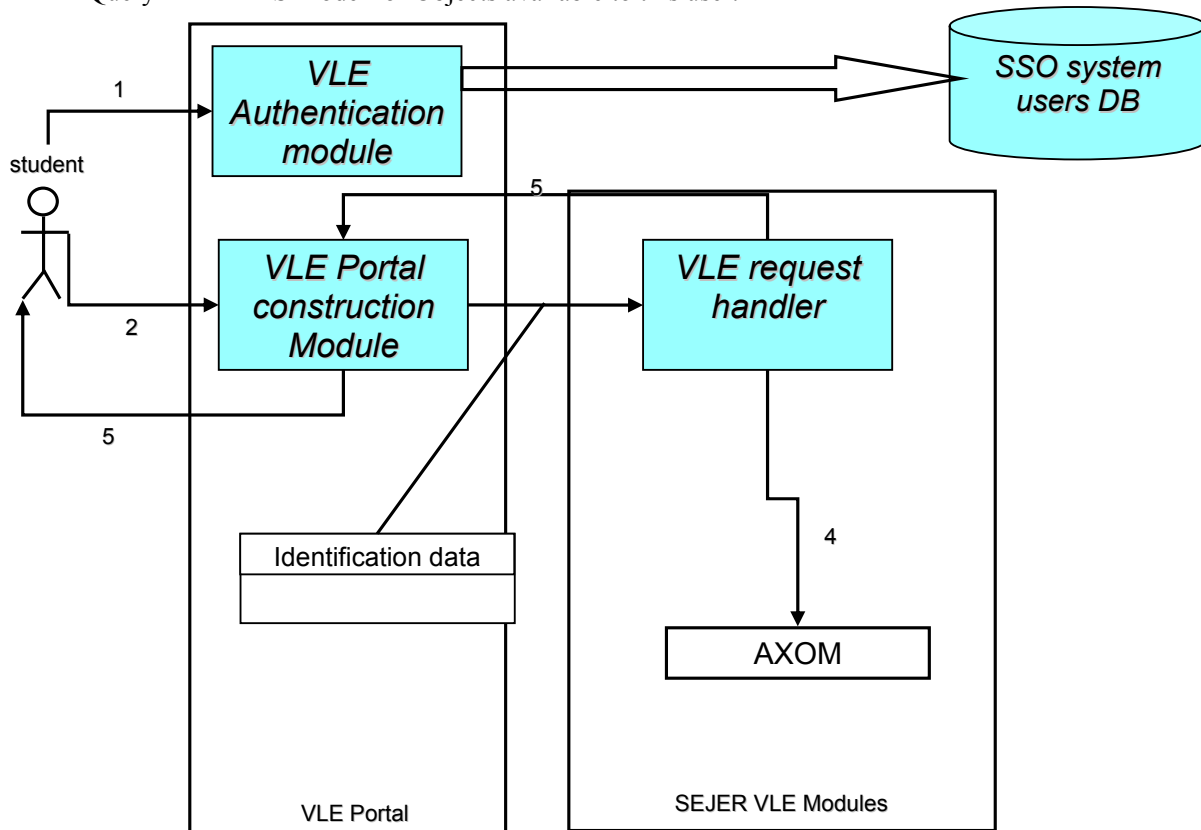
1a. Teacher select the product he wants to order

- 1b. Teacher select the school class or any list of student he wants to order to
2. Send the order to the publisher server, along with ID's of the future license owners (ordering server manages payment)[**Ids are IDS of the students in the VLE system. Need a way to map them into AXMEDIS UIDS**]
3. Assuming Payment has been done : Create N License for the N Users and store them.
4. possibly, licenses are sent back in the School Level VLE system directly ?

3.4.2 Consulting the content

When the student wants to consult the content that has been made available for him/her, he first connect to the VLE using his/hers login and password. The portal then display all the content the student has access to, so he/she can select the one he/she wants to consult. The available content list is gathered through several channels :

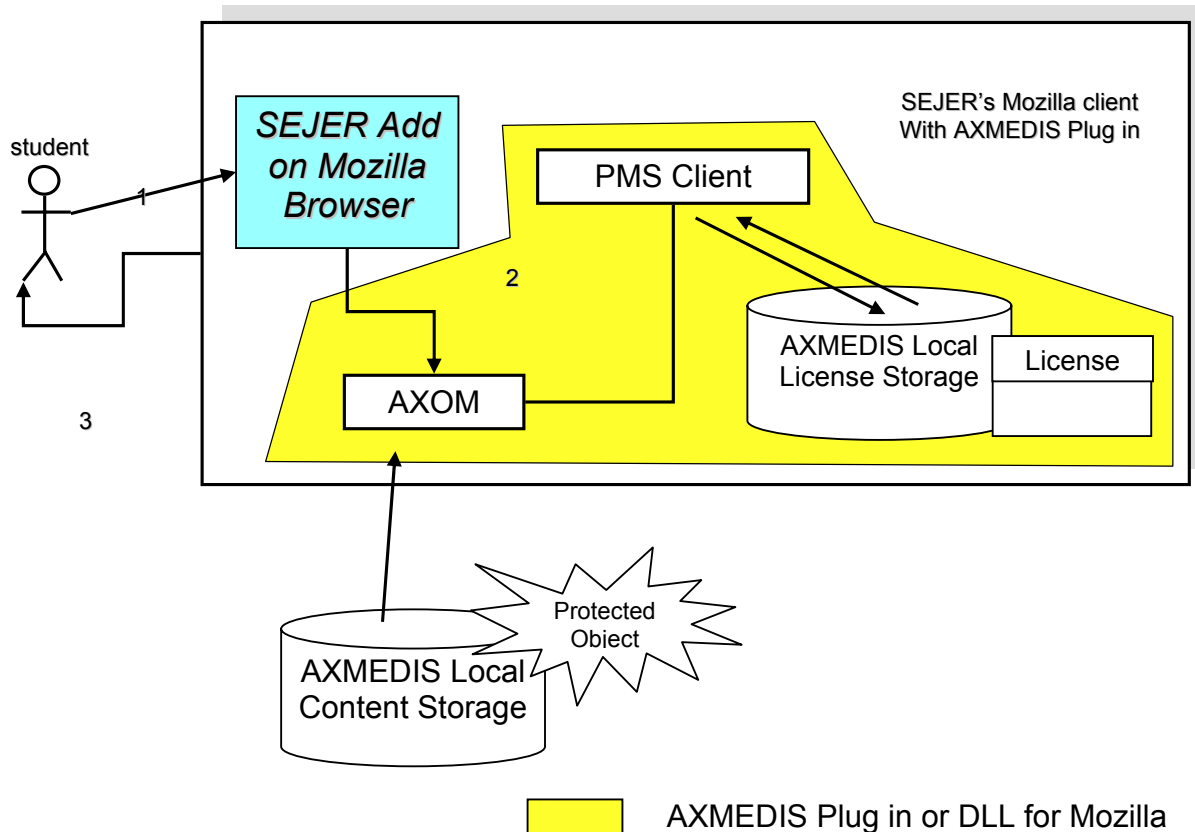
- List the references from affiliated content providers (on distant servers)
- Query AXMEDIS Model for Objects available to this user.



1. Authenticate through the SSO system
2. Request available content
3. Gather objects references from affiliated content providers (on distant servers)
4. Query AXMEDIS Model for Objects available to this user in this VLE
5. Send back available content list suitable for the portal
6. Display available resources list

3.5 SEJER player with AXMEDIS support (SEJER)

The diagram:below is focused on the how the Add on uses AXMEDIS tools.

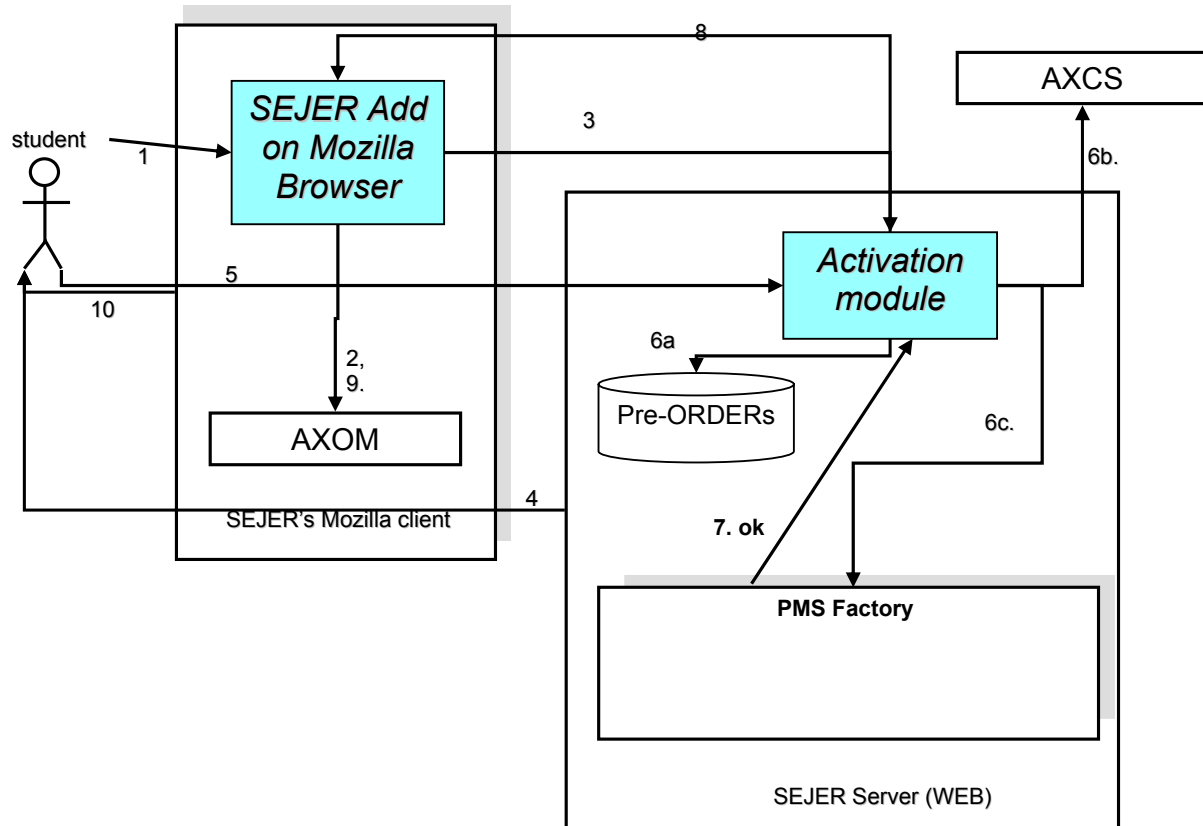


1. Request URL of the resource (through web portal – specific protocol name “nsxa://”)
2. Associate profile information (Device ID + computed User identification data), request object + license
3. Assuming License Parameters does not request connection for update/check : display content

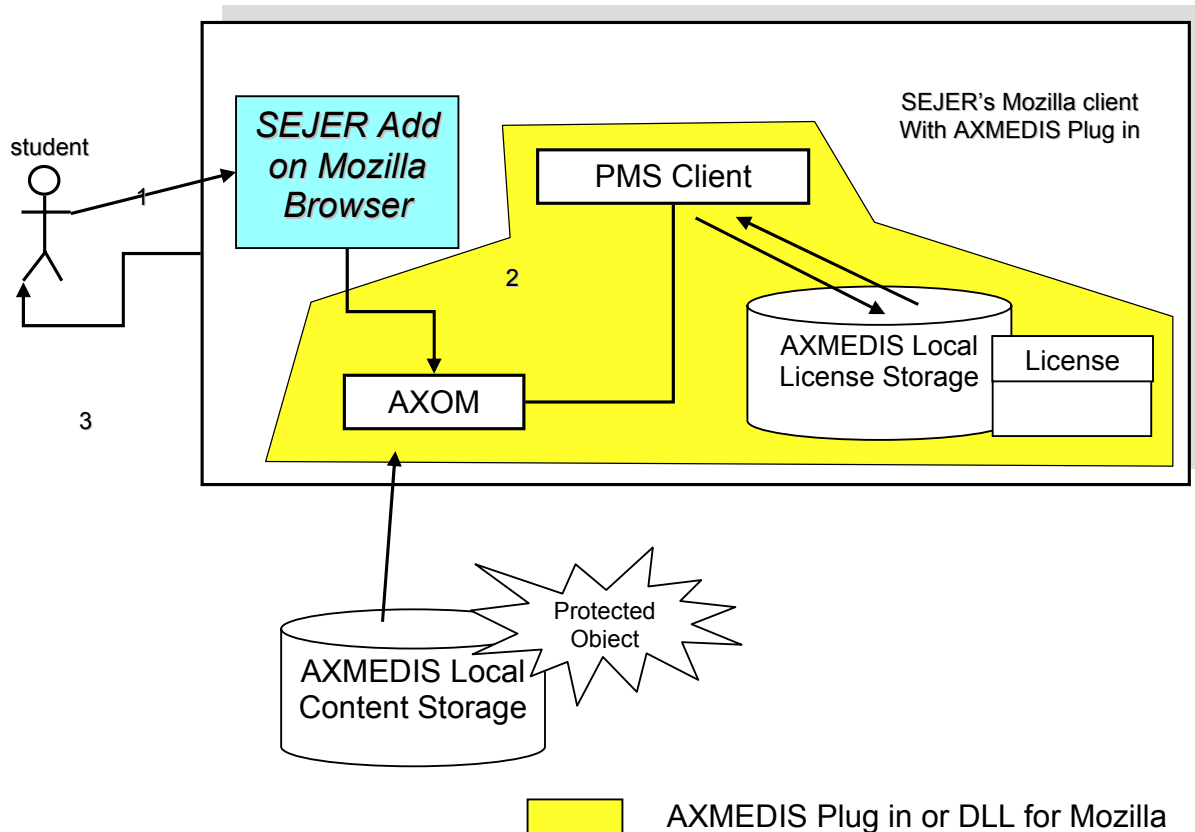
The only requirements for displaying AXMEDIS content are:

- Addressing it with and URL starting by the specific protocol name
- Having installed on the computer a plug in for Mozilla that is able to display the content, as if the URL with http://

The add on is an XPCOM component that can be installed via an XPI , Mozilla embedded installer component. The XPCOM technology is used to interface Mozilla and C code.



1. Request the object
2. Try to get the resource. Assuming answer is "Permission denied/No license".
3. Request Activation Form to the SEJER Server. **The Add On computes Personal Identification data and send them with the request.**
4. Serve Activation Form
5. Enter Activation Number
- 6.a. Update SEJER DB and Get Info related to the requested Activation Number : product ID, DRM Rules associated to the license etc.
- 6.b. Create an AXMEDIS user corresponding to the Identification Data, get back its UID/Certificate
- 6.c. Ask for the license creation for the user registered in 6.b.
7. create the license for this object & Final User (See SC 11.1 / slide 111) And send the OK to the activation module
8. Notify client that activation has been performed, and serve the corresponding object
9. Request the license for the object just downloaded. License is now available now available
10. Display the object to the user



2. Request URL of the resource (through web portal – specific protocol name “nsxa://”)
3. Associate profile information (Device ID + computed User identification data), request object + license
4. Assuming License Parameters does not request connection for update/check : display content

The only requirements for displaying AXMEDIS content are:

- Addressing it with and URL starting by the specific protocol name
- Having installed on the computer a plug in for Mozilla that is able to display the content, as if the URL with http://

4 Distribution towards Mobile (WP4.7, WP9.5: COMVERSE)

4.1 Architecture Territories:

The AXMEDIS enabled COMVERSE distribution system defines two main territories.

- The first and primary territory is *Distribution*. The goal for this territory is to integrate the existing COMVERSE distribution system with the AXMEDIS platform, thus enabling the browsing and consumption of AXMEDIS content offering via the COMVERSE system.
- The second territory is *Transcoding*. The goal for this territory is to automatically adapt content that is offered over the AXMEDIS network for consumption by mobile phone users through the integrated COMVERSE distribution system.

The territories share some of the modules and data. However, the architecture of each territory is independent of the other.

4.2 Common System Guidelines

4.2.1 System Interfacing Protocol

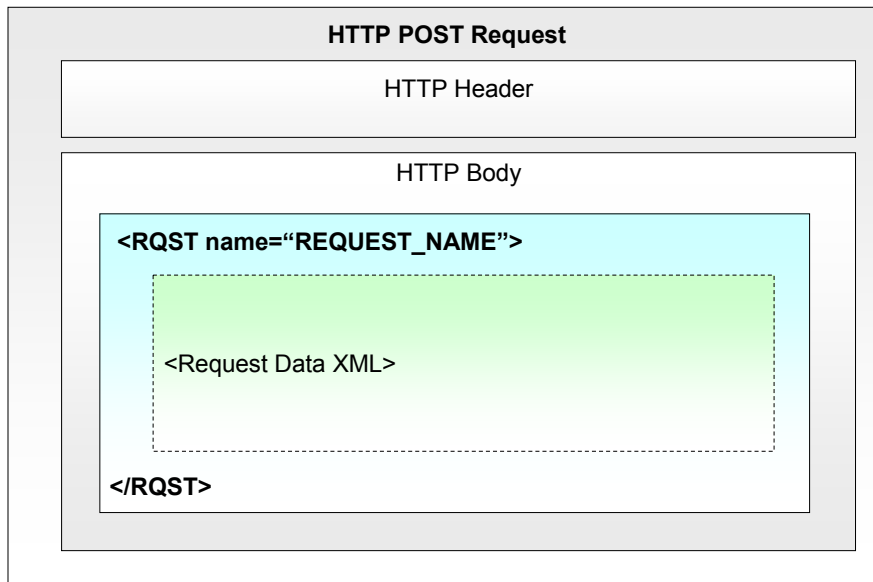
The interface between the sub systems shall rely on XML over HTTP, unless explicitly specified otherwise. All requests shall be made using the HTTP POST method. The request message XML shall be sent as the body of the HTTP POST request, wrapped in a standard COMVERSE RQST envelope. (See **Standard Request Structure Figure**)

The response message XML shall be sent as the body of the HTTP response, wrapped in a standard COMVERSE RPLY envelope. (See **Standard Reply Structure Figure**)

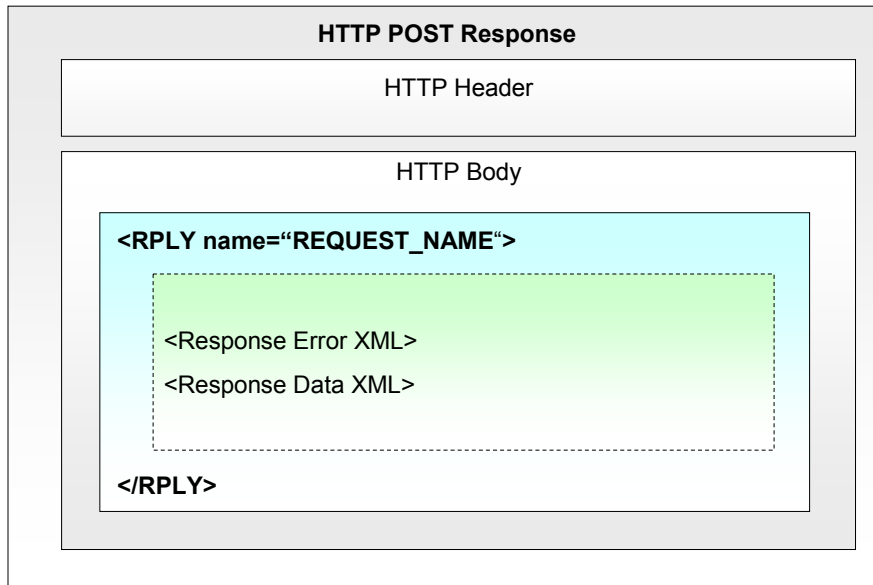
The Response message may include an XML Error block. (See **Reply Errors – Detailed Structure**)

Both request and response messages shall be formatted according to the specific request/reply XML schema.

Standard Request Structure Figure



Standard Response Structure Figure



Reply Errors – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	ERROR	Element		Error item (root)
ERROR	id	Attribute	1	The unique error ID
	description	Attribute	1	A short description of the error.
	DETAILS	Element	0..1	A container for a detailed error description
	ERRORS	Element	1	Collection of sub error items
DETAILS	text	text	1	A detailed description of the error, the cause, a directive for the user, etc
ERRORS	ERROR	Element	0..n	Error item (nesting)

Example: Error with details:

```
<ERROR
  id="HME2045"
  description="Bad request syntax">
  <DETAILS>
    The request could not be processed because it does not comply with the
    HMS REQUEST XML schema. Please refer to development documentation.
    If you are unable to follow this directive please contact the system administrator.
  </DETAILS>
</ERROR>
```

Example: Nesting Errors:

```
<ERROR
  id="HMS0012"
  description="The requested HMS Service is unavailable">
  <ERRORS>
    <ERROR
      id="HMS1590"
      description="Unable to create HME interface object">
```

</ERRORS>
</ERROR>

4.2.2 System Implementation and Deployment Platforms

Executable and Library modules shall be developed in C++ programming language. Unless explicitly specified otherwise. The modules shall be built to be executed and run on Microsoft Windows 2000 or Windows Server 2003.

Web API modules shall be developed as sets of web pages. The pages shall make use of the ASP technology for server-side scripting. The modules shall be deployed on Microsoft IIS5 web servers.

Each page shall make a single public function of the performer/server application available to the client applications.

4.3 Distribution Territory

4.3.1 General Architecture

The distribution territory incorporates several sub systems – entities that shall specialize to cover a specific field of functionality that is required for the integration of the COMVERSE distribution-to-mobile system with the AXMEIDS platform. Most of the sub systems are new designs to be developed, and shall include one or more modules.

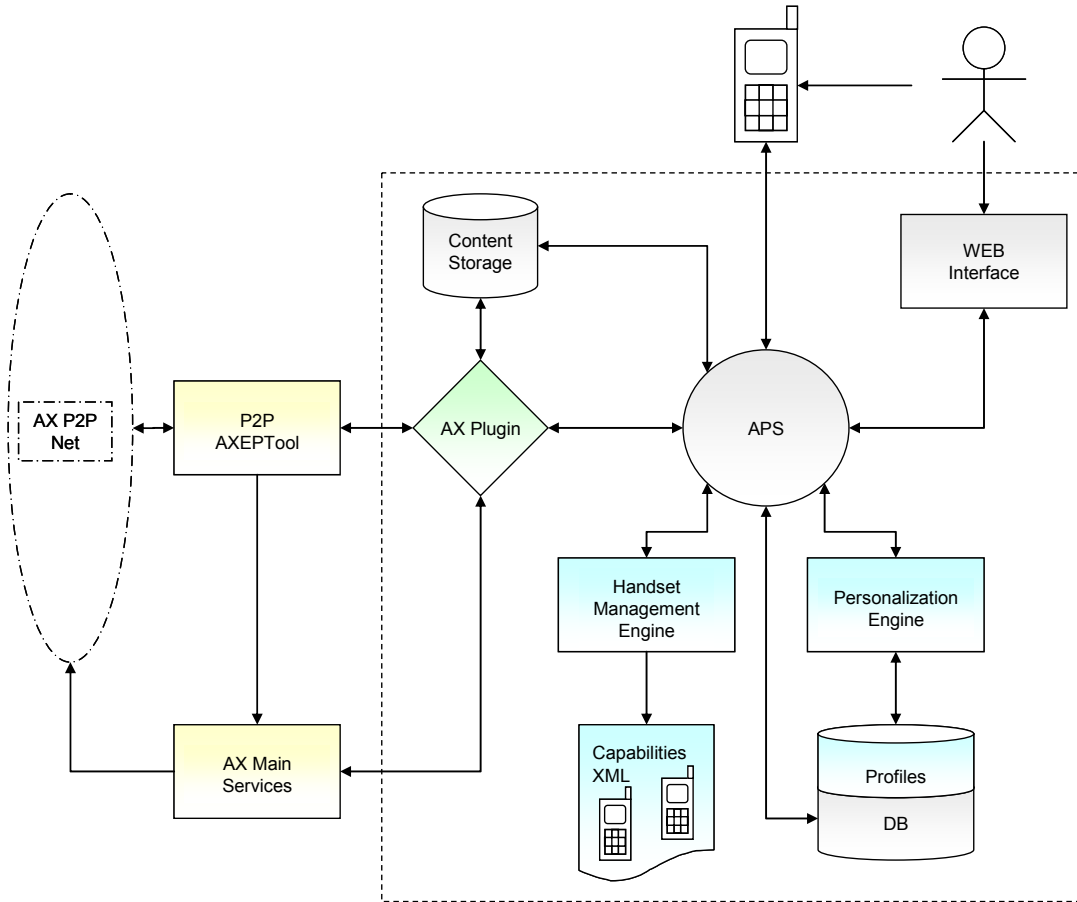
(See **Distribution Territory – Architecture Diagram**)

4.3.2 Sub Systems Incorporated

The sub systems that combine to create the distribution-to-mobile system are:

8. **APS** – COMVERSE *Application Server* – The existing COMVERSE Distribution Engine
9. **HMS** – COMVERSE *Handset Management Service* – A new system that is required for filtering content offerings by the capabilities of the consumer mobile device
10. **PS** – COMVERSE *Personalization Service* – A new system that is required for optimizing and personalizing the experience of browsing the vast amount of content offerings
11. **AX Plugin** – A new system that shall intermediate between the distribution system and the AXMEDIS platform, enabling and encapsulating AXMEDIS operations; retrieve catalogs, search for content, download media, permissions, etc
12. **AXEPTool** – An AXMEDIS system that is required for the database, search and storage services it offers

Distribution Territory – Architecture Diagram



4.3.3 Actors

The distribution system serves and is driven by a single actor – the Subscriber – a mobile phone owner and a multimedia consumer, who uses the various system user interfaces to initiate and navigate the system scenarios. The subscriber is registered in the system, and can be uniquely identified.

4.3.4 Main Scenarios

There are a few cross system scenarios that dominate the activity of the distribution system. They are:

4.3.4.1 Get Catalogue (Category Tree) Scenario

TRIGGER: This scenario shall occur when the APS is Started, Restarted, and when the APS shall receive an explicit request to refresh the catalog.

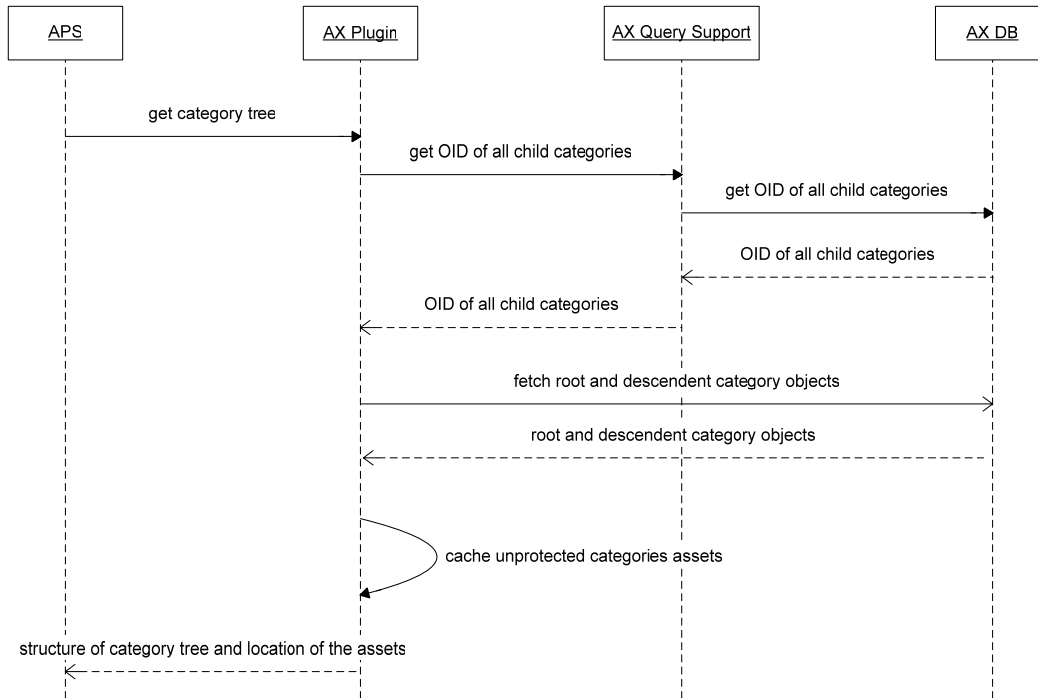
FUNCTION: The APS shall retrieve the Publisher's content catalog from the AXMEDIS network, through the AX Plugin.

FLOW: The APS shall request the catalog from the AX Plugin. The Plugin then shall retrieve the catalog Category Tree; For each AXMEDIS Catalog Category object the AX Plugin shall retrieve the category assets (narration audio, menu title text, description, meta data, etc), category content selection (Attached AX Selection), and all references to child categories. When the categories are downloaded the Plugin shall unprotect, unpack, and store the category assets in the system cache, as files, accessible to the APS. When the entire Category Tree has downloaded, and all the assets are cached, the Plugin shall respond to the APS

with the structure and info of the category tree, the attached info, and the location of the cached assets for each category, in the format that the APS knows.

VARIATION: None. (See *Get Catalogue (Category Tree) Scenario – Sequence Diagram*)

Distribution System – Get Catalogue (Category Tree) Scenario – Sequence Diagram



4.3.4.2 Get Category Content Offering, Personalized and Filtered by Device-Capability Scenario

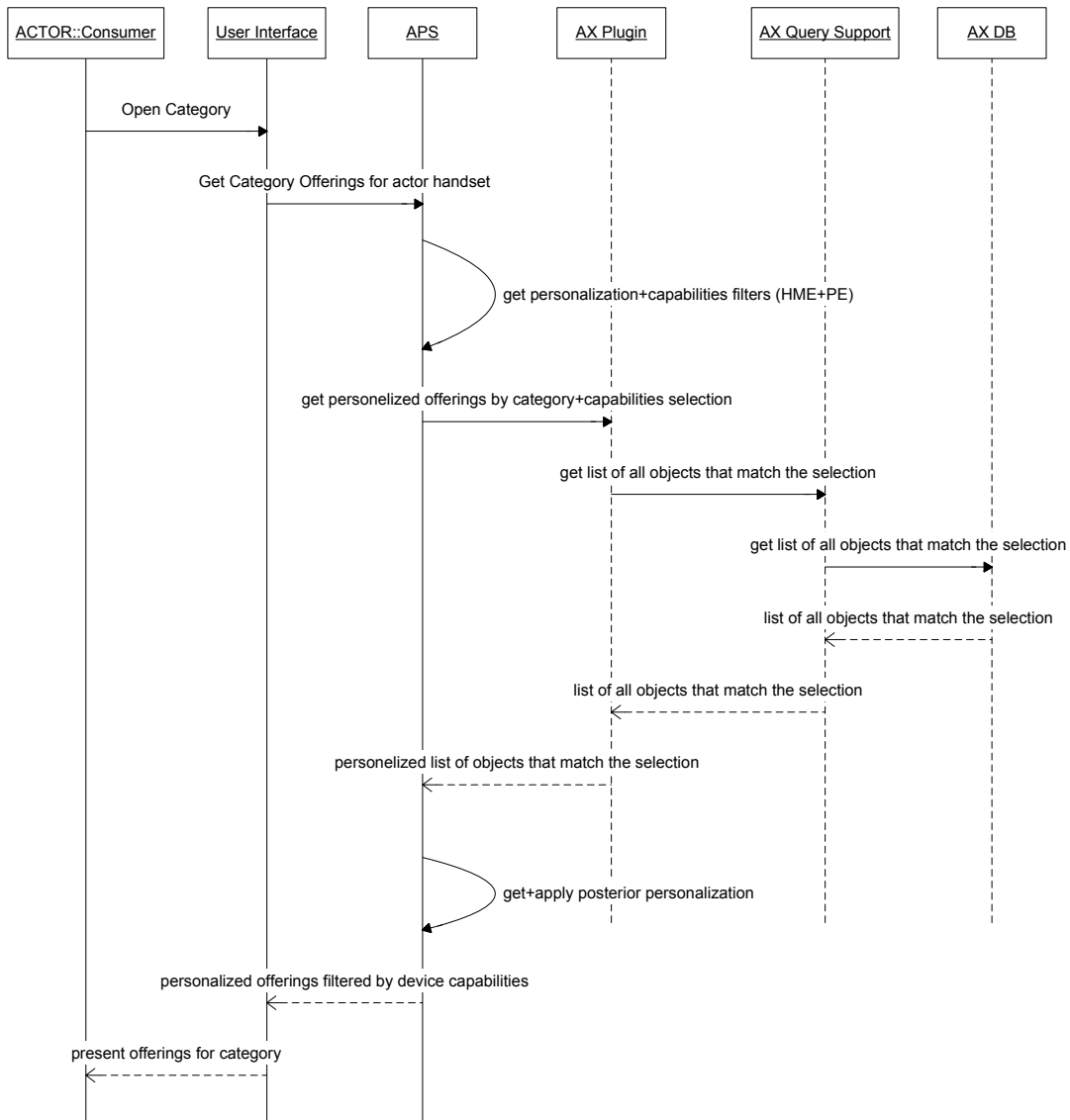
TRIGGER: This scenario shall occur when the Actor requests to see the list of content that is offered in a specific Category.

FUNCTION: The APS shall retrieve the list of content items that are offered in the category, and match the capabilities of the Actor's device, from the AXMEDIS network, personalized to the Actor's preferences.

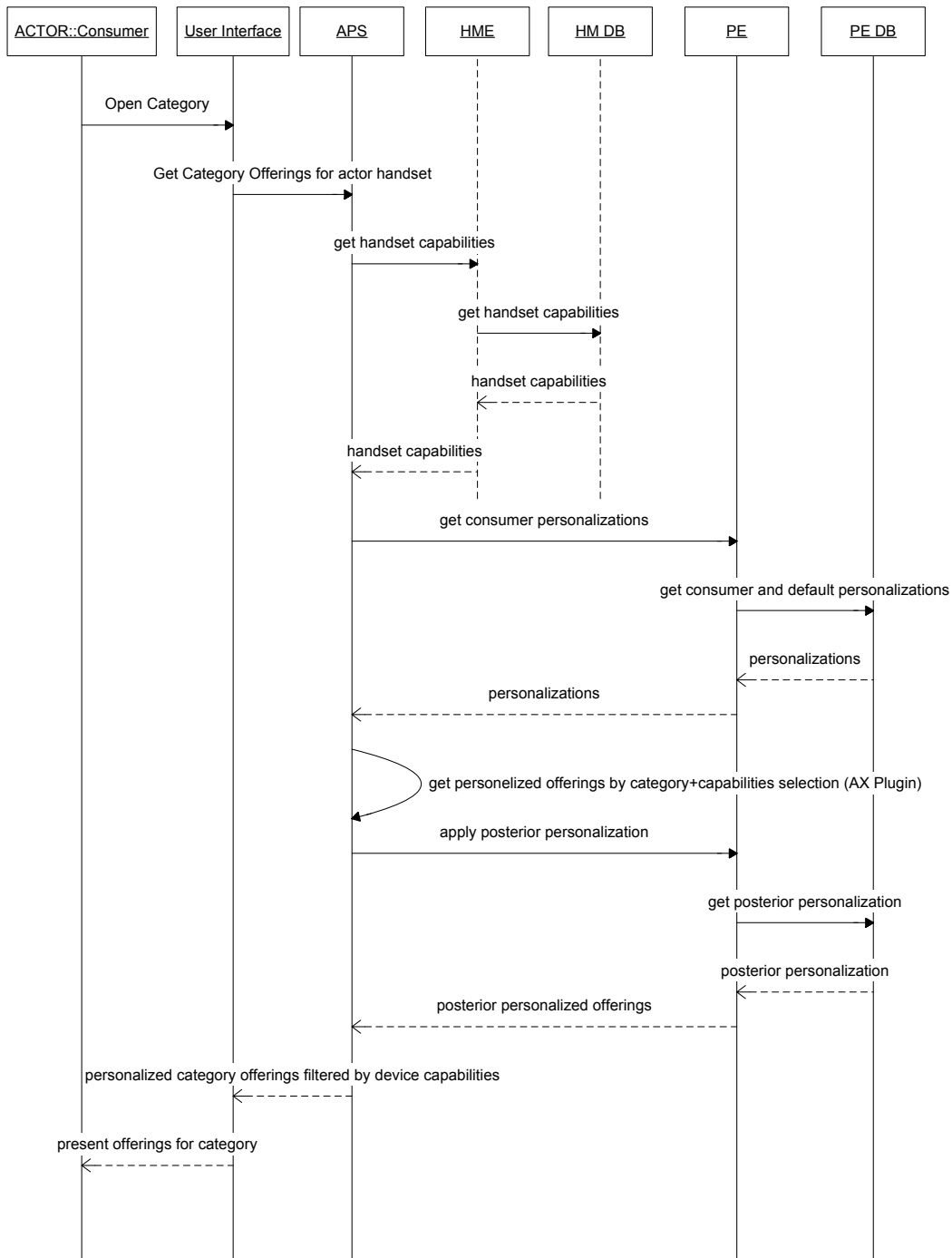
FLOW: The Actor shall request the APS, through the UI, to list the content offered in a specific category. The APS shall request Actor's device capabilities from the HMS, and the Actor's preferences from the PS. (See *Personalize Catalogue by Actor Preferences and Device Capabilities Scenario – Sequence Diagram*) The APS shall then request the list or content for the category, filtered by the device capabilities, and organized to the Actor's preferences, from the AX Plugin. The AX Plugin shall apply the APS query to the AX Query Support, and respond to the APS with the query result. The APS shall then apply any post-request personalization, and respond to the UI with desired information.. (See *Get Category Content Offering Scenario – Sequence Diagram*)

VARIATION: The APS may already have the personalized query, and query result cached. In this case the APS shall not send the request to the AX Plugin.

Distribution System – Get Category Content Offering Scenario – Sequence Diagram



Distribution System – Personalize Catalogue by Actor Preferences and Device Capabilities Scenario – Sequence Diagram



4.3.4.3 Sample Content Scenario

TRIGGER: This scenario shall occur when the Actor requests to sample a content item.

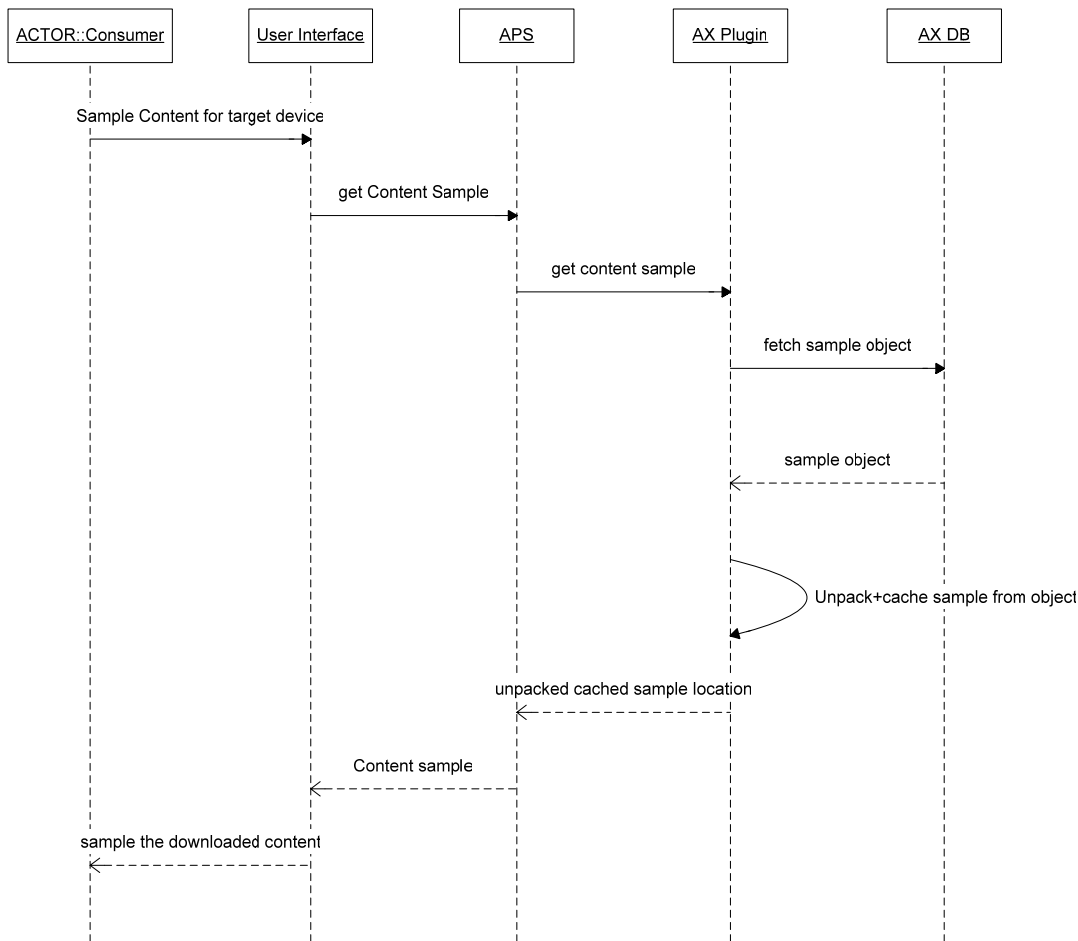
FUNCTION: The APS shall retrieve the sample file from the AXMEDIS network, and push it to the UI.

FLOW: The Actor shall request to sample a content item on a Device/UI. The APS shall request the AX Plugin to retrieve the content sample file. The AX Plugin shall retrieve the object from the AX DB,

unprotect and unpack it to the system cache, accessible to the APS. When the file is cached, the AX Plugin shall send a response to the APS with the location of the cached sample file. The APS shall then push the sample file to the UI. (See **Sample Content Scenario – Sequence Diagram**).

VARIATION: The Sample file may already be cached. In this case the APS shall not request the file from the AX Plugin.

Distribution System – Sample Content Scenario – Sequence Diagram



4.3.4.4 Buy and Push Content Scenario

TRIGGER: This scenario shall occur when the Actor requests to purchase a content item.

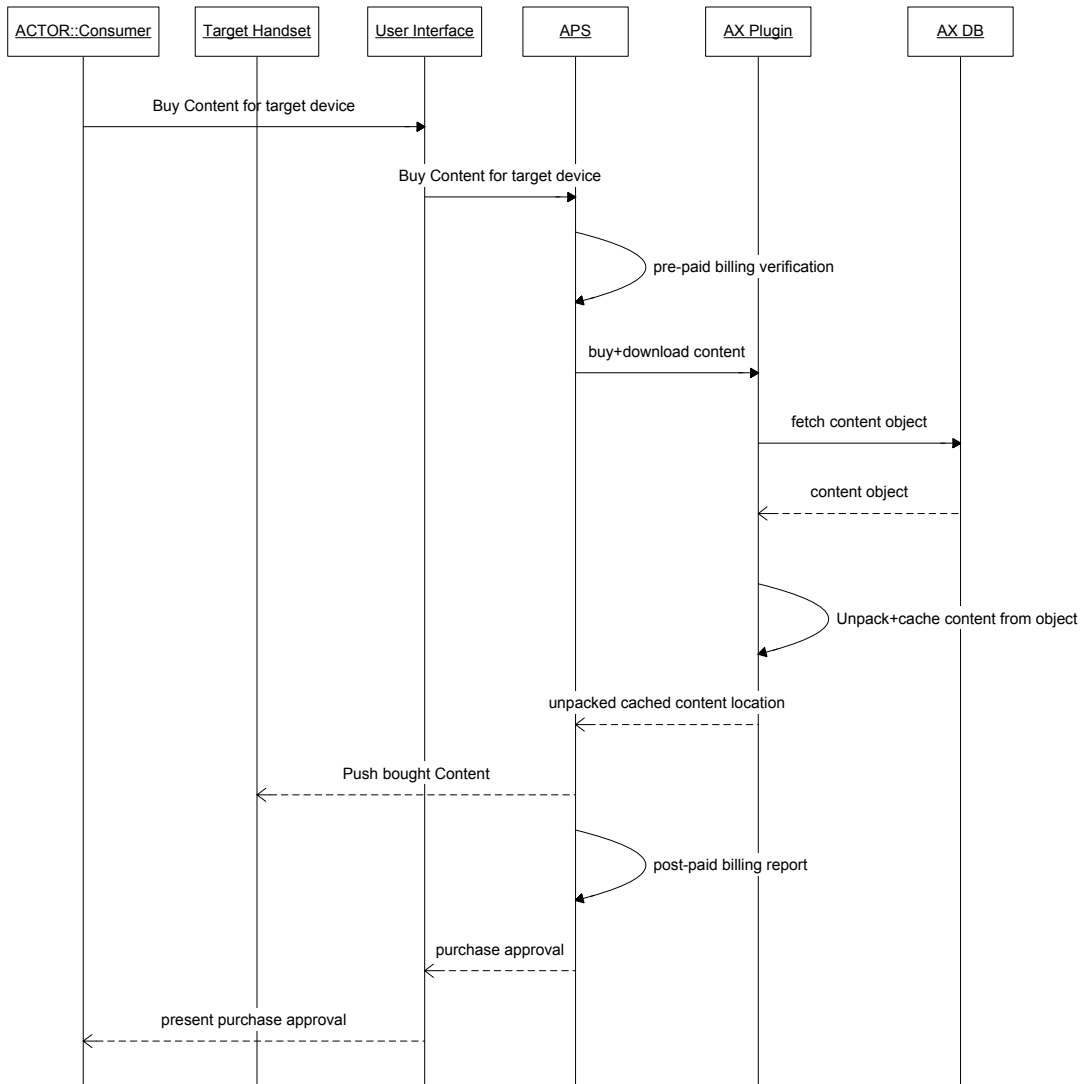
FUNCTION: The APS shall retrieve the content item from the AXMEDIS network and push it to the destination device.

FLOW: The APS shall apply Pre-Paid applications on the billing service (not an AXMEDIS service) to verify the Actor's credit. The APS shall then request the desired content from AX Plugin. The AX Plugin shall download the content AX object, unprotect and unpack it to the system cache, accessible to the APS. The APS shall then push the content to the destination device, apply Post-Paid applications on the billing service, and send the purchase approval to the UI. (See **Buy and Push Content Scenario – Sequence Diagram**)

VARIATION: The desired content item may already be cached. In this case the APS shall no request the item from the AX Plugin.

VARIATION: Different configurations may require Pre-Paid applications or Post-Paid applications, or both.

Distribution System – Buy and Push Content Scenario – Sequence Diagram



4.3.5 Data Structure

4.3.5.1 Content Domains

The system shall handle the following data domains:

1. Content Catalogue; A multimedia items catalogue that is structured as a tree of Category objects, where each category shall have a single Category parent, and may have one, none, or many child Categories. Each Category can reference one, none, or many multimedia items, i.e. content items.

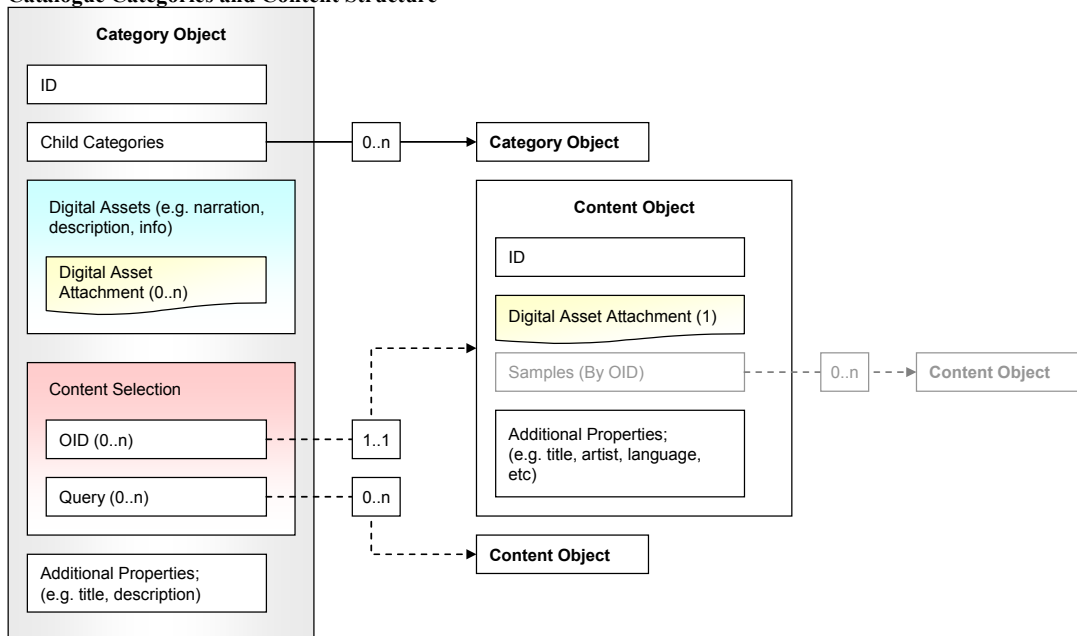
(See COMVERSE Category and Categories collection – Detailed Structure)

2. Content Items; Multimedia items, each containing multimedia (audio, video, graphics, text, etc) attachment that shall be distributed to the end consumer for sampling and consumption. Each multimedia content item shall include a primary multimedia attachment and a set of properties that identify and classify it. (See COMVERSE Multimedia Item and Media Items collection – Detailed Structure)
3. Mobile Device Capabilities; A collection of mobile devices that shall specify for each device its capabilities to download and handle multimedia items. The specifications may include, but not be limited to, supported multimedia formats, display resolution, audio capabilities, download protocols and methods, hyper text versions, etc. (See Device Capabilities Data – Detailed Structure)
4. Subscriber Personalization Preferences; A specific set of preferences that can be applied to the catalogue Categories and Media Items, such as Sort field, Sort Order, Most Recent Items, Wish List, etc. The system shall save, load and apply the personal preferences of each subscriber. (See Personalization Preferences - Detailed Structure)

4.3.5.2 Primary Objects

The integration of the systems shall handle and mobilize the two main data objects: Catalog Category and Media Item. The objects are primarily COMVERSE distribution system objects, and shall be reflected in the AXMEDIS platform. (See AX Catalogue Categories and Content Structure)

Catalogue Categories and Content Structure



4.3.5.3 Data Origins, Adaptation and Transformation

The distribution territory integrates the COMVERSE system and the AXMEDIS platform - two separate and independent systems, each generating and handling very different sets of data and structures.

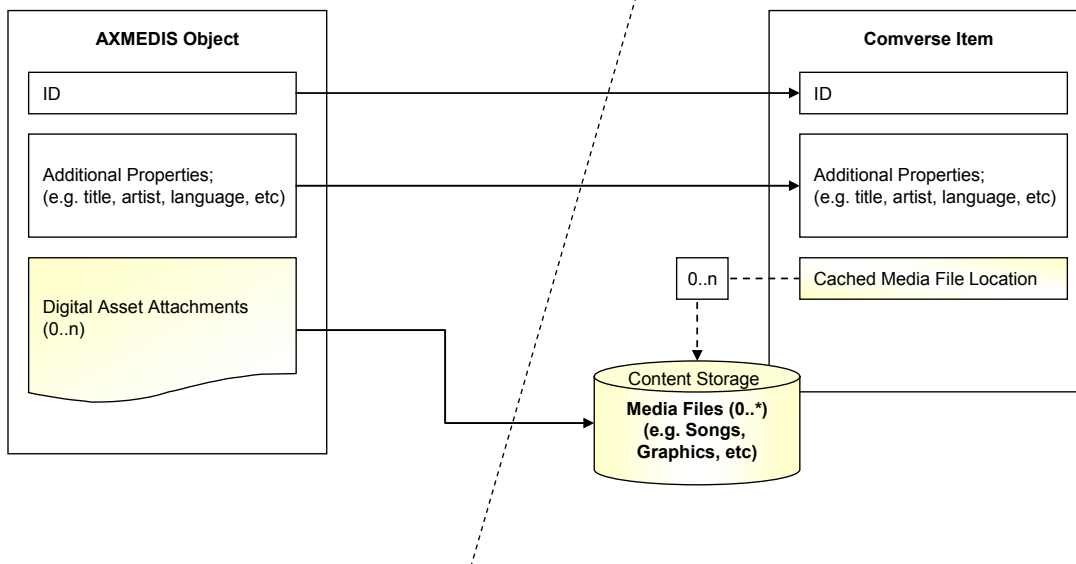
The integrated system will be engaged in scenarios that shall transport data, such as the Catalogue, Content, Device Capabilities, etc, from one independent system to the other. Each system shall also store references to information from each other.

To enable this, the AX Plugin, which shall function as mediator between the two systems, shall transform the data on the fly, and adapt it to the service that is desired on the system across.

Basic data adaptation guidelines:

The following adaptations rules shall apply if not explicitly specified otherwise.

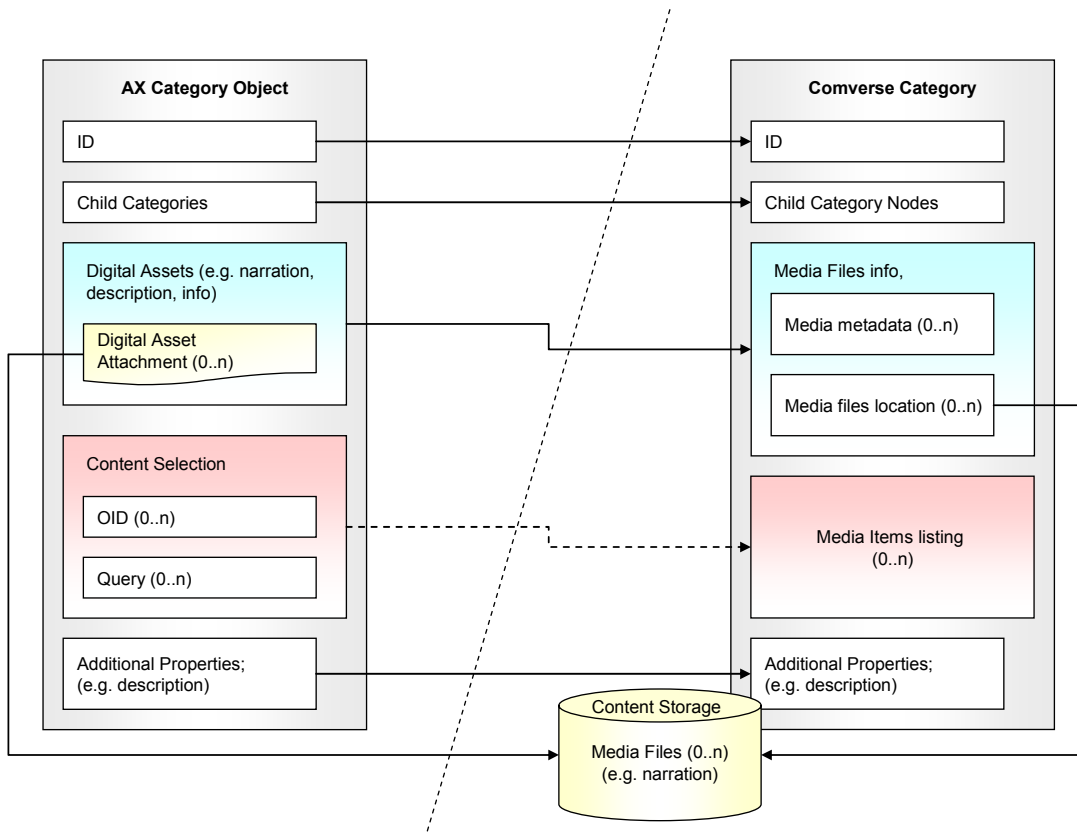
1. COMVERSE and AXMEDIS Object data, properties and values shall be described according in XML schema.
2. Digital assets contained (attached, not referenced) in AXMEDIS objects shall be unprotected, unpacked, and cached in a storage that is accessible to the COMVERSE applications. (See **Data Transformation – General AX Object to COMVERSE Item Map**). The XML description of the object shall reference the cached file in a FILE tag. (See **COMVERSE File and Files Collection – Detailed Structure**) COMVERSE objects referencing multimedia files shall be packed to AXMEDIS objects with the files – digital assets – attached.
3. Data that cannot be mapped to the object XML schema shall be added to a META_INFO block. (See **COMVERSE Meta Info – Detailed Structure**)

Data Transformation – General AX Object to COMVERSE Item Map**Data transformation within scenarios:**

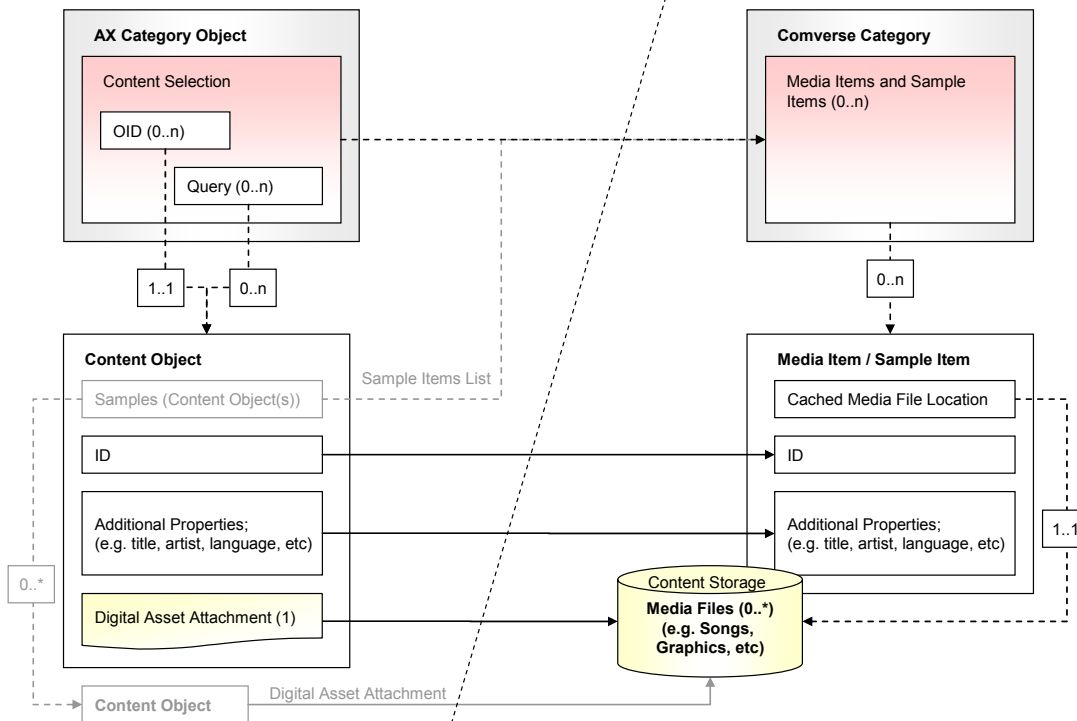
Data transformations that shall be performed with the progress the system scenarios:

1. Get Catalogue (Category Tree) Scenario
 - a. The Plugin retrieves the entire Category tree from the AXMEDIS network, to the APS:
 - i. The AX Plugin shall describe the entire structure of the Category Tree in XML. (See **COMVERSE Category and Categories collection – Detailed Structure**)
 - ii. The AX Plugin shall cache the digital assets of all categories, as files that are accessible to the APS. The category tree description shall include references to these files. (See **Data Transformation - AX Category to COMVERSE Category Map**)

Data Transformation - AX Category to COMVERSE Category Map



Data Transformation - AX Category Content Selection to COMVERSE Category Media Items Map



2. Get Category Content Offering, Personalized and Filtered by Device-Capability Scenario
 - a. The APS Queries for the content that is offered in a Category:
 - i. Device Capabilities are translated to content-properties criteria (Content type, resolution, etc)
 - ii. Personalization Preferences are translated to content-organization criteria (order by, etc)
 - iii. The category's Content Selection – the definition of the content to list under the category – is added with the content-properties and content-organization criteria before being sent to the Query Support.
 - b. Query Support result sent as the response to the Query for Category content offering:
 - i. Selection result from Query Support is returned as a Media Items XML. (See **AX Category Content Selection to COMVERSE Category Media Items**)
3. Sample Content Scenario
 - a. The AX Plugin shall cache the digital asset of the Sample object as a file that is accessible to the APS.
4. Buy and Push Content Scenario
 - a. The AX Plugin shall cache the digital asset of the Content object as a file that is accessible to the APS.

4.3.6 Sub Systems – Detailed Specifications

4.3.6.1 COMVERSE APS

Overview – Existing Module and Architecture Not to be Modified

The APS is the primary and most complex component of the COMVERSE distribution system. It maintains the system logic existing and is architecturally in the center of the system. It is the APS that shall be integrated with the AXMEDIS platform. The architecture of the APS cannot and must not be modified, and as little code modifications shall be applied to this component.

Functionality – Functionality and Capabilities Relevant to Integration

The relevant APS functionality and capabilities for the integration of the APS with AXMEDIS:

1. Retrieval of the Catalog Category Tree
The Integrated APS shall retrieve the entire category tree from the AX Plugin, in a single request.
2. Retrieval of the Media Items for each Category, Personalized to the subscriber preferences, and filtered by the destination device capabilities
The APS shall request the list of Media Items that are offered in each category, organized to the subscriber preferences and filtered by the capabilities of the destination device. To compose the query, the APS shall add the personalization preferences of the subscriber and destination device capabilities (retrieved from HMS and PS), and add it to the content Selection (AX Selection) definition that is set for the Category. The query shall be requested from and responded by the AX Plugin.
3. Fetching content sample and pushing it to the subscriber UI
The APS shall fetch the file from the AX Plugin and push it to the UI.
4. Fetching desired (for purchase) content and pushing it to the subscriber mobile The APS shall fetch the file from the AX Plugin, perform pre/post-paid applications, and push the content to the destination device.

5. Caching of Media Item Queries

The APS shall cache queries that are required for retrieving the list of multimedia items offered in a specific category, including the device capabilities and personalization criteria.

6. Caching of Media Item Query Results

The APS shall cache the results of queries that are required for retrieving the list of multimedia items that are offered in a specific category.

7. Caching of Media Item Info

The APS can cache the extended information and meta-data that is retrieved for each multimedia item.

(See Main Scenarios for detailed scenario description)

Configuration – Settings Relevant for Integration:

The APS shall be configured to:

1. Interact with the AXMEDIS platform as the source for Catalogue and Media Items.
2. Cache queries
3. Cache query results
4. Enable Device Capabilities Identifications
5. Enable Subscriber Personalization

4.3.6.2 Handset Management Service

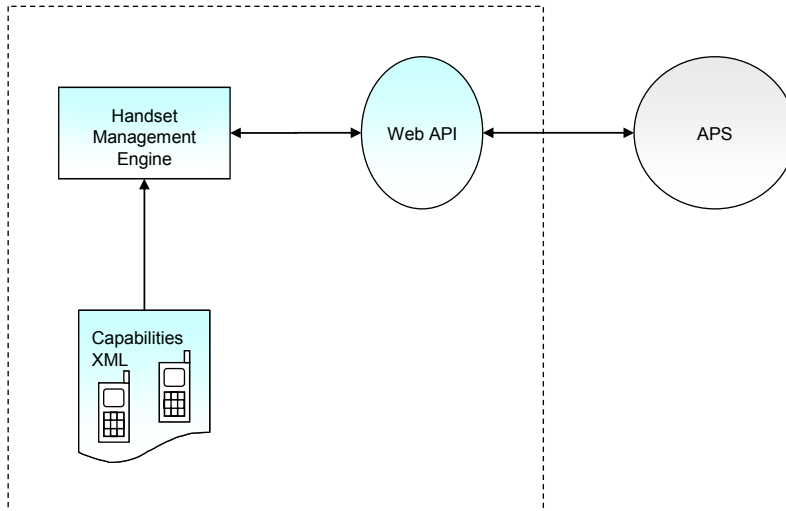
Overview

The HMS shall allow the COMVERSE Distribution-to-Mobile system to identify the capabilities of mobile phone devices to handle multimedia content and how to deliver it.

Architecture:

The HMS shall incorporate (a) a core HME module, which shall perform all the required operations, and (b) a Web API layer, which shall serve as the main interface to the HME for client applications. The interaction between client applications and the HMS, through the API shall rely on XML over HTTP(S).

HMS – Architecture Diagram



Module – HME (Handset Management Engine):

Classification:

The HME shall be developed as a stand alone, multi-threaded, executable.

Implementation Platform:

Common executable implementation platform (See **Common Guidelines, Implementation Platform**)

Functionality:

The HME shall perform the following functionalities:

1. Load configuration;
 TRIGGER: on application start and when Restarted.
 FUNCTION: The HME shall load the configuration file and apply the settings. Throughout the operation the HME shall not accept any requests made from client applications.
2. Load Device Capabilities Data;
 TRIGGER: on application start and when Restarted.
 FUNCTION: The HME shall load the device capabilities data from the resources to memory. Throughout the operation the HME shall not accept any requests made from client applications.
3. Find Device Capabilities by Device Identifiers;
 TRIGGER: On request from a client application, through the HME interface.
 FUNCTION: The HME shall search the device capabilities data that is resident in memory for a specific or fallback device, by the device identifies that were provided by the calling application (*see HME interface*). When found, the device capabilities list shall be returned to the calling application.
 (*Figure: HME – Get Device Capabilities Request*)

OPERATION: The search for the device shall iterate through the device attributes that were provided by the calling information, beginning with the most specific (i.e. ID) and ending with the least specific (i.e. Manufacturer). If iteration yields a device match, then the HME shall move to check if a device-fallback is required. If the HME shall fail to find device, iterating with all provided device parameters, than it shall fallback to the default device (defined in the configuration file).
 (*Figure: HME – Finding Device and Fallback*)

HMS – Get Device Capabilities Call Sequence

Errore. Non si possono creare oggetti dalla modifica di codici di campo.

DEVICE-FALLBACK: Each device has (a) a fallback-device-ID attribute, which references another device in the device capabilities data, and (b) a root-device attribute that indicates whether the device is a major version/generic. If a search operation shall yield a match that is not a root device then the HME shall “fall back” – it shall re-search the device capabilities data for the referenced fallback-device-ID. This allows the HME to optimize the search result when the calling application provides generic or ambiguous information.

HME – Finding Device and Fallback Process/Flow



Examples:

- a. The search for device by user agent “*Nokia3100/1.0 (05.65) Profile/MIDP-1.0 Configuration/CLDC-1.0*” resulted with device id “*nokia_3100_ver1_sub0565*”, which is not a root device. The HME shall fall back to the “*nokia_3100_ver1_sub0565*” root device - “*nokia_3100_ver1*”.

- b. The search for a device ID “*nokia_3100_ver1_sub0554100*” - a later or similar version of the “*nokia_3100_ver1*” device shall fall back to the root “*nokia_3100_ver1*”.
- c. The search for model *Nokia3100* yields multiple results (e.g. various versions of the same machine) - the HME shall fallback to the major root device “*nokia_3100_ver1*”.
- d. A search for a handset XX of manufacturer X implements the capabilities set of root device “*nokia_3100_ver1*” – the HME shall fallback to “*nokia_3100_ver1*”.
- e. The search for a device included no parameters or yielded no results – The HME shall fallback to the default device. (see *configuration*).

4. Restart:

TRIGGER: On request from a client application, through the HME interface.

FUNCTION: The HME shall restart itself;

OPERATION: The HME shall stop accepting requests until the operation is done. It shall finish all ongoing operations, then close all open files or connections, remove all objects, references, configuration data, device management data and all other data that is loaded in memory. When done, the HME shall reload and apply the configuration, and load the handset management data. When done, the HME shall return to accept application calls. (*Figure: HME – Restart Call*)

HMS – Restart Call Sequence

Errore. Non si possono creare oggetti dalla modifica di codici di campo.

5. Shut Down:

TRIGGER: On request from a client application, through the HME interface, and on manual shut down.

FUNCTION: The HME shall orderly shut itself down;

OPERATION: The HME shall stop accepting requests. It shall finish all ongoing operations, then close all open files or connections, remove all objects, references, configuration, device management and all other data that is loaded in memory. When done, the HME application shall terminate itself. (*Figure: HME – Shut Down Call*)

HMS – Shut Down Call Sequence

Errore. Non si possono creare oggetti dalla modifica di codici di campo.

Application Interface:

The HME application shall use the COM technology to make functionality public to client applications. It shall expose objects and methods for client applications to call. All interactions shall be initiated by the client applications only. The HME shall handle application calls to in parallel (multi-threaded). Each call shall be synchronous - the control shall be returned to the calling application/thread when the operation is done.

The HME shall expose the following COM object and methods for application interface:

COMVERSEHME.HandsetManagement

Object Name:	COMVERSEHME.HandsetManagement		
Method Name	Functionality	Parameter(s)	Return value
GetDeviceCapabilities	Find Device Capabilities by Device Identifiers	GET_DEVICE_CAPABILITIES request XML (see Get Device Capabilities Request – Detailed Structure)	GET_DEVICE_CAPABILITIES reply XML (see Get Device Capabilities Reply – Detailed Structure)

Restart	Restart	HME_RESTART request XML (see Restart HME Request – Detailed Structure)	HME_RESTART reply XML (see Restart HME Reply – Detailed Structure)
ShutDown	Shut Down	HME_SHUT_DOWN request XML (see Shut Down HME Request – Detailed Structure)	HME_SHUT_DOWN reply XML (see Shut Down HME Reply – Detailed Structure)

Data:Storage:

The device capabilities database is small and static. Each installation of the distribution-to-mobile system shall typically support only a subset of root devices and capabilities, which shall not change often. So there is no need for database services. Therefore the device capabilities data shall be available to the HME in the form of a file.

Content:

The data stored in the file shall list that includes, but is not limited to root devices, their later versions and non root devices, manufacturer basic specifications, and a default device.

The root device description shall include device identifiers (id, user agent, model, etc), detailed specifications of the capabilities, categorized to groups, and a reference to a more general device model to fall back to.

The non root device description shall include device identifiers, and a reference to a major/earlier version/more general root device.

The default device does not have to comply with a specific device manufacturer, model or version. It is typically required to describe a device that supports minimal or maximal capabilities, and shall be used when the device capabilities cannot be determined. The id of the default device is set in the configuration, but the definition must be specified in the device capabilities data.

Example:

nokia_3100_ver_1 is the id of a root device. It lists all the capabilities of the first Nokia 3100 version.

nokia_3100_ver1_sub0565 is the id of the non root device – a later version of the Nokia 3100, possibly customized for a specific market. This device shall not list its capabilities, but shall reference the *nokia_3100_ver_1* as the device to get the capabilities from – to fall back to.

Nokia_40_ver1 is the general basic device that all Nokia 3x models can fall back to. All its listed capabilities are supported by the 3x series (but not the other way around).

ALL_capabilities_ver1 (just an example id – not real) is a virtual device that supports all possible capabilities. It is not limited by any mean or measure. It is used as the default device in order to not limit the users from browsing by capabilities, when their device capabilities cannot be determined.

Format:

The devices and capabilities shall be described using XML. (See **Device Capabilities Data – Detailed Structure**)

Source and Updates:

There shall be no dedicated GUI for adding, editing or deleting devices from the device capabilities data file. Any XML/text tool can be used to edit the file.

A The device capabilities XML and DTD can be downloaded from websites of open source resources, such as the WURFL project (<http://wurfl.sourceforge.net>). However, different installations shall require different devices and capabilities configuration so the updater should use his/her discretion.

Configuration:

The HME configuration shall use the XML format. (See **HME configuration – detailed structure**)

The initial configuration shall instruct the application to:

1. Load the Devices Capabilities data from the device capabilities file (see **HMS deployment**)

2. Refer to a virtual-device with broadest capabilities as the default device (with the purpose to fall back to a limit-less device, when the device cannot be determined).
3. Log only the following events: Application Start, Application Restart, Application Shut Down, Error, Device Fallback, and Device Fallback to Default
4. Log the Event Date & Time and Request Body data for each event that is logged.

Module – Web API:

Classification:

The Web API shall be developed as a set of web pages, which shall intermediate between the HME and client applications. Each page shall make a single public HME function available to client applications.

Implementation Platform:

Common Web API implementation platform (See **Common Guidelines, Implementation Platform**)

Functionality by Page:

1. [hms_get_device_capabilities.asp](#):
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: HME Get Device Capabilities Request XML. (See **Get Device Capabilities Request – Detailed Structure**)
RESPONSE MESSAGE: Device Capabilities XML and/or Error block, indicating a successful, fallback, or failed attempt to process the request. (See **Get Device Capabilities Reply – Detailed Structure**)
OPERATION: Invoke the COMVERSEHME.HandsetManagement.GetDeviceCapabilites method, passing the message body as parameter, and set the returned value as the response body.
2. [hms_restart.asp](#):
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: HME Restart Request XML. (See **Restart HME Request – Detailed Structure**)
RESPONSE MESSAGE: HME Restart XML Error block, indicating a successful or failed HME Restart attempt. (See **Restart HME Reply – Detailed Structure**)
OPERATION: Invoke the COMVERSEHME.HandsetManagement.Restart method, passing the message body as parameter, and set the returned value as the response body.
3. [hms_shut_down.asp](#):
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: HME Shut Down Request XML. (See **Shut Down HME Request – Detailed Structure**)
RESPONSE MESSAGE: HME Shut Down Error block, indicating a successful or failed HME Shut Down attempt. (See **Shut Down HME Reply – Detailed Structure**)
OPERATION: Invoke the COMVERSEHME.HandsetManagement.ShutDown method, passing the message body as parameter, and set the returned value as the response body.

Deployment:

Infrastructure

The entire HMS system shall be deployed on a single Intel based machine, operated by Microsoft Windows 2000 server or Microsoft 3000 OS.

IIS5 shall be available and operative on the machine.

HME

The HME shall be installed on the machine, and its COM components registered. The Application, Configuration and data resources shall be deployed to the following folders:

1. Binary files to: [hme_application_folder]/bin/
2. Configuration files: [hme_application_folder]/cnf/
3. Data files: [hme_application_folder]/dat/

An [hme_application_folder]/log] folder shall be created to accommodate all HME logs.

API Layer

The API Layer shall be deployed on the IIS5, under a Web site named hms, to be accessed through: [http\(s\)://server/\[company_root\]/hms/page.asp](http(s)://server/[company_root]/hms/page.asp).

e.g. http://axserver/COMVERSE/hms/hms_get_device_capabilities.asp

4.3.6.3 Personalization Service

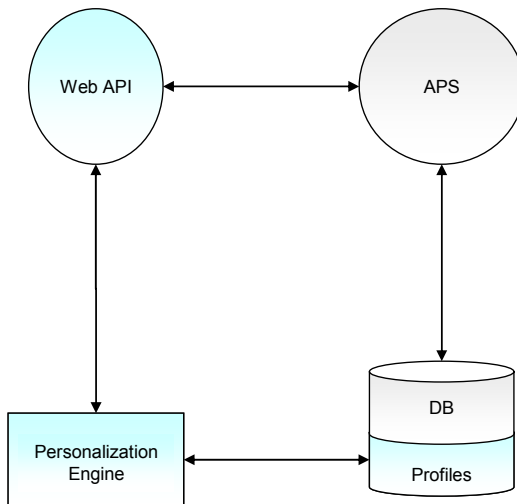
Overview

The PS shall allow the COMVERSE Distribution-to-Mobile system to store, load and apply the subscriber's browsing preferences, to enhance and optimize the user experience.

Architecture:

The PS shall incorporate (a) a core PE module, which shall perform all the required operations, (b) a database server for storing the personalization preferences, and (c) a Web API layer, which shall serve as the main interface to the PE for client applications. The interaction between client applications and the PS, through the API shall rely on XML over HTTP(S).

PS – Architecture Diagram



Module – PE (Personalization Engine):

Classification:

The PE shall be developed as a stand alone, multi-threaded, executable.

Implementation Platform:

Common executable implementation platform (See **Common Guidelines, Implementation Platform**)

Functionality:

The PE shall perform the following functionalities:

1. Load configuration;

TRIGGER: on application start and when Restarted.

FUNCTION: The PE shall load the configuration file and apply the settings. Throughout the operation the PE shall not accept any requests made from client applications.

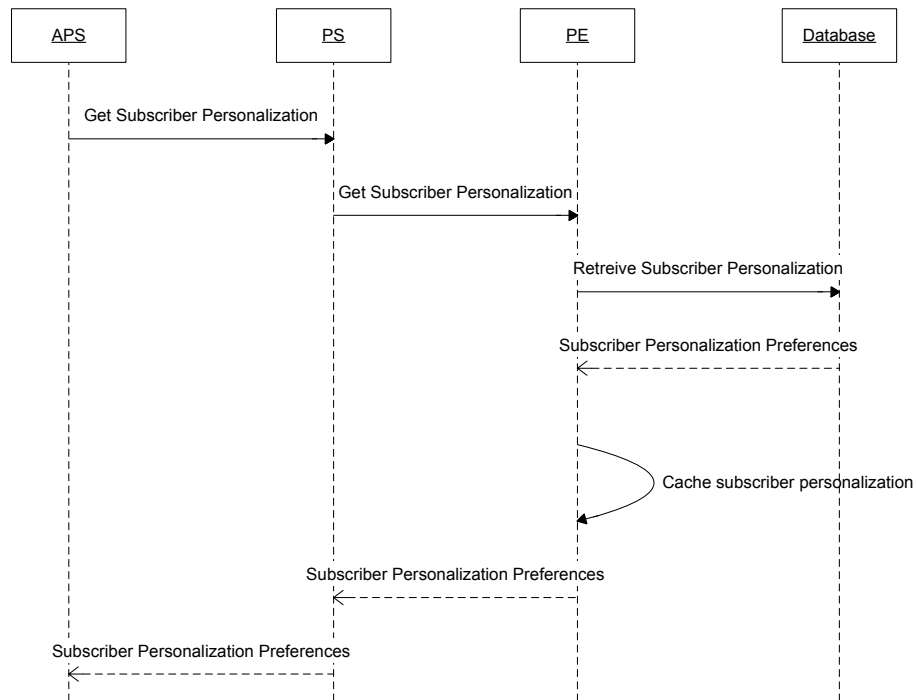
2. Get Subscriber Personalization Preferences

TRIGGER: On request from a client application, through the PE interface

FUNCTION: The PE shall load the personalization preferences of the subscriber from the database, cache it, and return in to the calling application.

OPERATION: The PE shall load the preferences of the subscriber by the Subscriber ID that was provided by the calling application. The PE shall cache the loaded preferences and return them to the calling application in XML. (**Figure: PE – Get Subscriber Personalization Call**)

VARIATION: If the subscriber's preferences are cached the PE shall return the cached preferences and not load it from the database.

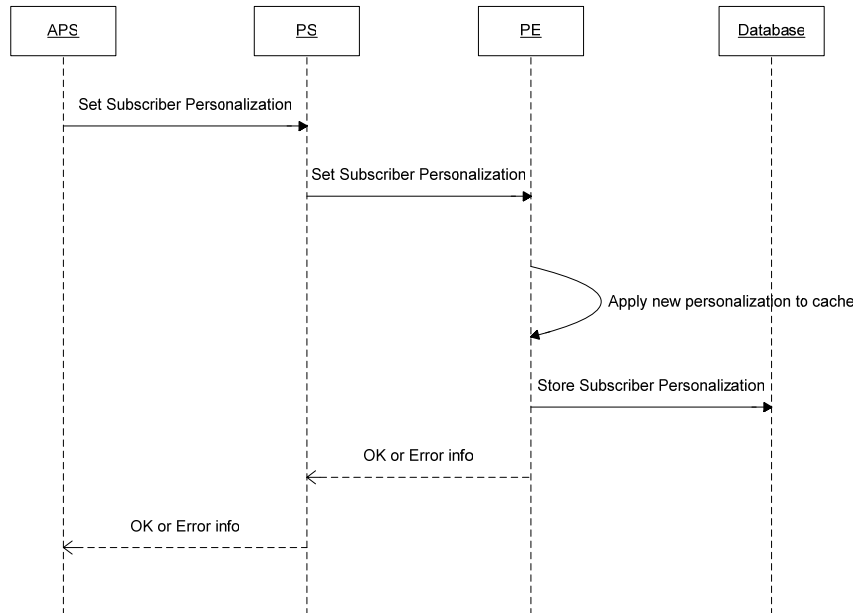
PS – Get Subscriber Personalization Call Sequence3. Set Subscriber Personalization Preferences

TRIGGER: On request from a client application, through the PE interface

FUNCTION: The PE shall apply the specified personalization preferences to the subscriber profile in the cache, and store it to the database.

OPERATION: The PE shall update the partial or full preferences that were provided by the calling application, to the cached preferences of the specific subscriber, and store the updated personalization info to the database. (**Figure: PE – Set Subscriber Personalization Call**)

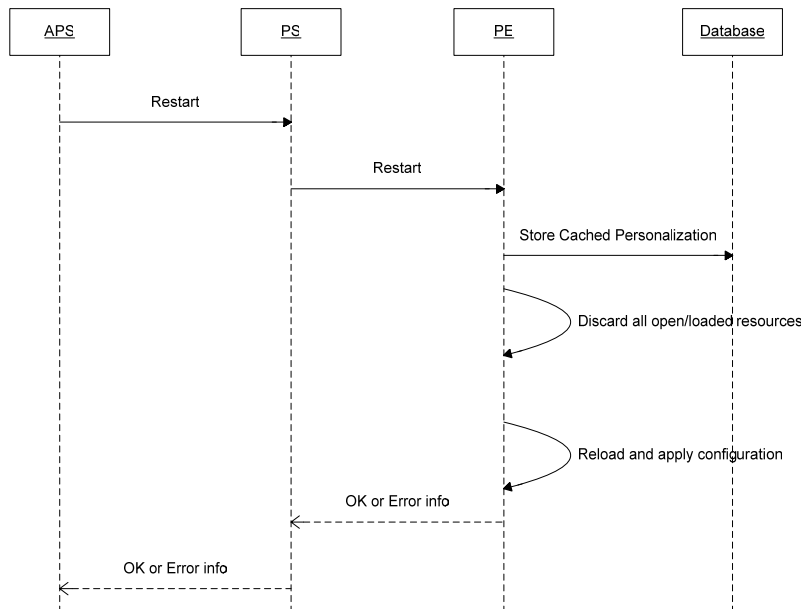
VARIATION: If the subscriber preferences are not cached then the PE shall only update the subscriber preferences in the database.

PS – Set Subscriber Personalization Call Sequence4. Restart:

TRIGGER: On request from a client application, through the PE interface.

FUNCTION: The PE shall restart itself;

OPERATION: The PE shall stop accepting requests until the operation is done. It shall finish all ongoing operations, store all subscriber personalization data that is cached, then close all open files or connections, remove all objects, references, configuration, personalization cache and all other data that is loaded in memory. When done, the PE shall reload and apply the configuration. When done, the PE shall return to accept application calls. (*Figure: PE – Restart Call*)

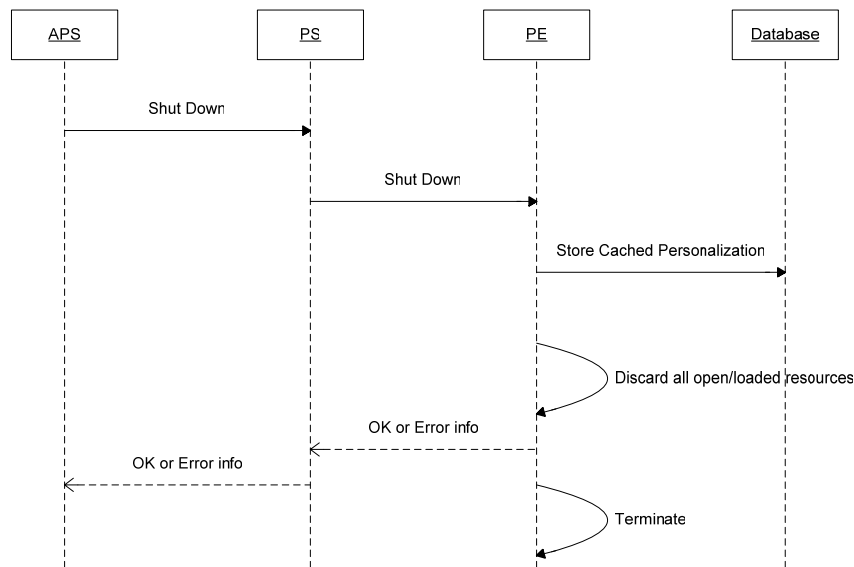
PS – Restart Call Sequence

5. Shut Down;

TRIGGER: On request from a client application, through the PE interface, and on manual shut down.

FUNCTION: The PE shall orderly shut itself down;

OPERATION: The PE shall stop accepting requests. It shall finish all ongoing operations and store all cached subscriber personalization data, then close all open files or connections, remove all objects, references, configuration, personalization cache and all other data that is loaded in memory. When done, the PE application shall terminate itself. (**Figure: PE – Shut Down Call**)

PS – Shut Down Call Sequence**Application Interface:**

The PE application shall use the COM technology to make functionality public to client applications. It shall expose objects and methods for client applications to call. All interactions shall be initiated by the client applications only. The PE shall handle application calls to in parallel (multi-threaded). Each call shall be synchronous - the control shall be returned to the calling application/thread when the operation is done.

The PE shall expose the following COM object and methods for application interface:

COMVERSEPE.Personalization

Object Name:	COMVERSEPE.Personalization		
Method Name	Functionality	Parameter(s)	Return value
GetSubscriberPersonalization	Load the Personalization preferences of the subscriber	GET_SUBSCRIBER_PERSONALIZATION request XML (see Get Subscriber Personalization Request – Detailed Structure)	GET_SUBSCRIBER_PERSONALIZATION reply XML (see Get Subscriber Personalization Reply – Detailed Structure)
SetSubscriberPersonalization	Apply preferences to the	SET_SUBSCRIBER_PERSONALIZATION request XML	SET_SUBSCRIBER_PERSONALIZATION reply XML

	Personalization preferences of the subscriber	(see Set Subscriber Personalization Request – Detailed Structure)	(see Set Subscriber Personalization Reply – Detailed Structure)
Restart	Restart	PE_RESTART request XML (see Restart PE Request – Detailed Structure)	PE_RESTART reply XML (see Restart PE Reply – Detailed Structure)
ShutDown	Shut Down	PE_SHUT_DOWN request XML (see Shut Down PE Request – Detailed Structure)	PE_SHUT_DOWN reply XML (see Shut Down PE Reply – Detailed Structure)

Data:Storage:

The Personalization data shall be stored in a database.

Content:

The Personalization Engine shall load, store and cache the following Preferences for each subscriber: (See **Personalization Preferences - Detailed Structure**)

1. Subscriber ID
2. The Name of the field that the Media Items list shall be sorted by
3. The Order of the sort; Ascending or Descending
4. A list of One, None, or Many Media Items that were added to the subscriber Wish List.

Configuration:

The PE configuration shall use the XML format. (See **PE configuration – detailed structure**)

The initial configuration shall instruct the application to:

5. Connect to the specific personalization database
6. Log only the following events: Application Start, Application Restart, Application Shut Down, Error
7. Log the Event Date & Time and Request Body data for each event that is logged.

Module – Web API:**Classification:**

The Web API shall be developed as a set of web pages, which shall intermediate between the PE and client applications. Each page shall make a single public PE function available to client applications.

Implementation Platform:

Common Web API implementation platform (See **Common Guidelines, Implementation Platform**)

Functionality by Page:

4. PS_get_subscriber_personalization.asp:
 INVOKE METHOD: HTTP Post
 EXPECTED REQUEST MESSAGE: PE Get Subscriber Personalization Request XML. (See **Get Subscriber Personalization Request – Detailed Structure**)
 RESPONSE MESSAGE: The personalization preferences of the Subscriber XML and/or Error block, indicating a successful, fallback, or failed attempt to process the request. (See **Get Subscriber Personalization Reply – Detailed Structure**)
 OPERATION: Invoke the COMVERSEPE.Personalization.GetSubscriberPersonalization method, passing the message body as parameter, and set the returned value as the response body.
5. PS_set_subscriber_personalization.asp:
 INVOKE METHOD: HTTP Post

EXPECTED REQUEST MESSAGE: PE Set Subscriber Personalization Request XML. (See **Set Subscriber Personalization Request – Detailed Structure**)

RESPONSE MESSAGE: The personalization preferences of the Subscriber XML and/or Error block, indicating a successful, fallback, or failed attempt to process the request. (See **Set Subscriber Personalization Reply – Detailed Structure**)

OPERATION: Invoke the COMVERSEPE.Personalization.SetSubscriberPersonalization method, passing the message body as parameter, and set the returned value as the response body.

6. PS_restart.asp:

INVOKE METHOD: HTTP Post

EXPECTED REQUEST MESSAGE: PE Restart Request XML. (See **Restart PE Request – Detailed Structure**)

RESPONSE MESSAGE: PE Restart XML Error block, indicating a successful or failed PE Restart attempt. (See **Restart PE Reply – Detailed Structure**)

OPERATION: Invoke the COMVERSEPE.Personalization.Restart method, passing the message body as parameter, and set the returned value as the response body.

7. PS_shut_down.asp:

INVOKE METHOD: HTTP Post

EXPECTED REQUEST MESSAGE: PE Shut Down Request XML. (See **Shut Down PE Request – Detailed Structure**)

RESPONSE MESSAGE: PE Shut Down Error block, indicating a successful or failed PE Shut Down attempt. (See **Shut Down PE Reply – Detailed Structure**)

OPERATION: Invoke the COMVERSEPE.Personalization.HandsetManagement.ShutDown method, passing the message body as parameter, and set the returned value as the response body.

Deployment:

Infrastructure

The entire PS system shall be deployed on a single Intel based machine, operated by Microsoft Windows 2000 server or Microsoft 3000 OS.

IIS5 shall be available and operative on the machine.

PE

The PE shall be installed on the machine, and its COM components registered. The Application, Configuration and data resources shall be deployed to the following folders:

4. Binary files to: [PE_application_folder]/bin/

5. Configuration files: [PE_application_folder]/cnf/

An [PE_application_folder]/log] folder shall be created to accommodate all PE logs.

API Layer

The API Layer shall be deployed on the IIS5, under a Web site named PS, to be accessed through: http(s)://server/[company_root]/PS/page.asp.

e.g. http://axserver/COMVERSE/PS/PS_get_subscriber_personalization.asp

4.3.6.4 AXMEDIS Plugin

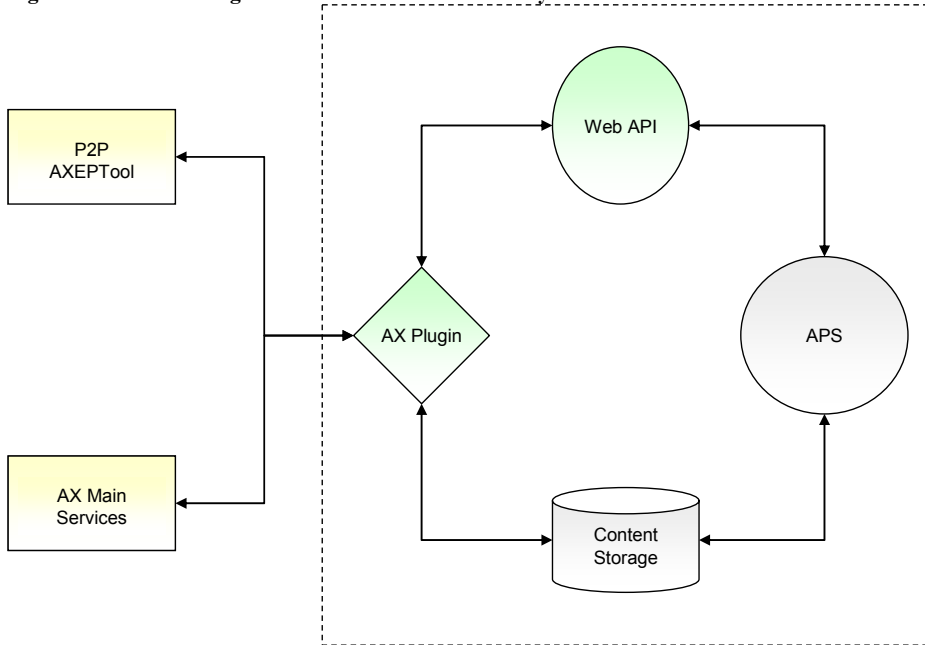
Overview

The AX Plugin Service shall allow the Comverse Distribution-to-Mobile system to query for catalog and content objects on the AXMEDIS, retrieve and use them. The AXMEDIS security and workflow aspects shall be encapsulated in the AX Plugin.

Architecture:

The AX Plugin Service shall incorporate (a) a core AX Plugin module, which shall perform all the required operations, and (b) a Web API layer, which shall serve as the main interface to the AX Plugin for client applications. The interaction between client applications and the AX Plugin Service, through the API shall rely on XML over HTTP(S). The AX Plugin shall function as a client application when interacting with the AXEPTool and AXMEDIS platform services.

Figure: AXMEDIS Plugin for Converse Distribution System Architecture

**Module – AX Plugin (AXMEDIS Plugin):****Classification:**

The AX Plugin shall be developed as a stand alone, multi-threaded, executable.

Implementation Platform:

Common executable implementation platform (See **Common Guidelines, Implementation Platform**)

Functionality:

The AX Plugin shall perform the following functionalities:

1. Load configuration;

TRIGGER: on application start and when Restarted.

FUNCTION: The AX Plugin shall load the configuration file and apply the settings. Throughout the operation the AX Plugin shall not accept any requests made from client applications.

2. Get Catalogue Category Tree;

TRIGGER: On request from a client application.

FUNCTION: The AX Plugin shall download the entire catalogue, including the required assets, and return the category tree structure and the location of the assets.

OPERATION: The AX Plugin shall download the entire category tree from the AXMEDIS. The download shall begin at the root category object, which is identified by OID in the configuration, and assets. The root category shall reference the child categories, which shall be downloaded next, and so on for all nesting categories. All the category assets shall be unprotected and unpacked, then cached on file storage that is accessible to the calling application. When done, the AX Plugin shall return an XML describing the category tree, with all category properties and references to the cached

assets of each category, to the calling application.

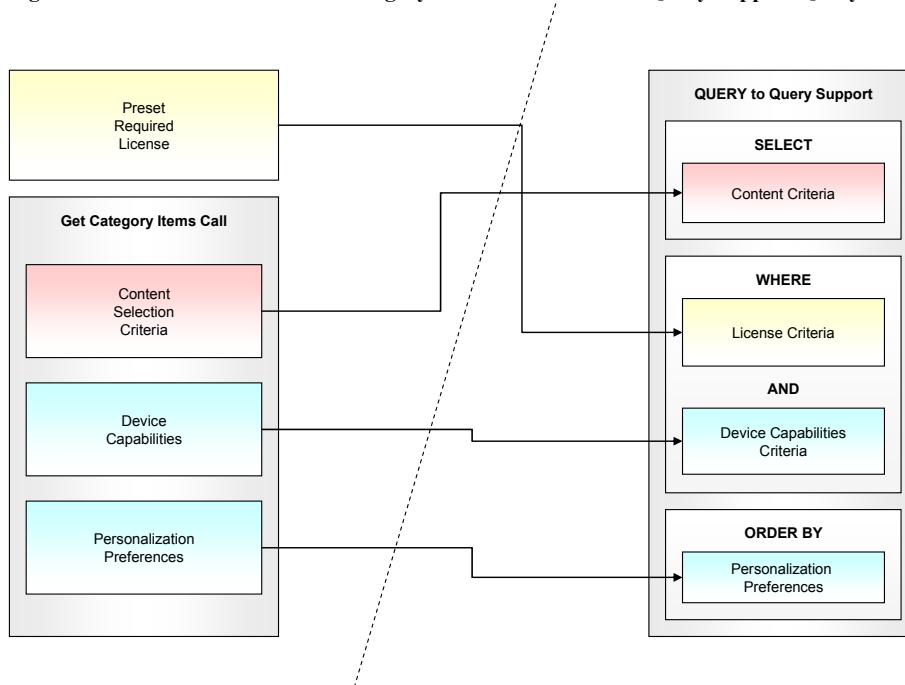
3. Get Category Content Items;

TRIGGER: On request from a client application.

FUNCTION: The AX Plugin shall query the AX DB for content objects that are offered from a Catalog category. The query shall consider license, device capabilities and user preferences. The results of the query shall be returned to the calling application.

OPERATION: The AX Plugin shall receive the call containing (a) the definition of the desired content, (b) the capabilities of the destination device, and (c) the personalized preferences of the subscriber. The plugin shall add the required license definition to the content information to transform it to a Query Support query XML. The transformed query shall **SELECT desired content criteria**, **WHERE license criteria AND device-capabilities criteria** **ORDER BY personalized preferences**.

Figure: Transformation from Get Category Content Items Call to Query Support Query XML



4. Get Content Sample Item;

TRIGGER: On request from a client application.

FUNCTION: The AX Plugin shall download a content sample object from AXMEDIS, cache its content and return the cached location to the calling application.

OPERATION: The AX Plugin shall receive a call to download an object by OID, as a content sample. The AX Plugin shall fetch the object from the AXEPTool or, if not available on the AXEPTool from the AX DB. The AX Plugin shall unprotect, unpack and cache the content in a file storage that is accessible to the calling application. The AX shall then return the location of the cached sample file to the calling application.

5. Get Content Item;

TRIGGER: On request from a client application.

FUNCTION: The AX Plugin shall download a content object from AXMEDIS, cache its content and return the cached location to the calling application. The AX Plugin shall report the operation to the proper AXMEDIS services..

OPERATION: The AX Plugin shall receive a call to download an object by OID for consumption.

The AX Plugin shall download the get the permissions to download, unprotect, unpack, cache and distribute the file as desired by the Comverse Distribution license (i.e. Download once, distribute many – report each distribution for consumption not by AXMEDIS). The AX Plugin shall download, unprotect, unpack and cache the content to a s file storage that is accessible to the calling application. The AX shall then return the location of the cached sample file to the calling application.

6. Restart:

TRIGGER: On request from a client application.

FUNCTION: The AX Plugin shall restart itself;

OPERATION: The AX Plugin shall stop accepting requests until the operation is done. It shall finish all ongoing operations, then close all open files or connections, remove all objects, references, configuration and all other data that is loaded in memory. When done, the AX Plugin shall reload and apply the configuration. When done, the AX Plugin shall return to accept application calls.

7. Shut Down:

TRIGGER: On request from a client application, and on manual shut down.

FUNCTION: The AX Plugin shall orderly shut itself down;

OPERATION: The AX Plugin shall stop accepting requests. It shall finish all ongoing operations, then close all open files or connections, remove all objects, references, configuration, device management and all other data that is loaded in memory. When done, the AX Plugin application shall terminate itself.

Application Interface:

The AX Plugin application shall use the COM technology to make functionality public to client applications. It shall expose objects and methods for client applications to call. All interactions shall be initiated by the client applications only. The AX Plugin shall handle application calls to in parallel (multi-threaded). Each call shall be synchronous - the control shall be returned to the calling application/thread when the operation is done.

The AX Plugin shall expose the following COM object and methods for application interface:

ComverseAXMEDIS.AXPlugin

Object Name:	ComverseAXMEDIS.AXPlugin		
Method Name	Functionality	Parameter(s)	Return value
GetCatalogue	Get Catalog Category Tree	AXPG_GET_CATALOG request XML	AXPG_GET_CATALOG reply XML
GetCategoryContentItems	Get Category Content Items	AXPG_GET_CATEGORY_CONTENT_ITEMS request XML	AXPG_GET_CATEGORY_CONTENT_ITEMS reply XML
GetContentSampleItem	Get Content Sample Item	AXPG_GET_CONTENT_SAMPLE_ITEM request XML	AXPG_GET_CONTENT_SAMPLE_ITEM reply XML
GetContentItem	Get Content Item	AXPG_GET_CONTENT_ITEM request XML	AXPG_GET_CONTENT_ITEM reply XML
Restart	Restart	AXPG_RESTART request XML	AXPG_RESTART reply XML
ShutDown	Shut Down	AXPG_SHUT_DOWN request XML	AXPG_SHUT_DOWN reply XML

Configuration:

The AX Plugin configuration shall use the XML format.

AXMEDIS Project

CONFIDENTIAL

The AX configuration shall include:

1. The OID of the root category of the Catalog
2. The Location of the file cache storage (i.e. path to drive or folder)
3. The Comverse Distribution-to-Mobile License attributes XML.

The initial configuration shall instruct the application to:

1. Log only the following events: Application Start, Application Restart, Application Shut Down, Error
2. Log the Event Date & Time and Request Body data for each event that is logged.

Module – Web API:

Classification:

The Web API shall be developed as a set of web pages, which shall intermediate between the AX Plugin and client applications. Each page shall make a single public AX Plugin function available to client applications.

Implementation Platform:

Common Web API implementation platform (See **Common Guidelines, Implementation Platform**)

Functionality by Page:

1. AXPG get catalogue.asp:
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: AXPG_GET_CATALOG request XML.
RESPONSE MESSAGE: AXPG_GET_CATALOG reply XML and/or Error block, indicating a successful or failed attempt to process the request.
OPERATION: Invoke the ComverseAXMEDIS.AXPlugin.GetCatalogue method.
2. AXPG get category content items.asp:
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: AXPG_GET_CATEGORY_CONTEN_ITEMS Request XML.
RESPONSE MESSAGE: AXPG_GET_CATEGORY_CONTEN_ITEMS reply XML and/or Error block, indicating a successful or failed attempt to process the request.
OPERATION: Invoke the ComverseAXMEDIS.AXPlugin.GetCategoryContentItems method.
3. AXPG get content sample item.asp:
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: AXPG_GET_CONTENT_SAMPE_ITEM Request XML.
RESPONSE MESSAGE: AXPG_GET_CONTENT_SAMPE_ITEM reply XML and/or Error block, indicating a successful or failed attempt to process the request.
OPERATION: Invoke the ComverseAXMEDIS.AXPlugin.GetCatalogue method.
4. AXPG get content item.asp:
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: AXPG_GET_CONTENT_ITEM Request XML.
RESPONSE MESSAGE: AXPG_GET_CONTENT_ITEM reply XML and/or Error block, indicating a successful or failed attempt to process the request.
OPERATION: Invoke the ComverseAXMEDIS.AXPlugin.GetCatalogue method.
5. AXPG restart.asp:
INVOKE METHOD: HTTP Post
EXPECTED REQUEST MESSAGE: AXPG_RESTART request XML.
RESPONSE MESSAGE: AXPG_RESTART reply XML Error block, indicating a successful or failed AX Plugin Restart attempt.

OPERATION: Invoke the ComverseAX PLUGIN.HandsetManagement.Restart method, passing the message body as parameter, and set the returned value as the response body.

6. AXPG shut_down.asp:

INVOKE METHOD: HTTP Post

EXPECTED REQUEST MESSAGE: AXPG_SHUT_DOWN Request XML.

RESPONSE MESSAGE: AX Plugin Shut Down Error block, indicating a successful or failed AX Plugin Shut Down attempt.

OPERATION: Invoke the ComverseAX PLUGIN.HandsetManagement.ShutDown method, passing the message body as parameter, and set the returned value as the response body.

Deployment:

Infrastructure

The entire AX Plugin Service system shall be deployed on a single Intel based machine, operated by Microsoft Windows 2000 server or Microsoft 3000 OS.

IIS5 shall be available and operative on the machine.

AX Plugin

The AX Plugin shall be installed on the machine, and its COM components registered. The Application, Configuration and data resources shall be deployed to the following folders:

1. Binary files to: [AX Plugin_application_folder]/bin/
2. Configuration files: [AX Plugin_application_folder]/cnf/
3. Data files: [AX Plugin_application_folder]/dat/

An [AX Plugin_application_folder]/log] folder shall be created to accommodate all AX Plugin logs.

API Layer

The API Layer shall be deployed on the IIS5, under a Web site named AXPS, to be accessed through: [http\(s\)://server/\[company_root\]/AXPS/page.asp](http(s)://server/[company_root]/AXPS/page.asp).

e.g. http://axserver/comverse/AXPS/AXPS_get_device_capabilities.asp

4.3.6.5 P2P AXEPTool

Role and Functionality

The AXEPTool role in the distribution territory is to make AXMEDIS objects available for fast fetching; namely to serve as the cache for AXMEDIS objects. This shall be achieved by adding Active Selections that define the desired objects for downloading and storing in the AXEPTool. The AX Plugin can then fast-fetch the files from the AXEPTool database rather than wait for them to be download from the network.

Desired Objects

The AXMEDIS objects that shall be required by distribution system are:

1. Catalog Definition – All Category objects that are in the Catalogue(s) that the APS uses, including the digital assets they contain.
2. Content Manager Choice; Multimedia Objects that are referenced in the catalog Categories, and are defined by the Content Manager as Priority, Marketing, Popular, etc.
3. Catalog Media – All Multimedia Objects that are referenced in the catalog Categories (this option requires storage-space control).

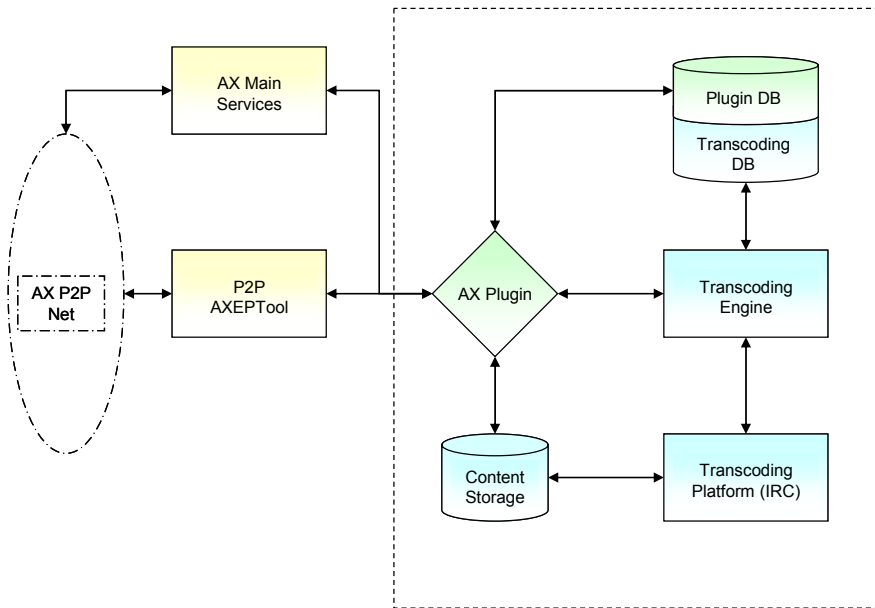
Common Active Selection Criteria

All Active Selection definition shall include the license and terms of usage criteria for the territory. That is – All Active Selections for the distribution territory shall include the license criteria that are required for distributing content via the COMVERSE Distribution System. Although not enforced, this feature is highly recommended for efficiency, performance and user experience (prevent unnecessary delays).

4.4 Transcoding Territory

4.4.1 General Architecture

Transcoding Territory – Architecture Diagram



4.4.2 Modules

4.4.2.1 COMVERSE Transcoding Server

Missing contribution from COMVERSE.

4.4.2.2 Transcoding Platform

Implementation of the OMA STI

COMVERSE has performed a research on existing and developing Transcoding Standards that suit the Telephony industry and found the OMA STI 1.0 (OMA Standard Transcoding Interface specification, version 1.0) to be the most suitable for the Transcoding Platform.

Therefore the Transcoding Platform service shall be implemented according to the OMA STI 1.0 specifications.

OMA STI Specification Summary

The main definitions and functionalities that are described by the OMA STI and shall be implemented in the version of the AXMEDIS Transcoding Platform include:

1. **Service Interface**
The interface between the application and the transcoding platform relies on SOAP 1.1 (not COMVERSE Proprietary XML) over HTTP(S). Both the request and the response contain the OMA STI SOAP envelope.
2. **Operations**
 - a. Each request defines a single transcoding operation.
 - b. Each transcoding operation contains one or more transcoding jobs.
 - c. Each job shall transcode a single or multi-part content file (e.g. Mpeg containing video + audio streams, text + graphics + audio presentations).
3. **Transcoding Content Access**
 - a. The source content for each job can be attached to transcoding request. The resulting content will be attached in the response.
 - b. The source content can be referenced by a URI. In this case the transcoder will upload the resulting content to a target location using the HTTP(S) PUT method. The spec requires the support of the HTTP(S) protocol for the URI, but it can also work with other protocols, such as FILE (for local/mapped storage device)
 - c. The request can contain a combination of jobs with content attached in the request and jobs with referenced content.
4. **Supported Content Types**
 - a. Basic content types supported: Image, Audio, Video (+ Audio stream), Text, Multipart.
 - b. Any content that is defined between the application and the transcoding platform
5. **Supported Codecs**
 - a. The platform supports Codecs for which a MIME type has been registered at IANA (<http://www.iana.org/assignments/media-types/>) (e.g. a file format registration such as audio/amr, audio/amr-wb, video/h263, audio/g723, etc)
 - b. Any codec defined between the transcoding platform and application
6. **Transitions**
For each job a set of transformations can be defined (e.g. image rotate, resize, audio duration limit, video frame rate, etc)
7. **Job Size Limit**
The size of each transcoding job can be limited at the content, job, operation and application level (e.g. at application level can specify that no resulting content will exceed 30K)
8. **Supports Adaptation Profiles and Parameters**
The transcoding platform can access an adaptation classes (URI specified for a job) which define the transcoding and formatting information to be considered when performing transcoding, such as equipment characteristics and capabilities
9. **Deployment**
The transcoding platform and application can share the same physical machine or even be part of the same software.

Specification EXCEPTIONS for AXMEDIS

The following exceptions shall apply to the AXMEDIS Implementation of the OMA STI specifications:

1. Operations
 - a. Each request defines a single transcoding operation.
 - b. Each transcoding operation contains one or more transcoding jobs.
 - c. Each job shall transcode a single or multi-part content file (e.g. Mpeg containing video + audio streams, text + graphics + audio presentations).
2. Transcoding Content Access
 - d. The source content for each job shall NOT be attached to transcoding request, nor shall the resulting content be attached in the response.
 - e. The source content for every job shall be referenced by a URI. The transcoder shall upload the resulting content to a target location using the HTTP(S) PUT method.
3. Supported Input Codecs
 - f. Windows Audio WAV, 44hz, 6 bit stereo or mono
 - g. GIF, JPEG, or Windows Bitmap graphics
4. Supported Output Codecs
 - h. COMVERSE has not yet decided what output formats shall be supported by the Transcoding Platform.
5. Deployment

The transcoding platform and application can share the same physical machine or even be part of the same software.

Informative and Sources

The OMA STI is a project of the BAC (Browser and Content) workgroup (http://www.openmobilealliance.org/tech/wg_committees/bac.html)

Public OMA STI material, including the STI Specifications documentation can be found at: http://member.openmobilealliance.org/ftp/public_documents/bac/STI/Permanent_documents/

4.4.2.3 P2P AXEPTool

Role and Functionality

The AXEPTool role in the Transcoding territory is to make AXMEDIS objects available for fast fetching; namely to serve as the cache for AXMEDIS objects. This shall be achieved by placing Active Selections that define the desired objects for downloading and storing in the AXEPTool. The AX Plugin can then fast-fetch the files from the AXEPTool database rather than wait for them to be download from the network. The Active Selections shall be updated by the transcoding system as required.

Desired Objects

The AXMEDIS objects that shall be required by distribution system are:

4. Catalog Definition – All Category objects that are in the Catalogue(s) that the APS uses, including the digital assets they contain.
5. Objects to be Transcoded; Multimedia Objects that needs to be transcoded and adapted for distribution to mobile devices.

Common Active Selection Criteria

All Active Selection definition shall include the license and terms of usage criteria for the territory. That is – All Active Selections for the Transcoding territory shall include the license criteria that are required for transcoding (which includes distribution criteria) content via the COMVERSE Distribution System.

4.4.2.4 System Common Data – Detailed Structure

COMVERSE File and Files Collection – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	FILES	Files Collection		A files data/collection
FILES	FILE	File Item	0..n	A file data/object
FILE	title	String	1	A title or short description of the file
	encoding_type	Enum	1	The encoding type that the asset was encoded with
	Path	String	1	The File Name and Path, referencing an asset (such as narration audio) file on a drive or a server

COMVERSE Meta Info – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	META_INFO	Info Collection		A collection of info items
META_INFO	INFO	Info	0..n	An Info data/object
INFO	title	String	1	The title of the media item
	data	String	1	The info data, such as Lyrics text, a reference to an Album Icon, reference to other multimedia items, etc.

COMVERSE Category and Categories collection – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	CATEGORIES	Category Collection		A collection of Category items
CATEGORIES	CATEGORY	Category	0..n	
CATEGORY	id	Number	1	The Category unique identifier
	type	Number	1	1 for genre, Or 2 for category
	owner	String	1	The name of the content owner; Typically used for identifying the content provider
	title	String	1	The category title
	full_path	String	1	
	description	String	1	Category Description
	FILES	Files Collection	1	References to asset files, and relevant information. (See COMVERSE File and Files Collection – Detailed Structure)
	META_INFO	Meta Info Collection	1	Related Meta Info, such as related items, etc (See COMVERSE Meta Info – Detailed Structure)
	CATEGORIES	Child Category Collection	1	A collection of the Category child Categories.
	ITEMS	Collection of	1	A collection of all the content items

		Multimedia Items		that this
--	--	------------------	--	-----------

COMVERSE Multimedia Item and Media Items collection – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	ITEMS	A Collection of ITEM items		
ITEMS	total	Number	1	The number of the items in the collection.
	ITEM	Multimedia Item	0..n	
ITEM	id	Number	1	The unique id of the item
	external_id	String	1	The non-COMVERSE id of an item (such as id of items from different system)
	type	?	1	Type of the content
	language	Enum	1	The language of the multimedia content
	owner	String	1	Name of the woner
	title	String	1	The title of the multimedia content
	artist	?	1	The artist that credited for the multimedia content
	length	Number	1	Length, in seconds, of the multimedia content
	billing_category	Enum	1	Always AXMEDIS
	billing_description	String	1	Always AXMEDIS
	expiration	Number	1	Number of days till the content expires.
	FILES		1	Files that are referenced by the item
	META INFO		1	Meta info that is related to the item

4.4.2.5 Configuration – Detailed Structure**Log Configuration – Detailed Structure**

Parent	Item	Type	Occur	Value/Comments
	LOGS	Element		List of logs
LOGS	LOG	Element	0..n	Specific Log Configuration
LOG	Id	Attribute	0..1	Optional, Unique ID of the Log
	type	Attribute	1	“file” only
	name	Attribute	1	General: Name of the write target device or container. Specific: Log file name to write to, optionally proceeded by the destination path. No path or relative path shall be in relation to the path of the application executable file. e.g. “../log/eventlog.log”
	EVENTS	Element	1	List of events to write to this log.
	FIELDS	Element	1	List of data fields to log for the logged events.
EVENTS	EVENT	Element	1..n	An event type to log
EVENT	name	Attribute	1	The name of the event to log. (e.g. “error”)
	log	Attribute	1	“true” to log the event; “false” to not log the event.
FIELDS	FIELD	Element	1..n	A data field to log for each event.

FIELD	name	Attribute	1	Name of data field (e.g. "date time")
	log	Attribute	1	"true" to log this field data; "false" to not log this field data

Example: HME Logs configuration:

```

<LOGS>
  <LOG id="event" type="file" name="../log/event.log">
    <EVENTS>
      <EVENT name="application_start " log="true"/>
      <EVENT name="application_restart" log="true"/>
      <EVENT name="application_shut_down" log="true"/>
      <EVENT name="request_error" log="true"/>
      <EVENT name="device_fallback" log="true"/>
      <EVENT name="device_default" log="true"/>
      <EVENT name="request" log="false"/>
    </EVENTS>
    <FIELDS>
      <FIELD name="date_time" log="true"/>
      <FIELD name="request" log="true"/>
    </FIELDS>
  </LOG>
</LOGS>

```

4.4.2.6 HMS/HME Configuration – Detailed Structure

Device Capabilities Configuration – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	DEVICE_CAPABILITIES	Element		List of DEVICES items
DEVICE_CAPABILITIES	DEVICES	Element	1..n	List of DEVICE items or reference to a DEVICES resource.
	DEFAULT_DEVICE	Element	1	Refers to the default device for all devices from all DEVICES sources that are grouped in the DEVICE_CAPABILITIES item.
DEVICES	name	Attribute	0..1	If the name attribute is specified, the application shall expect an external resource for the list of devices. Otherwise the application shall expect device child nodes, if any. The attribute value shall refer to the source of the resource. e.g. "devices/mobile_phones/nokia.xml".
	type	Attribute	0..1	The only supported value is "file". The type attribute is relevant only if the name attribute is specified.
DEFAULT_DEVICE	id	Attribute	1	The ID value of a device node in the devices list.

Example: HME device capabilities configuration refers to external file resource:

```

<DEVICE_CAPABILITIES>
  <DEVICES type="file" name="../dat/device_capabilities_all.xml"/>
  <DEFAULT_DEVICE id="generic"/>

```

```
</DEVICE_CAPABILITIES>
```

HMS/HME Configuration – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	HANDSET_MANAGEMENT_SERVICE	Element		Handset Management Service configuration root node.
HANDSET_MANAGEMENT_SERVICE	HANDSET_MANAGEMENT_ENGINE	Element	1	Handset Management Engine node
HANDSET_MANAGEMENT_ENGINE	DEVICE_CAPABILITIES	Element	1	Device Capabilities data and parameters
	LOGS	Element	1	Required Logs

Example: The HMS configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<HANDSET_MANAGEMENT_SERVICE>
  <HANDSET_MANAGEMENT_ENGINE>
    <DEVICE_CAPABILITIES>
      ...
    </DEVICE_CAPABILITIES>
    <LOGS>
      ...
    </LOGS>
  </HANDSET_MANAGEMENT_ENGINE>
</HANDSET_MANAGEMENT_SERVICE>
```

4.4.2.7 Device Capabilities Data – Detailed Structure

Device Capabilities Data – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	WURFL	Element		Wireless Universal Resource File container
WURFL	VERSION	Element	1	COMVERSE standard request name. Always “GET_DEVICE_CAPABILITIES”.
	DEVICES	Element	1	
VERSION	*	Any	0..n	Any text, attributes or child elements. This section is informative only.
DEVICES	DEVICE	Element	1..n	A device node (See Device Node – Capabilities Data – Detailed Structure)

Example: General structure of the Device Capabilities data.

```
<?xml version="1.0" encoding="utf-8" ?>
<WURFL>
  <VERSION>
    Version 1.0
  </VERSION>
```

```

<DEVICES>
  <DEVICE>
    <GROUP>
      <CAPABILITY name="[cap_name]" value="[cap_value]">
    </GROUP>
  </DEVICE>
  ...
</DEVICES>
</WURFL>

```

Device Node – Capabilities Data – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	DEVICE	Element		A device node
DEVICE	id	Attribute	1	The unique ID of the specific device
	user_agent	Attribute	1	The user_agent of the device
	actual_device_root	Attribute	0..1	“true” if the device is a device to fall back to from a non-root device.
	fall_back	Attribute	1	The ID of another device to fall back to. This element also available on root device to allow falling back to generic capability specifications.
	GROUP	Element		A group of capabilities that fall under the same category (e.g. audio formats, mms, markup, etc).
GROUP	id	Attribute		The unique category name or of the group
GROUP	CAPABILITY	Element		A single capability
CAPABILITY	name	Attribute		The name of the capability (e.g. video_max_height, mms_mp4, etc)
	value	Attribute		The value of the capability. Typically the value is “true” if the capability is supported, or “false” if not. Some capabilities require measure units (dimensions, bitrate, memory size, color, etc) or string (e.g. init call string)
	Note that a Device node structure is not restricted to capabilities data. It may also include other elements, attributes, texts, etc.			

Example: Device capabilities of the root device Nokia 3125, listing the various types, download, formats, and other capabilities of the device, grouped by capability categories.

```

<DEVICE
  id="nokia_3125_ver1"
  user_agent="Nokia3125"
  actual_device_root="true"
  fall_back="nokia_generic_series40">

```

```

<GROUP id="product_info">
  <CAPABILITY
    name="model_name"
    value="3125"/>
</GROUP>
<GROUP id="markup">
  <CAPABILITY
    name="preferred_markup"
    value="html_wi_oma_xhtmlmp_1_0"/>
  <CAPABILITY
    name="html_wi_w3_xhtmlbasic"
    value="true"/>
  <CAPABILITY
    name="html_wi_oma_xhtmlmp_1_0"
    value="true"/>
</GROUP>
<GROUP id="object_download">
  <CAPABILITY
    name="ringtone_amr"
    value="true"/>
  <CAPABILITY
    name="ringtone_mp3"
    value="true"/>
  <CAPABILITY
    name="ringtone_aac"
    value="true"/>
  <CAPABILITY
    name="video"
    value="true"/>
  <CAPABILITY
    name="video_3gpp"
    value="true"/>
  <CAPABILITY
    name="video_mp4"
    value="true"/>
  <CAPABILITY
    name="video_max_width"
    value="128"/>
  <CAPABILITY
    name="video_max_height"
    value="96"/>
  <CAPABILITY
    name="video_preferred_width"
    value="128"/>
  <CAPABILITY
    name="video_preferred_height"
    value="96"/>
  <CAPABILITY
    name="video_directdownload_size_limit"
    value="102400"/>
</GROUP>
<GROUP id="mms">
  <CAPABILITY
    name="mms_video"
    value="true"/>
  <CAPABILITY
    name="mms_mp4"
    value="true"/>

```



```

    <CAPABILITY
      name="mms_3gpp"
      value="true"/>
  </GROUP>
  <GROUP id="xhtml_ui">
    <CAPABILITY
      name="xhtml_supports_table_for_layout"
      value="true"/>
    <CAPABILITY
      name="xhtml_supports_css_cell_table_coloring"
      value="true"/>
    <CAPABILITY
      name="xhtml_readable_background_color1"
      value="#99CCFF"/>
    <CAPABILITY
      name="xhtml_readable_background_color2"
      value="#FFFFFF"/>
    <CAPABILITY
      name="xhtml_make_phone_call_string"
      value="wtai://wp/mc;"/>
    <CAPABILITY
      name="xhtml_table_support"
      value="true"/>
  </GROUP>
</DEVICE>

```

4.4.2.8 HMS/HME Protocol – Detailed Structure

Get Device Capabilities Request – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RQST	Element		COMVERSE standard Request header
RQST	name	Attribute	1	COMVERSE standard request name. Always “GET_DEVICE_CAPABILITIES”.
	DEVICES	Element	1	
DEVICES	DEVICE	Element	1	Desired Device node
DEVICE	id	Attribute	0..1	The ID of the device
	user_agent	Attribute	0..1	The user agent string as identified by the device.
	model_name	Attribute	0..1	The device model name (e.g. “3100” for Nokia 3100 series)
	manufacturer	Attribute	0..1	The name of the device manufacturer.

Example: Request to get the capabilities of the root device Nokia 3100 version 1.0, by ID:

```

<?xml version="1.0" encoding="utf-8" ?>
<RQST name="GET_DEVICE_CAPABILITIES">
  <DEVICES>
    <DEVICE id="nokia_3100_ver1"/>
  </DEVICES>
</RQST>

```

Example: Request to get the capabilities of the non-root device Nokia 3100 sub version 0554, by user agent. If not found, it will fall search the general manufacturer (Nokia) device.

```

<?xml version="1.0" encoding="utf-8" ?>
<RQST name="GET_DEVICE_CAPABILITIES">
  <DEVICES>
    <DEVICE
      manufacturer="nokia"
      model_name="3100"
      user_agent="Nokia3100/1.0" (05.54) Profile/MIDP-1.0
    </DEVICE>
  </DEVICES>
</RQST>

```

Example: Request to get the capabilities with no parameters. The search shall fallback to return the default device capabilities.

```

<?xml version="1.0" encoding="utf-8" ?>
<RQST name="GET_DEVICE_CAPABILITIES">
  <DEVICES>
    <DEVICE/>
  </DEVICES>
</RQST>

```

Get Device Capabilities Reply – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RPLY	Element		COMVERSE standard Reply header
RPLY	name	Attribute	1	COMVERSE standard request name. Always "GET_DEVICE_CAPABILITIES".
	ERROR	Element	0..1	Error block; Describes the error(s) that occurred while the request was processed, warnings to be considered, and device-fallback info.
	DEVICES	Element	1	Devices block
DEVICES	DEVICE	Element	1	A standard DEVICE node, with the capabilities groups and info. (See Device Node – Capabilities Data – Detailed Structure)

Example: A possible reply for the Get Device Capabilities request, listing the capabilities of device *nokia 3125 ver1*, and a fallback info in the error block.

```

<?xml version="1.0" encoding="utf-8" ?>
<RPLY name="GET_DEVICE_CAPABILITIES">
  <ERROR
    id="HMS0222"
    description="Device Fall Back">
    <ERRORS>
      <ERROR
        id="HMS0205"
        description="Search for device by Model yielded multiple
results">
      </ERROR>
    </ERRORS>
  </ERROR>
  <DEVICES>
    <DEVICE
      user_agent="Nokia3125"
      actual_device_root="true"
    </DEVICE>
  </DEVICES>
</RPLY>

```

```

fall_back="nokia_generic_series40"
id="nokia_3125_ver1">

<GROUP id="product_info">
  <CAPABILITY
    name="model_name"
    value="3125"/>
</GROUP>
<GROUP id="markup">
  <CAPABILITY
    name="preferred_markup"
    value="html_wi_oma_xhtmlmp_1_0"/>
  <CAPABILITY
    name="html_wi_w3_xhtmlbasic"
    value="true"/>
  <CAPABILITY
    name="html_wi_oma_xhtmlmp_1_0"
    value="true"/>
</GROUP>
<GROUP id="object_download">
  <CAPABILITY
    name="ringtone_amr"
    value="true"/>
  <CAPABILITY
    name="ringtone_mp3"
    value="true"/>
  <CAPABILITY
    name="ringtone_aac"
    value="true"/>
  <CAPABILITY
    name="video"
    value="true"/>
  <CAPABILITY
    name="video_3gpp"
    value="true"/>
  <CAPABILITY
    name="video_mp4"
    value="true"/>
  <CAPABILITY
    name="video_max_width"
    value="128"/>
  <CAPABILITY
    name="video_max_height"
    value="96"/>
  <CAPABILITY
    name="video_preferred_width"
    value="128"/>
  <CAPABILITY
    name="video_preferred_height"
    value="96"/>
  <CAPABILITY
    name="video_directdownload_size_limit"
    value="102400"/>
</GROUP>
<GROUP id="mms">
  <CAPABILITY
    name="mms_video"
    value="true"/>
  <CAPABILITY

```

```

        name="mms_mp4"
        value="true"/>
    <CAPABILITY
        name="mms_3gpp"
        value="true"/>
</GROUP>
<GROUP id="xhtml_ui">
    <CAPABILITY
        name="xhtml_supports_table_for_layout"
        value="true"/>
    <CAPABILITY
        name="xhtml_supports_css_cell_table_coloring"
        value="true"/>
    <CAPABILITY
        name="xhtml_readable_background_color1"
        value="#99CCFF"/>
    <CAPABILITY
        name="xhtml_readable_background_color2"
        value="FFFFFF"/>
    <CAPABILITY
        name="xhtml_make_phone_call_string"
        value="wtai://wp/mc;"/>
    <CAPABILITY
        name="xhtml_table_support"
        value="true"/>
</GROUP>
</DEVICE>
</DEVICES>
</RPLY>

```

Restart HME Request – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RQST	Element		COMVERSE standard Request header
RQST	name	Attribute	1	COMVERSE standard request name. Always “HME RESTART”.

Example: Request to Restart the HME:

```

<?xml version="1.0" encoding="utf-8" ?>
<RQST name="HME_RESTART">
</RQST>

```

Restart HME Reply – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RPLY	Element		COMVERSE standard Reply header
RPLY	name	Attribute	1	COMVERSE standard request name. Always “HME RESTART”.
	ERROR	Element	1	Error block; Describes success of the operation or the errors that occurred while processing the request.

Example: Possible reply to a Request to restart the HME; the request was processed and the HME was successfully restarted:

```
<?xml version="1.0" encoding="utf-8" ?>
<RQST name="HME_RESTART">
  <ERROR id="HMS0003" description="HME Restarted Successful"/>
</RQST>
```

Shut Down HME Request – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RQST	Element		COMVERSE standard Request header
RQST	name	Attribute	1	COMVERSE standard request name. Always “HME_SHUT_DOWN”.

Example: Request to Shut Down the HME:

```
<?xml version="1.0" encoding="utf-8" ?>
<RQST name="HME_SHUT_DOWN">
</RQST>
```

Shut Down HME Reply – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RPLY	Element		COMVERSE standard Reply header
RPLY	name	Attribute	1	COMVERSE standard request name. Always “HME_SHUT_DOWN”.
	ERROR	Element	1	Error block; Describes success of the operation or the errors that occurred while processing the request.

Example: Possible reply to a Request to Shut Down the HME; the request was processed and the HME was successfully Shut Down:

```
<?xml version="1.0" encoding="utf-8" ?>
<RQST name="HME_SHUT_DOWN">
  <ERROR id="HMS2003" description="HME Shut Down Successful"/>
</RQST>
```

4.4.2.9 PS/PE Protocol – Detailed Structure

Restart PE Request – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RQST	Element		COMVERSE standard Request header
RQST	name	Attribute	1	COMVERSE standard request name. Always “PE_RESTART”.

Example: Request to Restart the PE:

```
<?xml version="1.0" encoding="utf-8" ?>
<RQST name="PE_RESTART">
</RQST>
```

Restart PE Reply – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RPLY	Element		COMVERSE standard Reply header
RPLY	name	Attribute	1	COMVERSE standard request name. Always “PE_RESTART”.
	ERROR	Element	1	Error block; Describes success of the operation or the errors that occurred while processing the request.

Example: Possible reply to a Request to restart the PE; the request was processed and the PE was successfully restarted:

```
<?xml version="1.0" encoding="utf-8" ?>
<RQST name="PE_RESTART">
  <ERROR id="HMS0303" description="PE Restarted Successful"/>
</RQST>
```

Shut Down PE Request – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RQST	Element		COMVERSE standard Request header
RQST	name	Attribute	1	COMVERSE standard request name. Always “PE_SHUT_DOWN”.

Example: Request to Shut Down the PE:

```
<?xml version="1.0" encoding="utf-8" ?>
<RQST name="PE_SHUT_DOWN">
</RQST>
```

Shut Down PE Reply – Detailed Structure

Parent	Item	Type	Occur	Value/Comments
	RPLY	Element		COMVERSE standard Reply header
RPLY	name	Attribute	1	COMVERSE standard request name. Always “PE_SHUT_DOWN”.
	ERROR	Element	1	Error block; Describes success of the operation or the errors that occurred while processing the request.

Example: Possible reply to a Request to Shut Down the PE; the request was processed and the PE was successfully Shut Down:

```
<?xml version="1.0" encoding="utf-8" ?>
<RQST name="HME_SHUT_DOWN">
  <ERROR id="HMS2003" description="HME Shut Down Successful"/>
</RQST>
```

</RQST>

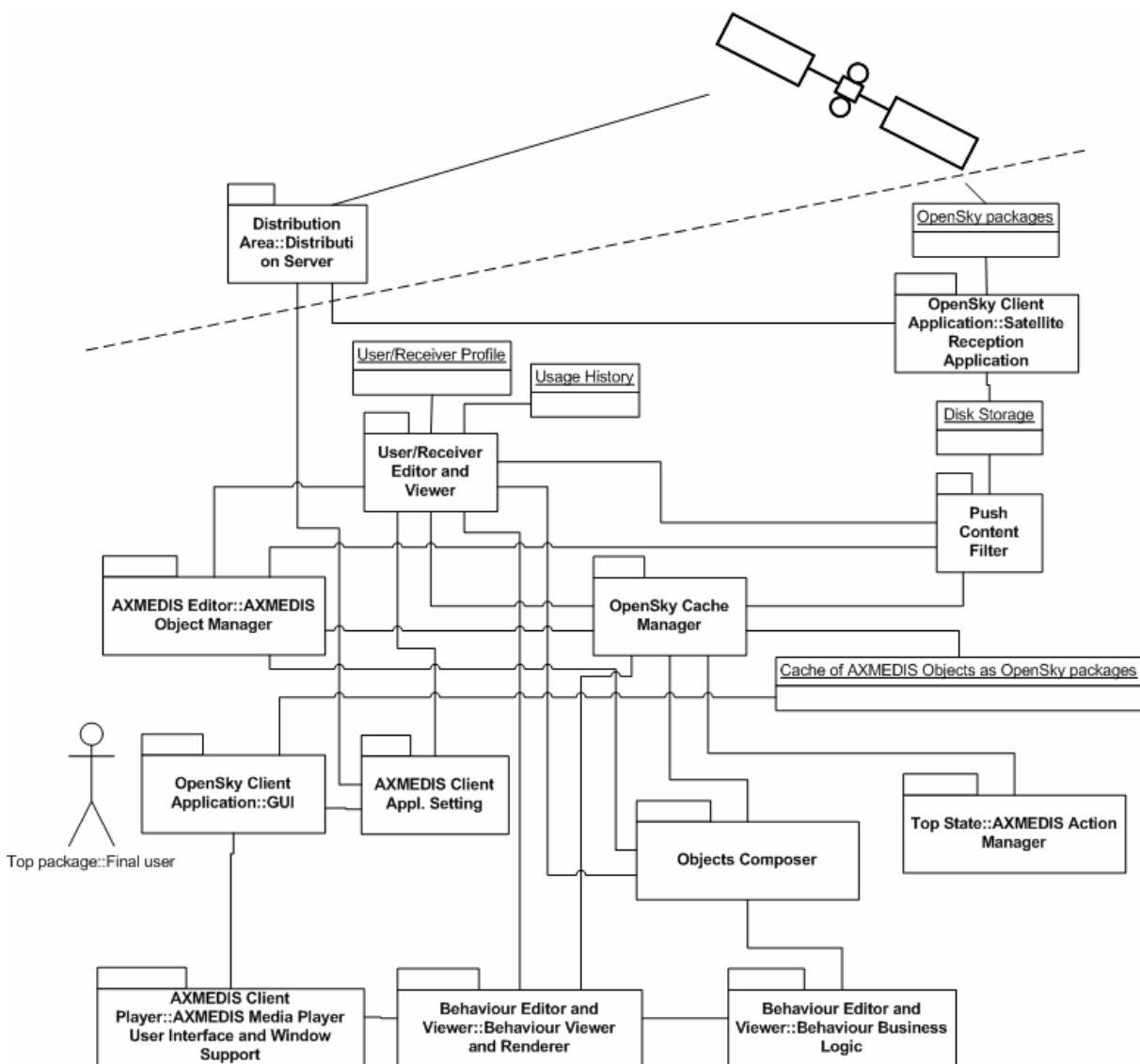
4.5 Supporting protection and security with AXMEDIS support (COMVERSE)

All Distribution and Transcoding modules, AXMEDIS Plugin and AXEPTTool included, shall be deployed on the same network environment, Firewall protected. The communication between all modules shall rely on HTTPS for security reason. The content that is stored or cached can be accessed only by the using applications and only for a specific purpose (e.g. AX Plugin can only write to cache storage, Only the APS can only read from the cache storage for distribution purposes only, AXEPTTool is the only entity that can access the DB In/Out).

COMVERSE has not contributed as expected to this deliverable. In their distribution channel, presently, there is not evidence or report to the content owner about how many objects have been sold or at least this number is only estimated counting transactions in a NON AXMEDIS compliant environment. The number provided has to be trusted by the content owner while the objects are distributed without monitoring any transaction or distribution with AXMEDIS tools. In addition, the so called Transcoding tool has not been described neither integrated on paper with the AXMEDIS tools.

This means that the COMVERSE distribution solution cannot get in some sense the “AXMEDIS certification”, that means that their solution presently is not AXMEDIS oriented since it does not guarantee the safeness and trace ability of the activities performed. This problem has to be solved to provide the right demonstrator of AXMEDIS in WP9.

5 Distribution towards I-TV via satellite (WP4.8, WP9.3: EUTELSAT)



5.1 Passing from AXMEDIS AXInfo to OPENSKY Package on the Server Side (EUTELSAT)

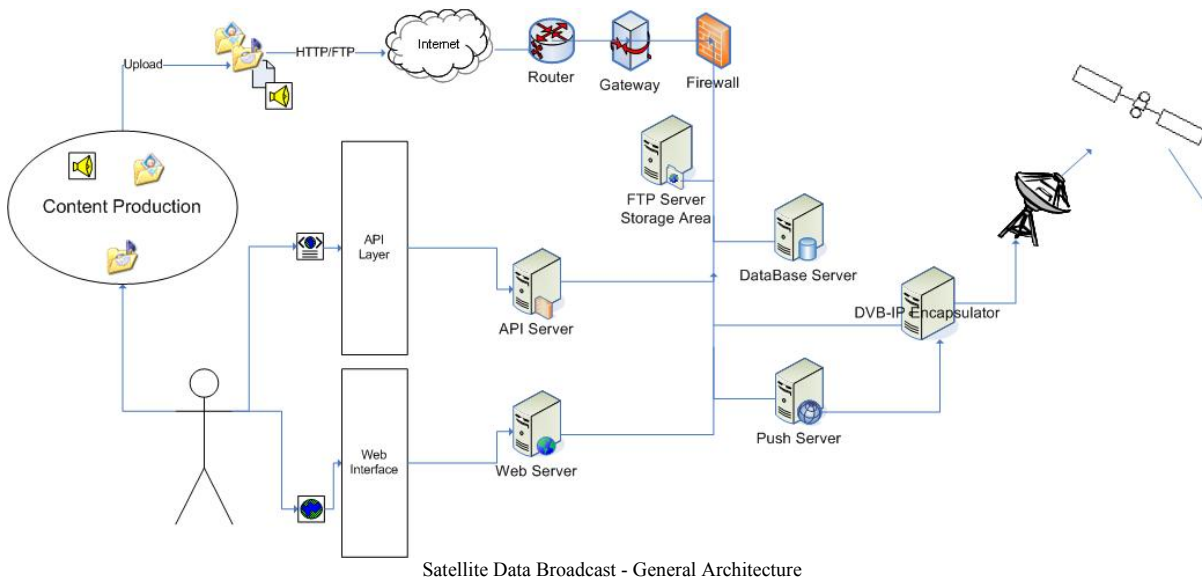
Module Profile	
<name API Server>	
Executable or Library(Support)	API Server
Single Thread or Multithread	Multithread
Language of Development	Java, Python
Responsible Name	Orazio PULVIRENTI
Responsible Partner	EUTELSAT

Status (proposed/approved)	Approved	
Platforms supported	Windows, Linux	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
Push Server, Database		Proprietary: protected
File Formats Used	Shared with	File format name or reference to a section
XML format		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Script based		
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

The Satellite Data Broadcast (SDB) is responsible for the management and transmission of content to be broadcasted to users. The system allows Publishers (Content Provider/Owner) to control the whole workflow of the multicast distribution.

Different modules compose the architecture:

1. FTP Server
2. Web Server
3. API Server
4. Push Server
5. Database Server
6. Encapsulator IP over DVB-S.



The Push Server is the core of the configuration because it is entrusted with multicast transmissions, jobs scheduling, and event signalisation to all connected stations. Furthermore the Push Server interacts with all other components in order to accomplish its mission.

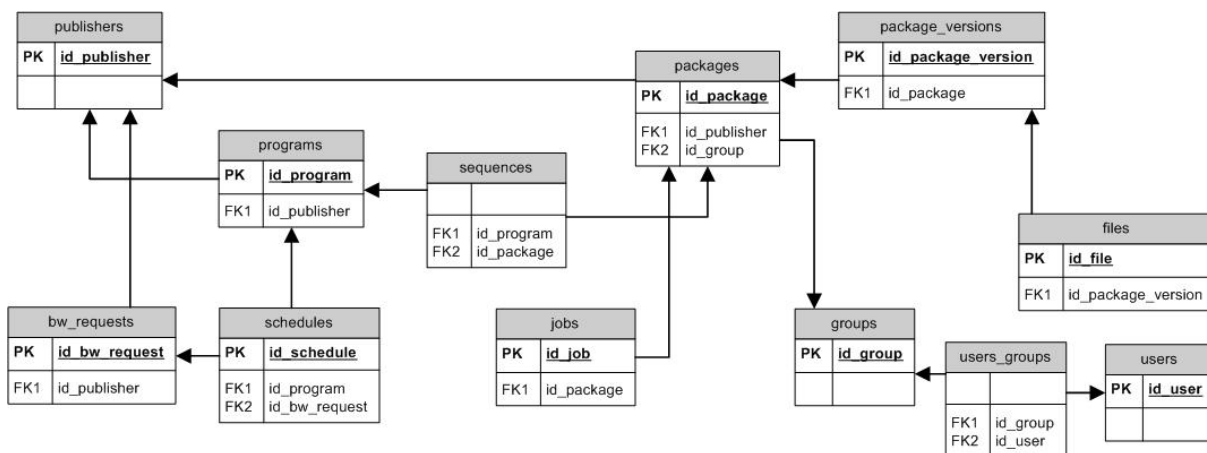
An advanced Content Management System is integrated in the SDB. It uses a transactional Database engine, in order to store all metadata related to Packages, Programs, Publishers, and so on.

The CMS is responsible for:

- Update/Changes of the internal content for a given package;
- Management of different versions of the same package;
- Computing the checksum (CRC32) for guarantee the content integrity.

The main entity used in the SDB is the Package. The Package is the minimal logical unit that can be transmitted in multicast.

When content reaches (via upload) the platform is initially stocked in the Storage Area (FTP Server), after creating a Package (or loading an existing one from the Database), then the content is associated to a Package (updating of version and checksum computation). All content associated to a Package are stored in the Sending Area (Push Server) and they are ready to be associated to an active Program and be sent via satellite.



CMS Database Schema

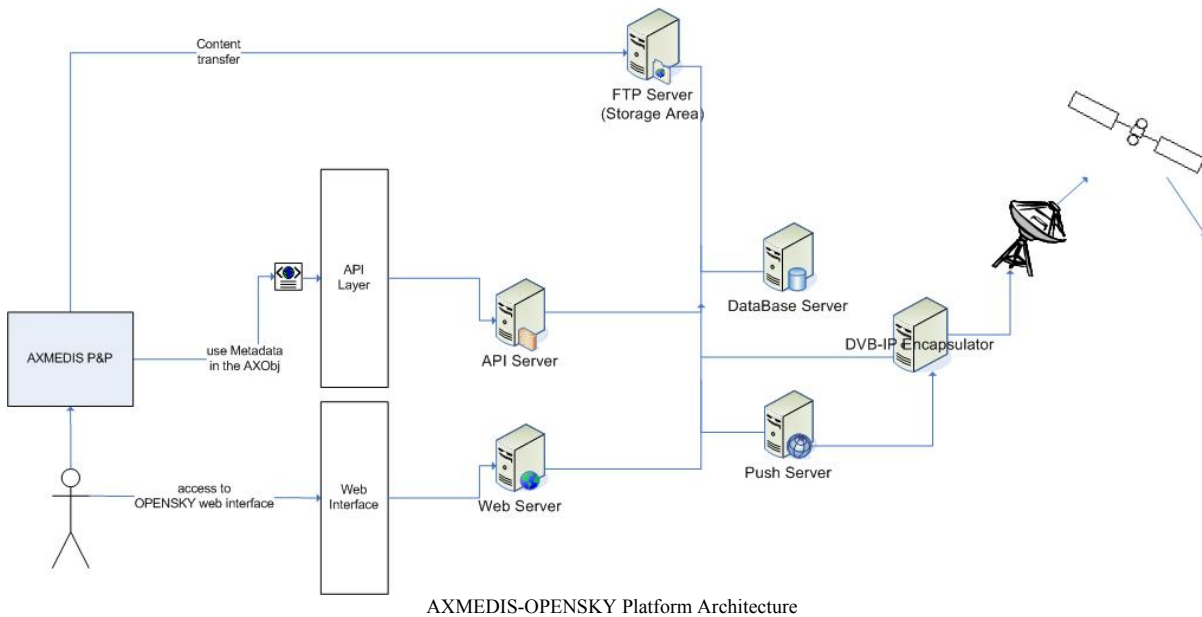
The Broadcast Service Guide (BSG) represents the main communication channel from the server to the client side. The BSG is a file (text or xml) internally generated and simultaneously sent to all connected users, it contains both all metadata giving Packages and Publishers information and the start time of incoming jobs.

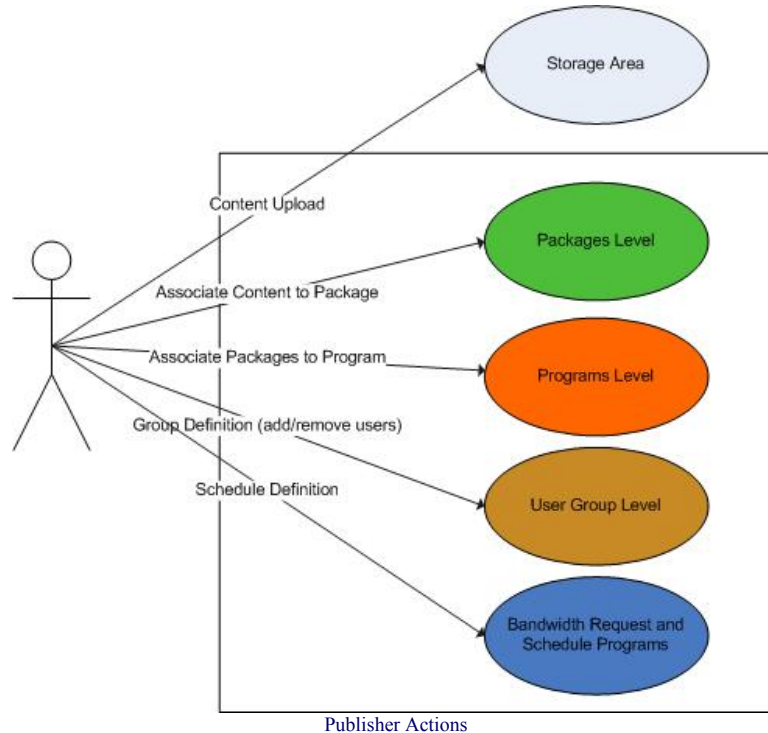
5.1.1 Basic Features

The Satellite Data Broadcast gives to Publishers the total control of their micro platform of multicast file delivery. A Publisher interacts with the system either by web access or by sending XML-structured commands (treated by the API Server).

Each Publisher has access to the Storage Area where he stores his uploaded content. This area is independent from the Push System, and the Publisher accesses it by means of tools like ftp etc.

Once content has been the Storage Area, the Publisher may manage Packages (grouping files taken from his Storage Area), create Programs (grouping Packages previously created), and manage Groups of Users authorised to receive Packages, require some bandwidth and schedule transmissions of his Programs.





The publication of content is very simple and liable. Publishers are able to:

1. Send materials to headquarters
2. Add to a 'carousel' and repeatedly broadcast to all authorized users
3. Determine who can and cannot receive the data
4. Decide to encrypt outgoing PackagesAccess easily the web based interface for Push Managers

Some highlights are now provided to summarize global features of the Satellite Data Broadcast system:

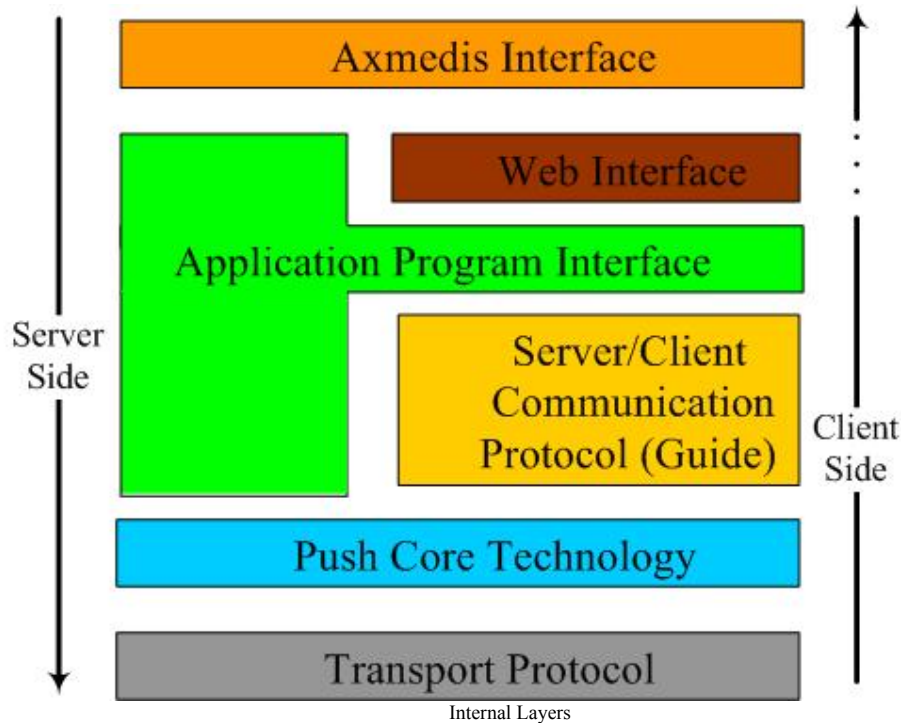
- Multimedia Content can be uploaded by Internet connectionContent Provider decides schedulingTransmission mode is Carousel or One ShotError Correction can be used to recover lost packetsAutomatic update of pushed contentFlexibility in bandwidth allocationCryptography is available for pushed content at transport level (CAS) and at content level (DRM)Group definition to address a content to community of usersClient application is platform independent

5.1.2 Internal Layers

The following section provides an overview of the Satellite Data Broadcast presented per layers, from the bottom:

1. **Transport:** it is responsible of send/receive operations (open socket, create/join multicast group, packetize/reassembly files).
2. **Server/Client Communication (Guide):** it is entrusted with the communication from the server towards all connected clients. It sends the EPG (periodically refreshed) and it manages the Event Signalling Channel, where the server announces all incoming transmission start dates. From the client towards the server, this level manages the packet reparation in unicast.
3. **API:** it is the most important level, because it allows Publishers to directly interact with the Push Technology (it is valid for both server context and client context).
4. **Web Interface:** it offers a graphical web interface to Publishers. It implements its functionalities by calling the lower API Level.

5. **AXMEDIS Interface:** this level manages the integration between AXMEDIS and OPENSKY platform (i.e., this level includes the AXEPTool at the server side and the AXMEDIS Player at the client side).



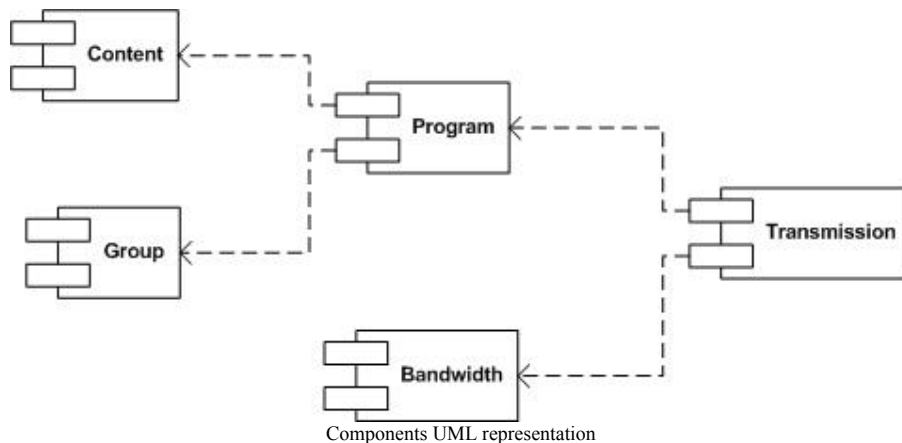
5.1.3 System Structure

The Satellite Data Broadcast has two main actors, the Publisher and the Push Server. The former is responsible of managing the content and of scheduling its transmission; the latter is responsible of the effective transmission.

The Publishers interacts with the system trough a set of API that will provide all the functionalities needed.

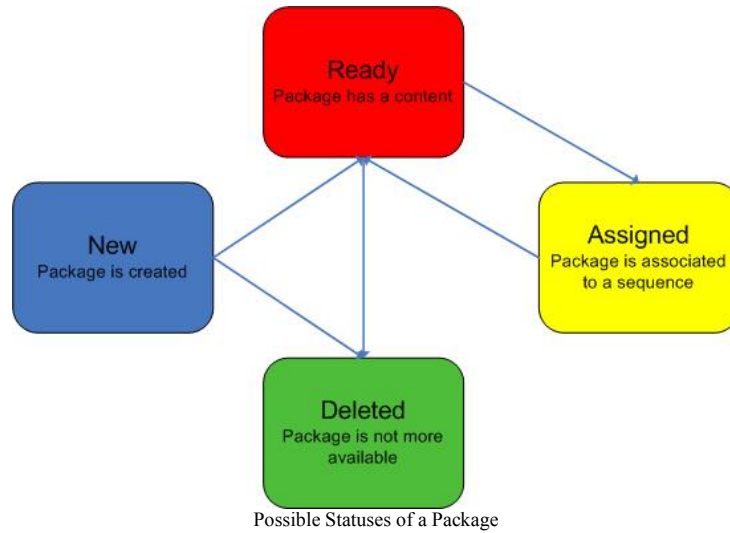
5.1.4 Modules

The main components of the system and their dependencies are depicted on the figure below:



Content Component

- Packages Management (create/delete)



- Package Versions Management
- Package has one or more file(s) in its versions
- Assign Content to Package

Group Component

- Management (create/delete)
- Add/remove users into/from groups
- Used like target in the transmissions
- Possible Types (Roaming):
 - List: dynamically add/remove users from a list
 - File: the user belongs to a group if he has a given group file

Program Component

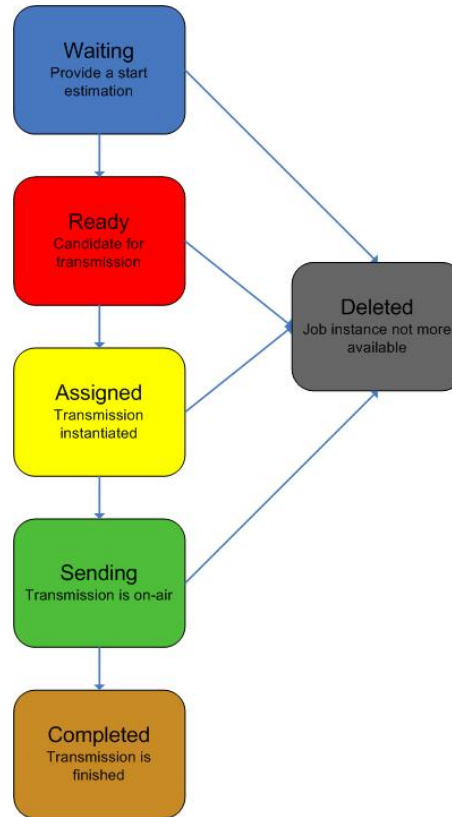
- Management (create/delete)
- Regulate the add/remove of Packages into/from the Sequence

Bandwidth Request

- Manage request of bandwidth
- A *Request* has:
 - Start date
 - Duration
 - Period
 - Repetition
 - Bandwidth

Transmission Component

- Schedule a Program (by using an existing Bandwidth Request)
- Prepare Jobs for transmissions
- Job is an instance of transmission for a given Package.



Possible Statues of a Job

5.1.5 Actors

Publisher (AXMEDIS Distributor/Content Provider)

Create and upload the content
Schedule transmissions

Broadcaster (EUTELSAT using OPENSKEY Platform)

Send content
Owner of technology
Responsible of transport level

Final User

Access to content

Group

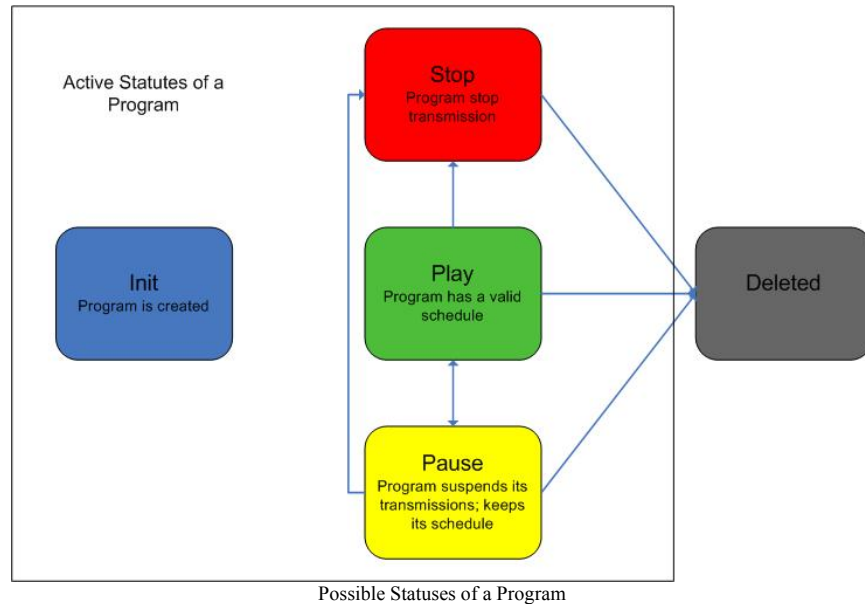
Allow addressing group of users

Package

Minimal logical unit transmission
Contain one or more file

Program/Schedule

Transmit one or more package following a pre-defined sequence
Use the assigned bandwidth to its schedule
Define transmission details (bw,time, # repetition, mode)

**Job**

Transmission instance for a given package

Channel

The Publisher can create one or more **Channel** and he can decide the Channel Name. A Channel contains Active Programs.

The Channel description contains a special template with language, target, and theme. The **Channel Template** has to contain some basic information allowing a simple match with **User Profile** values.

Channel Template Values	User Profile Values
Language	Nationality
Theme	Hobby, Occupation
Target	Age, Sex
Playable	Operating System, IT Knowledge

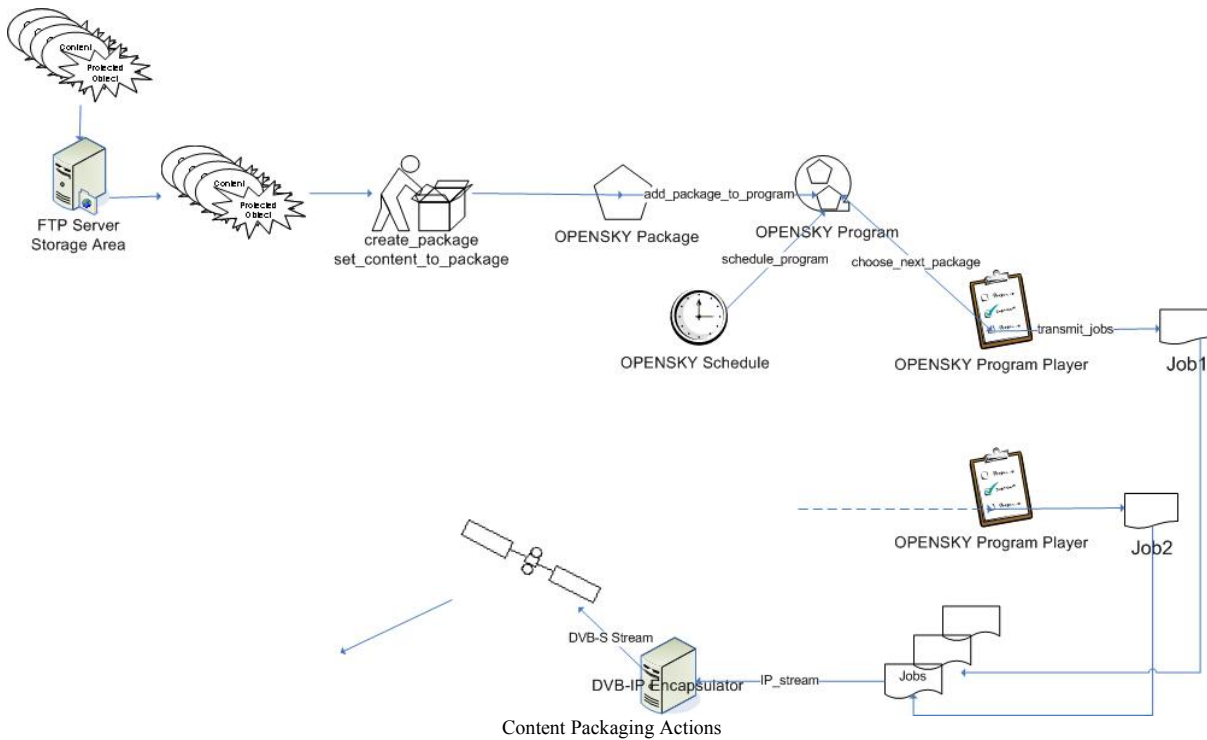
A Channel can be **public** or **private**.

- **Public** – Each user could see the content of this Channel, if the Publisher wants to propose his Channel to him.
- **Private** – A user can receive the content of a Private Channel as per specific request to the Publisher. The Publisher could propose these channels only to particular categories and/or groups of users. Usually Private Channels contain Protected Content. So, they carry out Encrypted Content and the channel reception is connected with a CAS Contract subscription. In any case by using this type of channel, a Publisher can know how many users have asked for his content.

In this architecture Programs are components of Channel's Schedule. A Program can be associated to one or more Channel(s). It means that a Program can be published on different Channels making both simultaneous and asynchronous transmissions.

5.1.6 Content Packaging

The picture below summarizes all steps that an AXMEDIS Publisher has to execute once the AXMEDIS Content is available for distribution. AXMEDIS Objects are uploaded in the OPENSKEY Platform, then they are associated to the OPENSKEY Package, added into an active (because it has a Schedule) OPENSKEY Program and transmitted to all authorized Users.



5.1.7 Content Sending

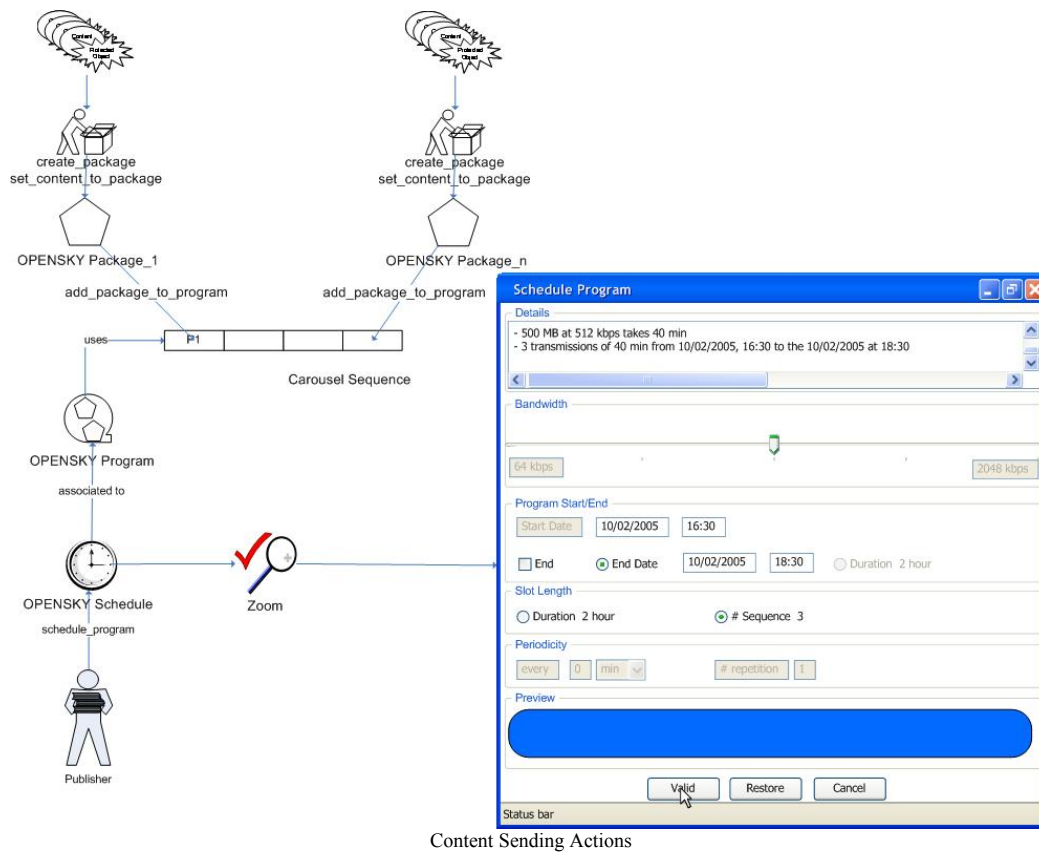
The picture below summarizes all steps that an AXMEDIS Publisher has to execute once the content is packaged in the OPENSKEY format. One or more instances of a single Package can be added to the Carousel Sequence in order to create the Base for the Carousel. Then a Schedule is created taking into account all possible variables:

1. Bandwidth (from 64 Kbps to 20 Mbps)
2. Start Date/ End Date of the Program (during 3 months or 1 hour)
3. Slot Length is a parameter related to the Program duration and the Periodicity. It is less than the Program duration (i.e., program_duration=3 months; slot_length=1 hour)
4. Periodicity is greater than the Slot length value. There is a correlation between Periodicity and Slot length (i.e., slot_length=1 hour, periodicity=1 day, mean that transmissions will be on air 1 hour per day)

Example:

Program_name	=	P1
bandwidth	=	512 Kbps
program_duration	=	3 months
slot_length	=	1 hour
periodicity	=	1 day

Previous values mean that during 3 months there will be an active Program P1, transmitting 1 hour per day at 512 Kbps.

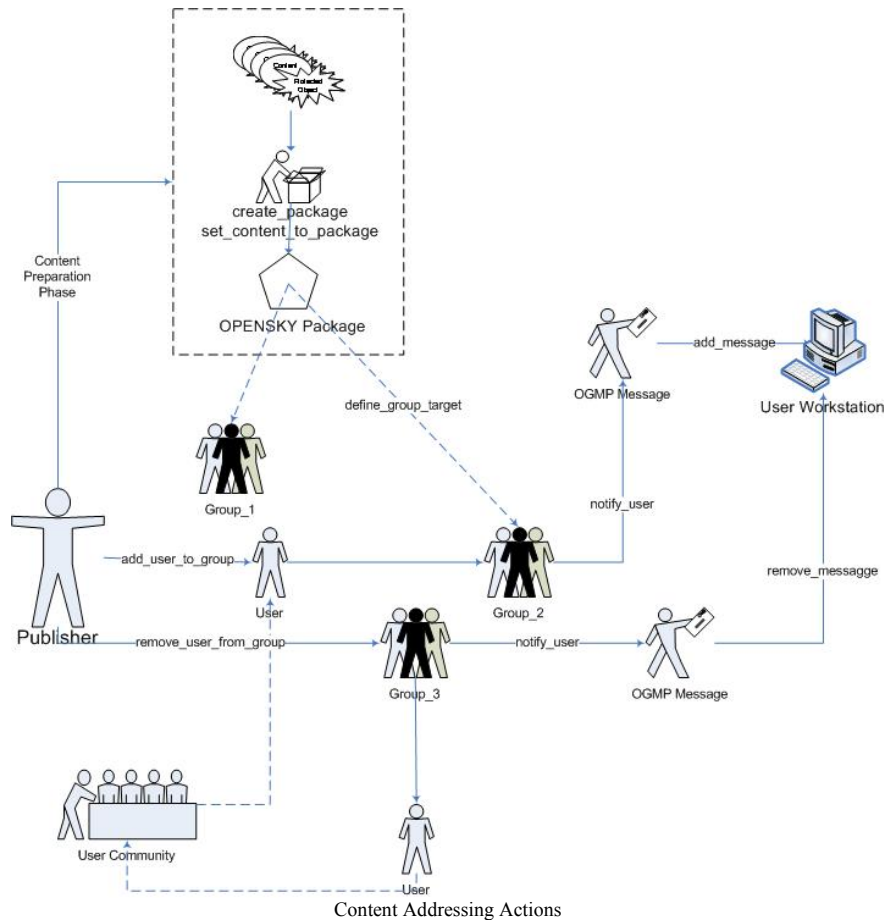


5.1.8 Content Transmission

This module is exclusively internal to the Satellite Data Broadcast system.

5.1.9 Content Addressing

This section defines all criteria and rules about Group Management made by Publishers. A User can be added /deleted to/from a given Group at any moment, each change is immediately notified to the user using the existing OPENSKY Group Management Protocol (OGMP).



5.1.10 From AXInfo to OPENSKEY Package

In the Content Packaging phase a very important step is the passage from the standard AXMEDIS format to the OPENSKEY one.

At the moment of the upload, the content is packaged following the AXMEDIS rules (see part A of this document).

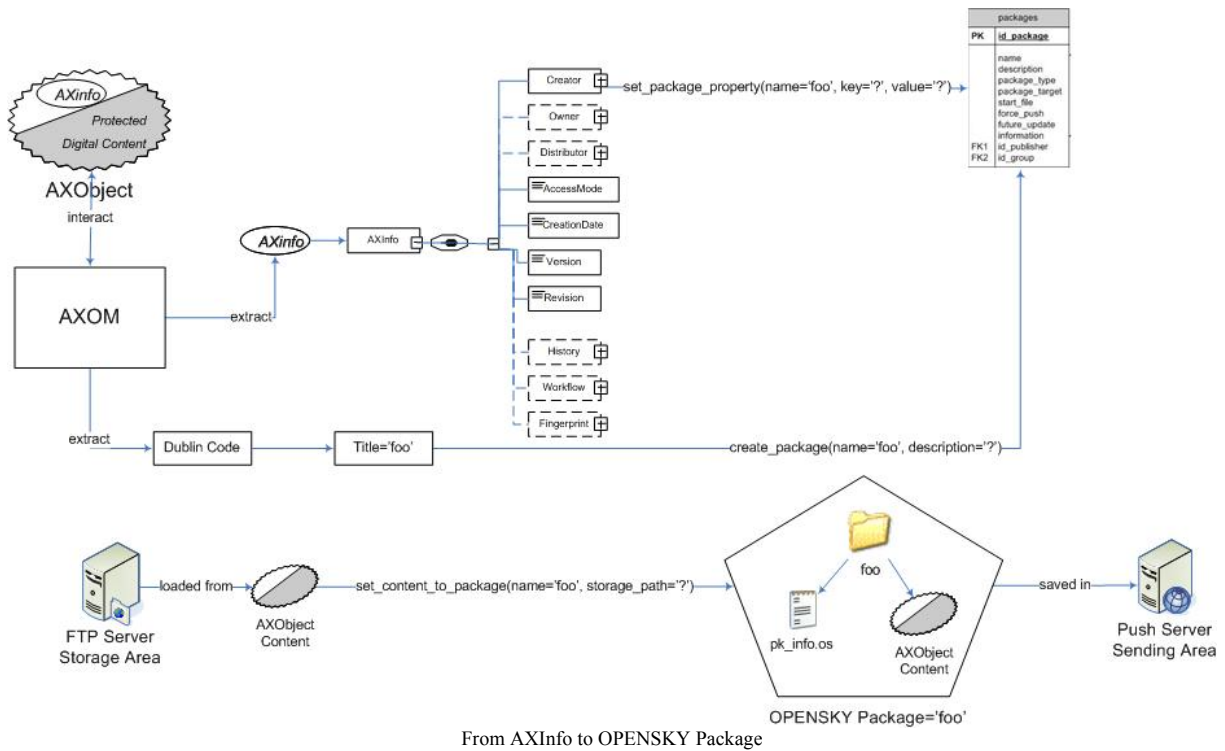
Thanks to the AXOM is possible to manage the uploaded AXObj and extract all needed metadata useful for the OPENSKEY Package creation (name, description): *create_package*.

Then it is possible to assign a series of other metadata to the created Package, based on the Package structure defined in the OPENSKEY CMS Database. A specific API is defined at this scope: *set_package_property*. The list of Package properties is variable and it is possible to add some other metadata by previous evaluation of OPENSKEY technical team.

Finally it is possible to assign the totality of the AXObj (AXInfo is included) to the OPENSKEY Package. This means that the AXObj reaches the designed folder corresponding to the created Package: *set_content_to_package*.

An OPENSKEY Package is constituted by:

- an entry in the Packages table of the CMS Database;
- a folder labelled with the Package Name, containing the AXObj,
- (only at the client side) an additional metadata file called *pk_info.os* (see specific section later)

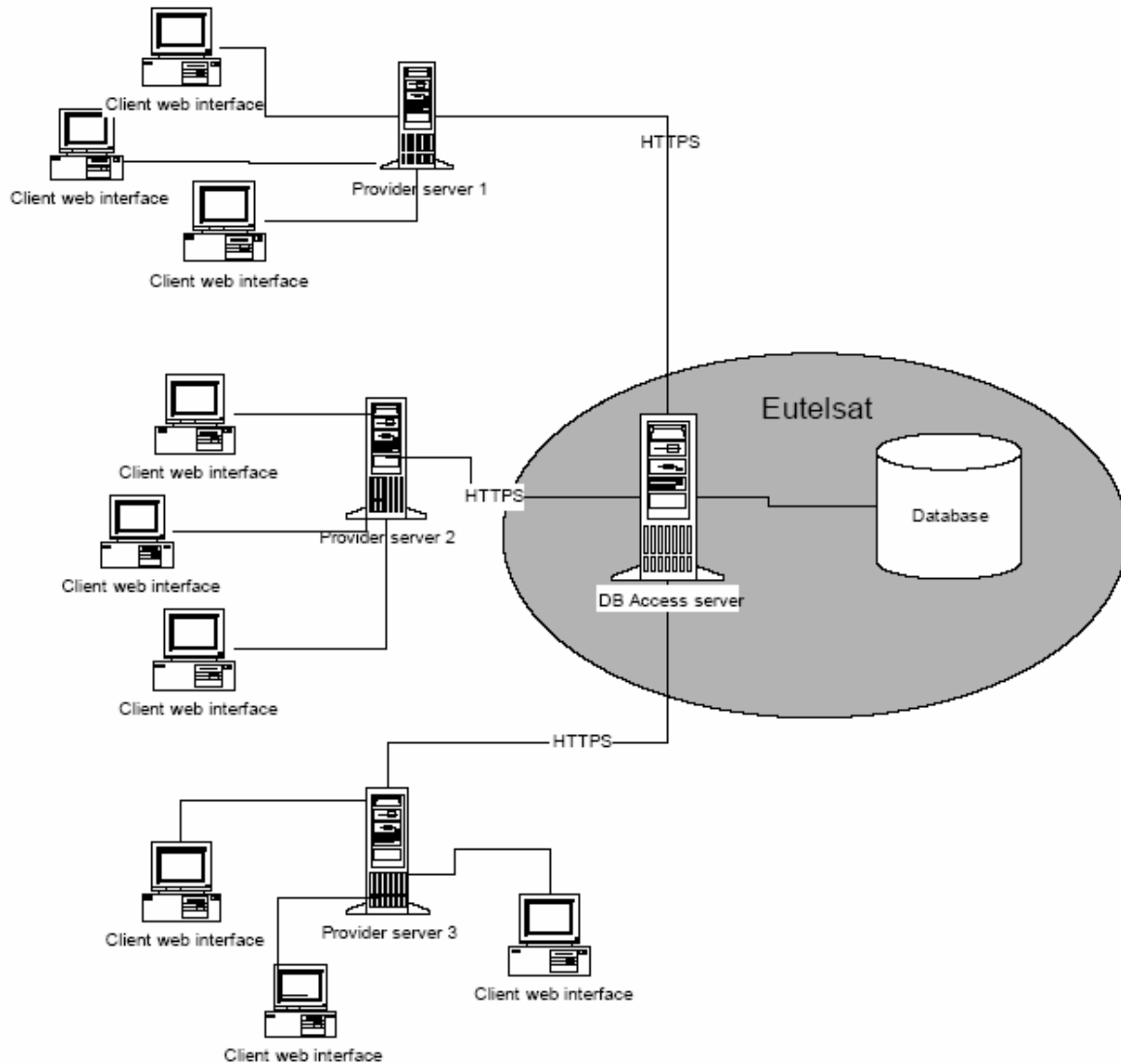


5.1.11 API Server

The communication between the Database access server and the server of a given provider are transmitted using an XML protocol, encapsulated in HTTPS protocol.

The data format is XML.

The structure of the communication architecture is shown in the figure below. The EMP Database clients send request to a server provider that asks for data the Database access server. Both the Database access server and EMP Database are owned and administrated by Eutelsat S.A.

**Example:**

The following lines provide an example of how calling a method of Push Server API, and how parse the response of the API Server.

A complete set of public functions, possible responses (successful/unsuccessful) are contained in the document *API Push Server public functions* [APS].

Definition: create_package

Function to create a package.

Params	Notes
name	Package name
description	Package description

Request

```

<dbrequest>
  <administrator login="my_login" password="my_password" />
  <operation name="create_package">

```

```

        <param name="name">packagename</param>
        <param name="description">the package
description</param>
    </operation>
</dbrequest>

```

Response

```

<?xml version="1.0" encoding="UTF-8" ?>
<response>
  <operation name="create_package">
    <row>
      <code>0</code>
      <message>Package created</message>
    </row>
  </operation>
</response>

```

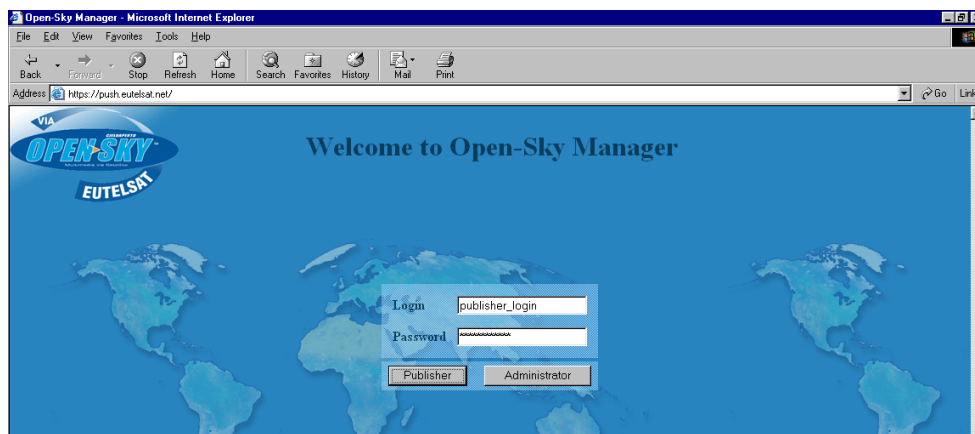
5.1.12 Web Site

The Open-Sky Content Manager Interface is the web interface dedicated to Publishers. They schedule their transmissions, choose the way to broadcast their files, stop or start the broadcast according to their choice. They administrate their own account in the frame of their Agreement with EUTELAST SA.

Publishers can upload files to be broadcast directly on the web site. For any question and troubleshooting please contact publishers@opensky.eutelsat.net.

In the following sections, you will find all the procedure to send your content to your destination using the Open-Sky Content Manager Interface. The next section is describing all the steps to follow to broadcast a file. The last section is more organized like a FAQ.

The URL of the Open-Sky Content Manager Interface is: <https://push.eutelsat.net>



Home Page of the Web Interface

Refer to the OPENSKEY Push Web Interface Publisher Guide [OWP] document for a full description of the web interface structure (OPENSKEY Push_Web_Interface_Publisher_Guide.pdf)

5.1.13 Broadcast Service Guide

The Broadcast Service Guide (BSG) is a file periodically regenerated and refreshed (very short time interval) by the system. Each change of the internal content of the BSG is immediately notified to all connected users.

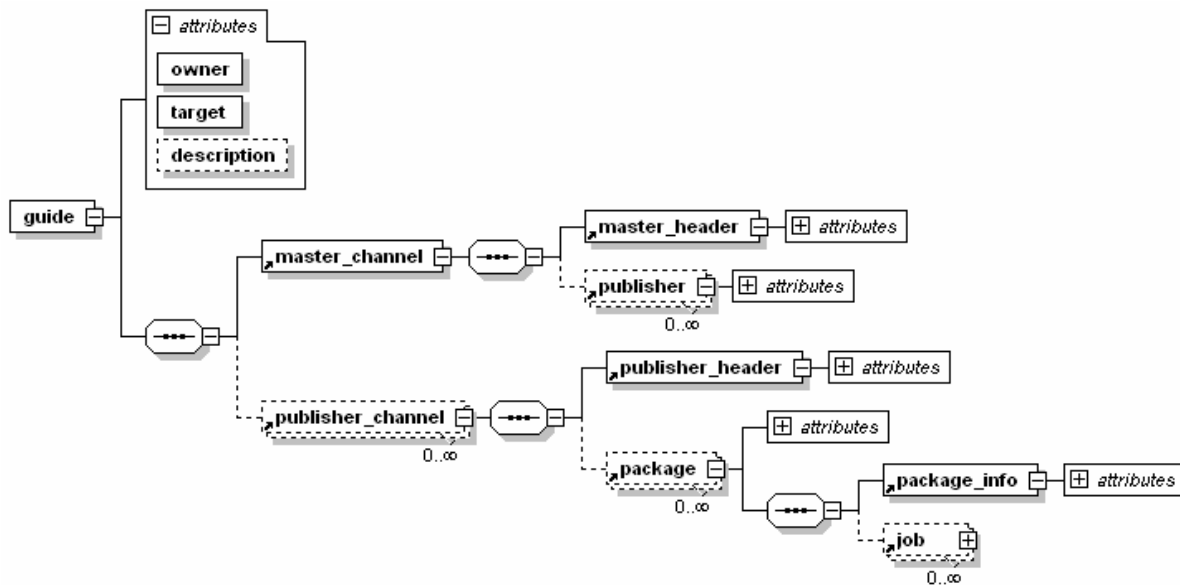
The BSG is the main way for the server to communicate with clients.

The internal data are open and available for all OPENSKEY partners that are free to parse this information in order to develop their own GUIs. These operations are possible using specified API expressly designed for that task.

An Event Signalling Channel is going to be developed.

In practice the BSG will be split in two parts:

- Static Part containing all metadata related to on-air Packages. An XML file that will be compressed and sent to users will represent it. Moreover, since it will not contain any time related information
 - it will be regenerated less frequently;
 - it will contain more information, like Object previews and (if it possible) part of the information contained in the AXInfo.
- Time Critical Part containing all essential information needed for the content reception. The load of this channel should be kept as low as possible, this will lead to frequent refresh of the internal data and rapid notification of incoming events towards all connected users.



W3C Schema of the BSG

Example:

In the following a real file representing an instance of BSG, sent to connected Users and then parsed from the Client Application:

```
<guide description="Broadcast Service Guide for OPENSKEY Users" owner="OPENSKEY Business Unit
Copyright 2003" target="OPENSKEY Users">
  <master_channel>
    <master_header client_version="1.2.7" publishers_amount="4"
server_time="1049711248768" servlets_url="http://push.mm.eutelsat.net/pushservlets/" system_status="o"
update_flag="0"/>
    <publisher bandwidth="1024" carousel_flag="1" description="" email="info@deagostini.it"
frequency="10750" id="22" multicast_address="224.79.80.5" name="DeAgostini" pid="2401"
polarization="H" public_flag="1" satellite="Atlantic Bird 1 12.5 W" size="0" symbol_rate="27500"
txp="eutelsat-local-test.2401" web_site="www.deagostini.it"/>
    <publisher bandwidth="1024" carousel_flag="1" description="Euphon content"
email="info@euphonweb.com" frequency="10750" id="20" multicast_address="224.79.80.2"
name="Euphon" pid="2401" polarization="H" public_flag="1" satellite="Atlantic Bird 1 12.5 W"
size="475" symbol_rate="27500" txp="eutelsat-local-test.2401" web_site="www.euphonweb.com"/>
    <publisher bandwidth="1024" carousel_flag="1" description="Eutelsat S.A. is one of the
world's most established and experienced providers of satellite communications resources and expertise on a
global basis." email="opensky@opensky.eutelsat.net" frequency="10750" id="7"
multicast_address="224.79.80.7" name="Eutelsat" pid="2401" polarization="H" public_flag="1"
satellite="Atlantic Bird 1 12.5 W" size="1348248231" symbol_rate="27500" txp="eutelsat-local-test.2401"
web_site="www.eutelsat.net"/>
    <publisher bandwidth="1024" carousel_flag="1" description="Stay informed with the latest
Reuters News : stories, images and more of all, Videos News." email="" frequency="10750" id="25"
multicast_address="224.79.80.9" name="Reuters" pid="2401" polarization="H" public_flag="1"
satellite="Atlantic Bird 1 12.5 W" size="56501303" symbol_rate="27500" txp="eutelsat-local-test.2401"
web_site="www.reuters.com"/>
  </master_channel>
  <publisher_channel>
    <publisher_header encryption_flag="0" id_publisher="22" packages_amount="0"
update_flag="0"/>
  </publisher_channel>
  <publisher_channel>
    <publisher_header encryption_flag="0" id_publisher="20" packages_amount="0"
update_flag="0"/>
  </publisher_channel>
  <publisher_channel>
    <publisher_header encryption_flag="0" id_publisher="7" packages_amount="1"
update_flag="0"/>
  </publisher_channel>
  <package autolisten_flag="0" bandwidth="128" checksum="3821867146" description="Il
sito web di Roberto D'Agostino" fec_algorithm="XOR" files_amount="3" id="71" id_cas_service="1"
id_cas_settings="1" id_drm_service="0" input_group="100" jobs_amount="1"
launch_file="212.239.51.72/public_html/index.html" logic_version="159" name="Dagospia"
needed_version="0" output_group="110" port="11021" public_flag="1" size="146769462" target="all"
type="web" updatable_flag="1" version="160" view_target_group="all">
    <package_info value1="comp"/>
    <job bandwidth="128" download_target_group="all" further_jobs_amount="0"
id="-1" start_date="1049743923000"/>
  </package>
</publisher_channel>
```



```

<publisher_channel>
  <publisher_header encryption_flag="0" id_publisher="25" packages_amount="4"
update_flag="0"/>
  <package autolisten_flag="0" bandwidth="512" checksum="2487187917" description="Les
dernieres infos fournies par Reuters via OPENSKEY" fec_algorithm="XOR" files_amount="1" id="381"
id_cas_service="0" id_cas_settings="0" id_drm_service="0" input_group="100" jobs_amount="1"
launch_file="eutelnv_FR_index.htm" logic_version="13916" name="ReutersNews_FR"
needed_version="0" output_group="104" port="11006" public_flag="1" size="16597728" target="all"
type="web" updatable_flag="1" version="7694" view_target_group="all">
    <package_info value1="comp"/>
    <job bandwidth="512" download_target_group="all" further_jobs_amount="0"
id="-1" start_date="1049711417000"/>
  </package>
  <package autolisten_flag="0" bandwidth="512" checksum="3877395915" description="The
Latest News with Reuters via OPENSKEY" fec_algorithm="XOR" files_amount="1" id="383"
id_cas_service="0" id_cas_settings="0" id_drm_service="0" input_group="100" jobs_amount="1"
launch_file="eutelnv_GB_index.htm" logic_version="12384" name="ReutersNews_GB"
needed_version="12383" output_group="104" port="11008" public_flag="1" size="133872" target="all"
type="web" updatable_flag="1" version="7427" view_target_group="all">
    <package_info value1="comp"/>
    <job bandwidth="512" download_target_group="all" further_jobs_amount="0"
id="-1" start_date="1049711700000"/>
  </package>
  <package autolisten_flag="0" bandwidth="512" checksum="526908969"
description="Ultime Notizie con Reuters via OPENSKEY" fec_algorithm="XOR" files_amount="1" id="382"
id_cas_service="0" id_cas_settings="0" id_drm_service="0" input_group="100" jobs_amount="1"
launch_file="eutelnv_IT_index.htm" logic_version="13459" name="ReutersNews_IT" needed_version="0"
output_group="104" port="11007" public_flag="1" size="12737606" target="all" type="web"
updatable_flag="1" version="7139" view_target_group="all">
    <package_info value1="comp"/>
    <job bandwidth="512" download_target_group="all" further_jobs_amount="0"
id="392446" start_date="1049711722000"/>
  </package>
</publisher_channel>
</guide>

```

5.2 Receiving OPENSKEY Packages on the Client Side (EUTELSAT)

Module Profile		
<name OPENSKEY Client>		
Executable or Library(Support)	JDK 1.3..1_07 (Sun)	
Single Thread or Multithread	Multithread	
Language of Development	Java,	
Responsible Name	Alexandre BRUNELLE	
Responsible Partner	EUTELSAT	
Status (proposed/approved)	Approved	
Platforms supported	Windows, Linux, Mac Intosh	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
		Proprietary: protected

File Formats Used	Shared with	File format name or reference to a section
XML format		
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Java Bases	SWING, AWT	
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not
Java Native Interface		

The OPENSKEY client application gathers three services for multimedia applications via satellite: Internet via satellite with asymmetric connection (satellite / modem), multicast video streaming (MPEG4) and multicast content download.

The OPENSKEY client software itself is made up of two parts: one mandatory part contains very few graphical elements, called OSListener that should always be running, another part which is solely intended to provide the end user with a graphical user interface.

The multicast content download called SatKiosk for the B2C market and Push for the B2B market, makes it possible downloading any kind of content offline via satellite.

Multicast download is similar to TV broadcast in that common content is being broadcast to a set of clients at the same time. The same way a user decides to watch or not a specific program on TV, a user will decide to receive or not a specific content on his/her computer.

The SatKiosk/Push application for multicast content download is the part of the OPENSKEY client application we are interested in for the AXMEDIS project.

Following is a description of what are the possible interactions with the content within the OPENSKEY SatKiosk application.

OPENSKEY content handling

As a user consults the TV program guide to know what is coming next on TV and what time, a user will consult the SatKiosk content program guide to know what is coming next on SatKiosk and what time.

As of today, the content program guide is made of a single text file an application can parse to extract critical content information. For a full description of the content program guide, refer to the Push Protocol [PP] document.

From this content program guide, a user or application can execute different actions:

- specify content download folder
- select content to download (download once or keep content updated)
- interrupt content download
- view downloaded content (open folder or default file)
- unzip compressed content
- repair corrupted content
- delete content

Following is a description of how to execute these actions with the graphical user interface.

5.2.1 OPENSKY B2C Application for the Content Consumer (GUI)

The OPENSKY SatKiosk application offers a set of graphical user interface for OPENSKY content handling:

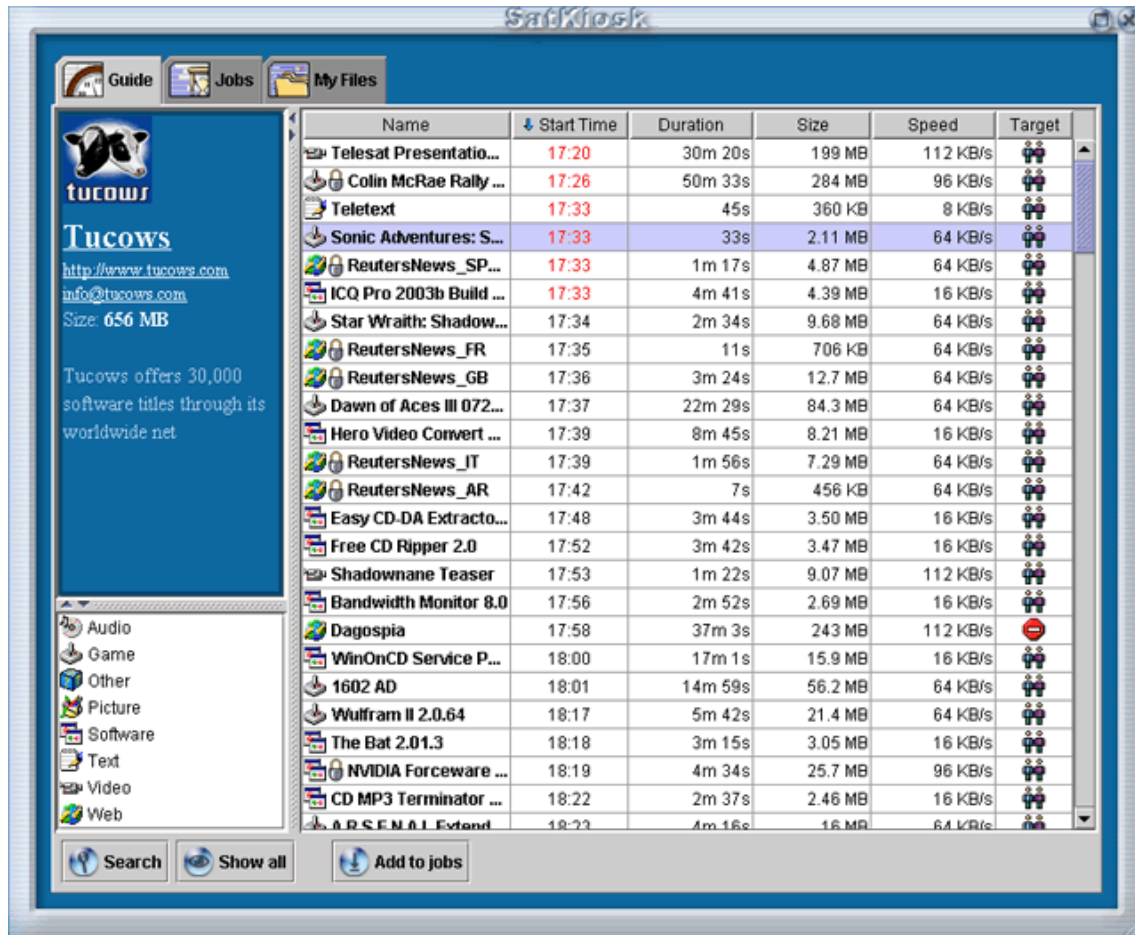


SatKiosk settings dialog

The settings dialog allows user to specify SatKiosk settings and other options. User can define SatKiosk content download folder on the disk.

The SatKiosk content program guide displays the list of available content, download start time, download duration, content size on disk, download speed...

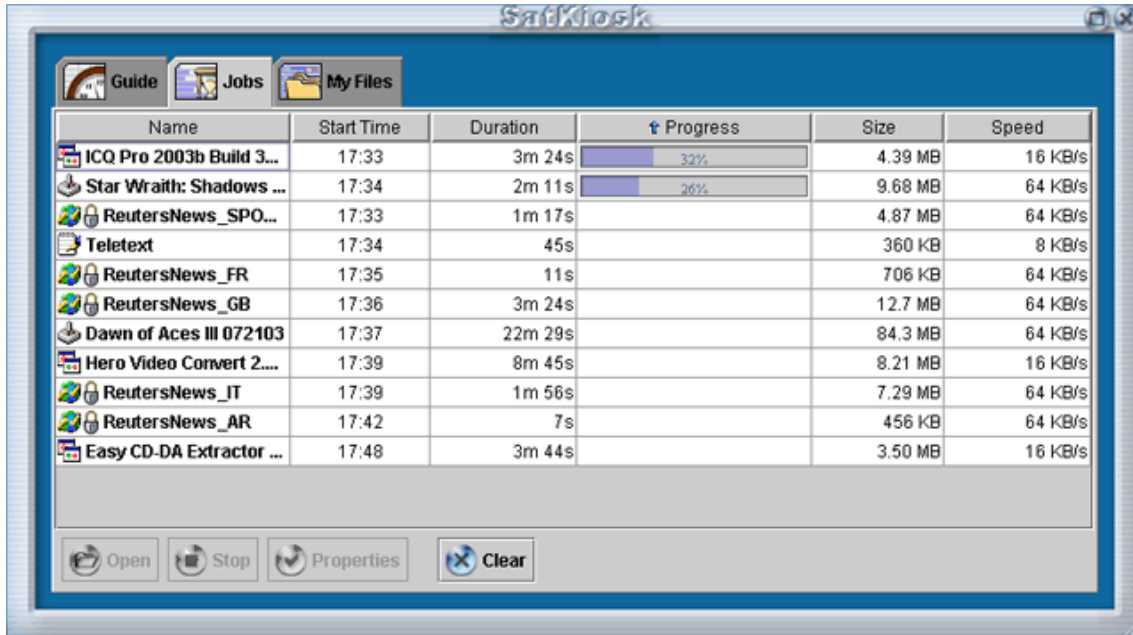
User can filter out content upon type (audio, game, video, software...) and provider.



SatKiosk content program guide

From this section, user selects content to download and gets to the following section.

The SatKiosk jobs section displays the list of currently downloading content. User sees content name, download start time, download completion in percent, content size and download speed. From this section, user can interrupt a content download or browse the folder of the content being downloaded.



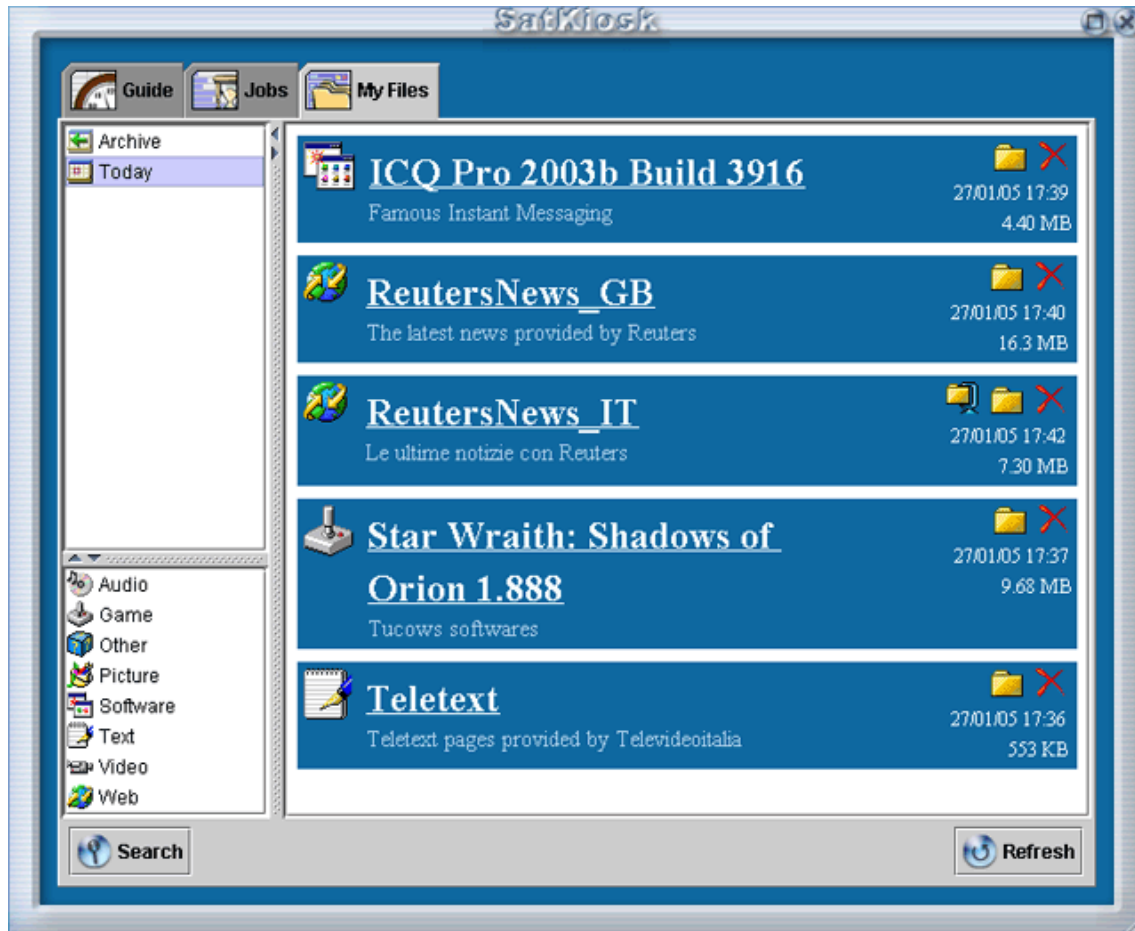
Name	Start Time	Duration	Progress	Size	Speed
ICQ Pro 2003b Build 3...	17:33	3m 24s	32%	4.39 MB	16 KB/s
Star Wraith: Shadows ...	17:34	2m 11s	26%	9.68 MB	64 KB/s
ReutersNews_SPO...	17:33	1m 17s		4.87 MB	64 KB/s
Teletext	17:34	45s		360 KB	8 KB/s
ReutersNews_FR	17:35	11s		706 KB	64 KB/s
ReutersNews_GB	17:36	3m 24s		12.7 MB	64 KB/s
Dawn of Aces III 072103	17:37	22m 29s		84.3 MB	64 KB/s
Hero Video Convert 2...	17:39	8m 45s		8.21 MB	16 KB/s
ReutersNews_IT	17:39	1m 56s		7.29 MB	64 KB/s
ReutersNews_AR	17:42	7s		456 KB	64 KB/s
Easy CD-DA Extractor ...	17:48	3m 44s		3.50 MB	16 KB/s

SatKiosk jobs

When content download is complete, user can move to the last section of the SatKiosk application.

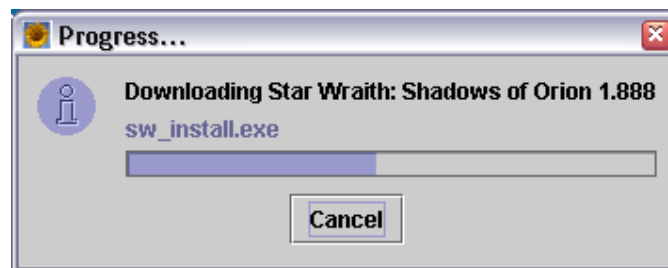
The My Files section of SatKiosk displays the list of downloaded content: content name, description, type, download date and time and content size.

As in the content program guide section, user can filter out content upon type (audio, game, video, software...) and download date (archive, last week, Monday, Tuesday... today).



SatKiosk My Files section

In this section, user can browse or delete downloaded content. User can also unzip compressed content or repair corrupted content via modem as shown on the following screenshot:



Downloading part of a corrupted content via modem

All user interactions with SatKiosk content through the graphical user interface are possible from an application through the application programming interface, as described in the following section.

5.2.2 OPENSKY B2B Application for the Content Provider (API)

The OPENSKY SatKiosk application offers a set of application programming interface for OPENSKY content handling. This API makes it possible to control SatKiosk configuration and content reception without the need of the GUI, only OSListener needs to be started.

OPENSKY settings

Like the settings dialog counterpart of the GUI, the settings files enable any application to read and change OPENSKY settings. There is a set of settings files that can be edited to modify the behaviour of the OPENSKY client application.

Refer to the OPENSKY Client Settings [OCS] document for a full description of the OPENSKY client application main settings file (opensky.ini).

Refer to the OPENSKY Content Configs document [OCC] for a full description of the OPENSKY client application content configuration files (general.cfg, oscontrol.cfg, services.cfg, transponders.txt, streamdb.cfg...).

An application can modify any of the previous settings files to obtain the desired result and behaviour for the OPENSKY client application.

SatKiosk content handling

Like the SatKiosk dialog counterpart of the GUI, there is an API based on files on the disk that help third party applications handling content.

Refer to the Push Protocol document [PP] for a full description of SatKiosk content handling with files. Basically, there are five files an application can read and edit in order to query content information and execute content actions:

- guide.txt is the content program guide OSListener receives via satellite providing a list of available content, download start time, download duration, content size on disk, download speed... Its counterpart on the GUI is the content program guide section (which is actually built up on this very file). An application can use this guide to know the available content, when the download is starting and what is the private group and encryption applied to the content.
- jobs.txt is used to trigger content download actions like “start listening once”, “start listening and keep updated” and “interrupt download”. It also gives information on currently listening content, as it is updated along with content download progress.
- pk_info.os is the meta-data file that keeps content information on the disk when download is complete. It gathers content name, description, size, type...
- pk_lost.os provides the application with the amount of packets lost during transmission. The size of a packet being currently 1409 bytes. The packet loss information allows the application to start content repair (lost packets download via modem), or decide to download the content via satellite once more.
- fl_lost.os provides the application with the amount of files lost during transmission. The file loss information allows the application to start content repair (lost files download via modem), or decide to download the content via satellite once more.

Note that the opensky.ini file described in the previous section, together with the content name and provider name (found in the guide.txt) define the content location on the disk. For instance, if content MovieTrailers

AXMEDIS Project

from provider Warnerbros is downloaded on the disk, it can be found in the following folder: contentFolder\Warnerbros\MovieTrailers where “contentFolder” is the root folder for SatKiosk content, as defined in the opensky.ini file.

Another important point for content handling is the content target group. The target group defines the group of users allowed to download a specific content. This property is defined for content but handled on the server side. Indeed it is the content publisher who decides which user is assigned to a group, hence which user can download a specific content. Provided that the client configuration is properly setup for group reception, the content filtering will occur automatically on the client side without the application interaction. Once downloaded, target groups can be found in the Eutelsat\Groups folder within SatKiosk content folder.

5.2.3 OPENSKY towards AXMEDIS

API

The whole API based on file reading and writing makes it possible to communicate with OSListener from a third party application in order to setup and handle content reception.

The OPENSKY development planning focuses on implementing a common API where interactions can occur through a single entry point. Applications will be able to get/set configuration, start/stop SatKiosk content and use the content on the disk through one single object. The new API could sum up as the call to one single method like:

```
CommandResult = API.execute( CommandName, CommandProperties )
```

Where CommandName could be a value from the following list:

- GetSettings
- SetSettings
- GetContentList
- GetContentDetails
- StartContentDownload
- StopContentDownload
- GetContentLocation
- ...

Let us see in detail the usage of the GetContentDetails command:

```
import eutelsat.ei.api.APIMaster;
import eutelsat.ei.api.CommandProperties;
import eutelsat.ei.api.CommandResult;

...
CommandProperties myContentDetails = new CommandProperties();
myContentDetails.setProperty( "name", "AXMEDIS content" );

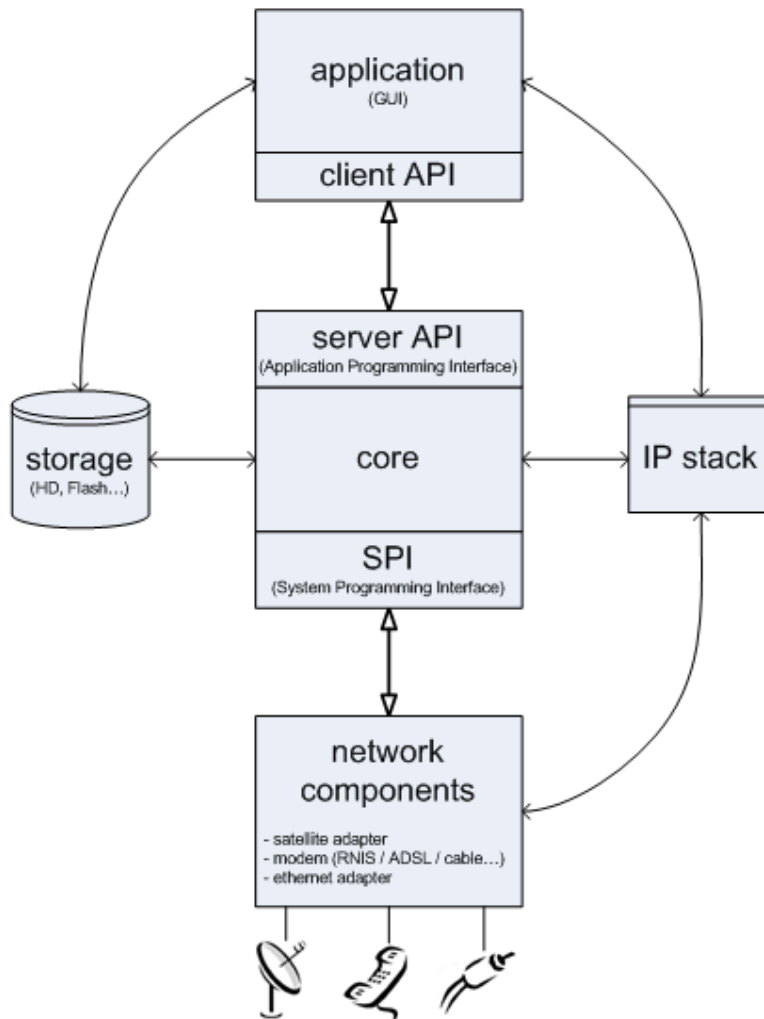
CommandResult myResult = null;
myResult = APIMaster.instance().execute( "GetContentDetails",
                                         myContentDetails );

if( myResult.isSuccessful() ) {
    // use received details
    myContentDetails.getProperty( "id" );
    myContentDetails.getProperty( "start_time" );
    myContentDetails.getProperty( "size" );
    ...
}
```


Design

The evolution of the API of OPENSKEY is only one part of the overall evolution of the OPENSKEY client application. The whole design of the application is due to evolve and look as shown on the diagram below:

client software overview

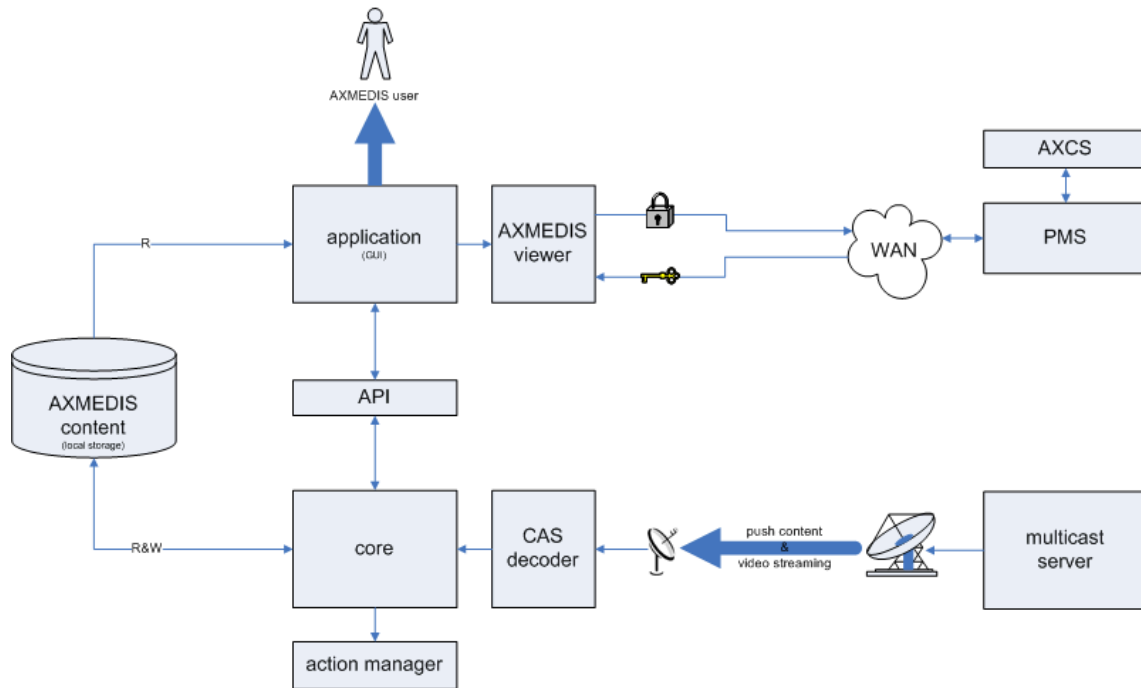


The core is to replace OSListener with a new design as well as the System Programming Interface (SPI) is due to replace and gather OSControl and OSJni, respectively responsible for DVB adapter and modem control. As shown on the diagram the application will have to include the API in order to communicate with the core.

AXMEDIS client

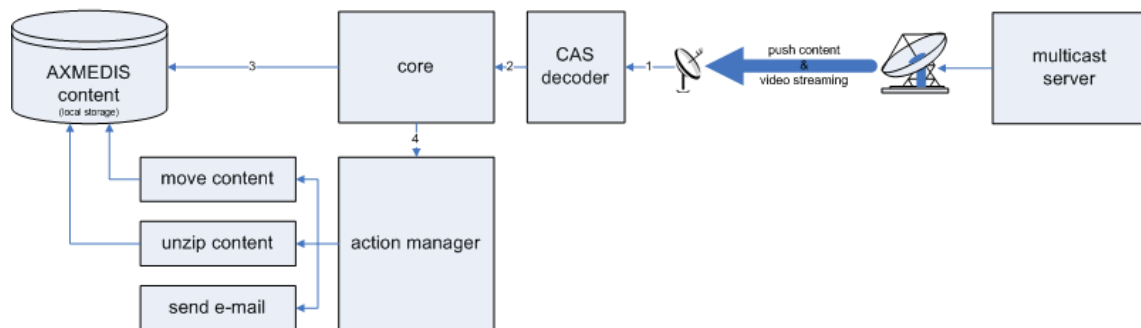
General diagram

Here is a general diagram of the AXMEDIS client: OPENSKEY for the AXMEDIS content consumer / viewer.



AXMEDIS content download scenario

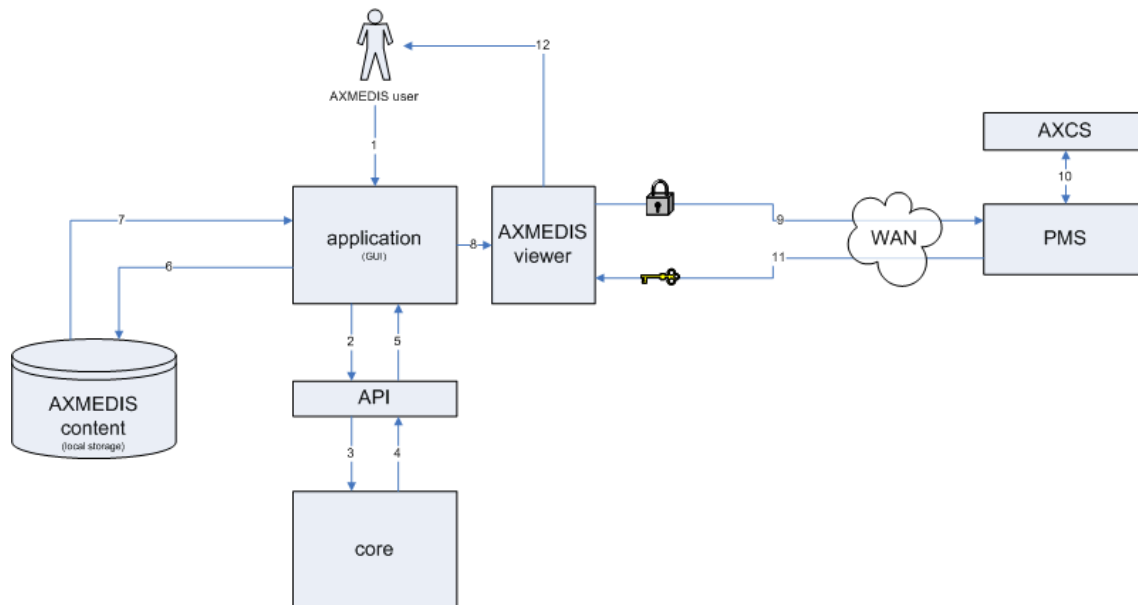
Let us see the overall application in action with the scenario of AXMEDIS content downloaded on the disk via satellite:



1. the content is received via satellite.
2. the content is decrypted with the CAS decoder (if needed).
3. once decrypted, the core applies “on the fly” actions like checksum calculation and stores the content on the disk.
4. once completely downloaded, the core notifies the action manager to take actions like moving the content to another location, unzipping the content, sending an e-mail or any other possible default or custom actions...

AXMEDIS user scenario

And now let us see the scenario of a user accessing AXMEDIS content:



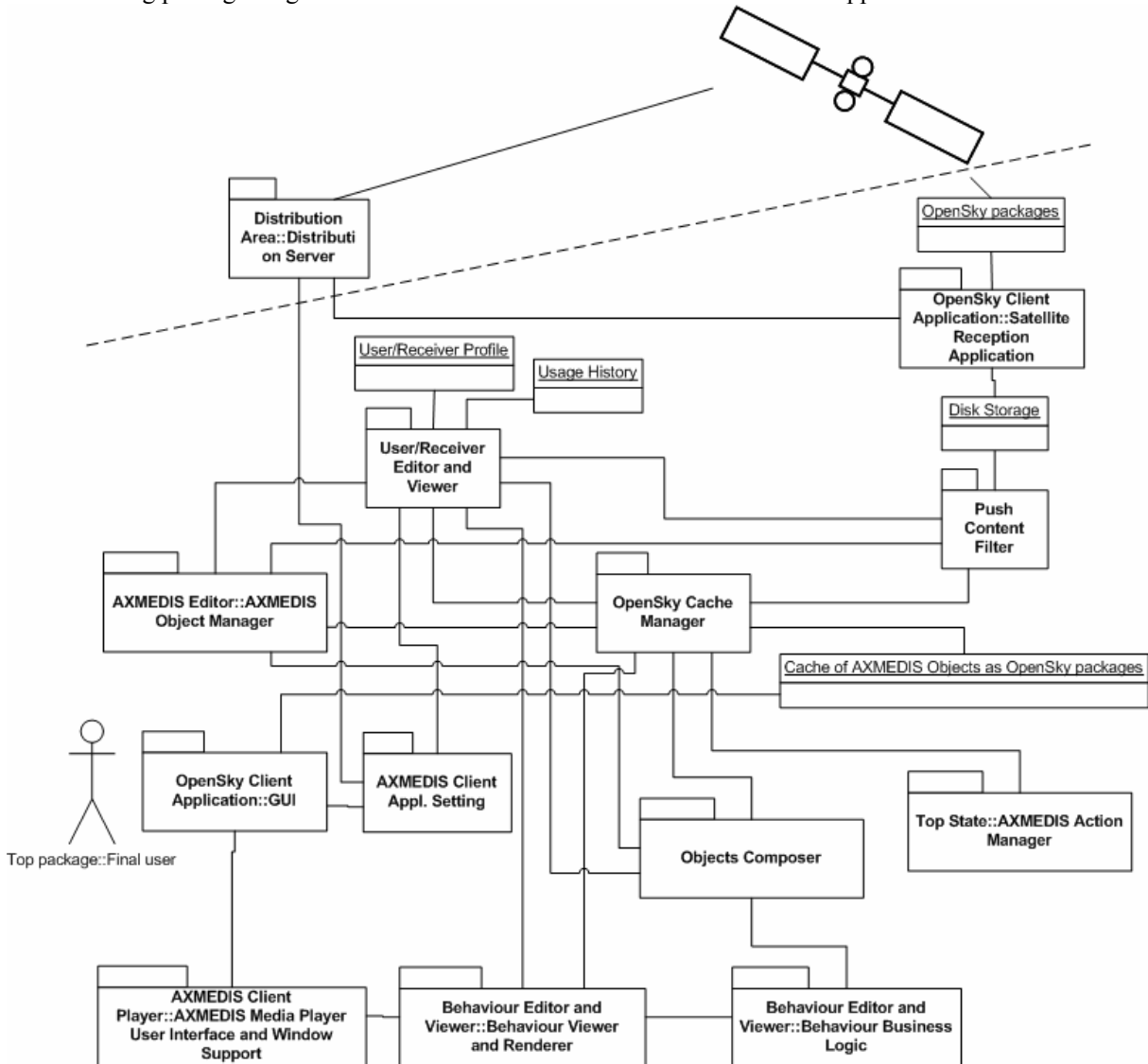
1. the AXMEDIS user selects (clicks) a specific content on the application user interface.
2. the application requests the content details to the core using the API, as described in the API section (i.e. `APIMaster.instance().execute("GetContentDetails", myContentDetails)`).
3. the API request is received by the core, which executes the corresponding function.
4. the core returns the requested information (content details).
5. from the application point of view, the API request is seen as one function call providing a result.
6. the result of the API request is used to locate content on the disk.
7. the application loads the content from the disk.
8. the content is passed to the AXMEDIS viewer.
9. the AXMEDIS viewer uses the content information obtained on step 5 to conclude that the content is protected and requires decryption with a key before viewing. The request for the specific key is sent to the Protection Manager Server (PMS).
10. the PMS gets certification from the AXMEDIS Certified Supervisor (AXCS).
11. the PMS returns the requested key to the AXMEDIS viewer. Le later uses the key to decrypt the selected content.
12. the AXMEDIS user can view the selected content with the AXMEDIS viewer.

5.3 Filtering of AXMEDIS Objects on the Client Side (CRS4)

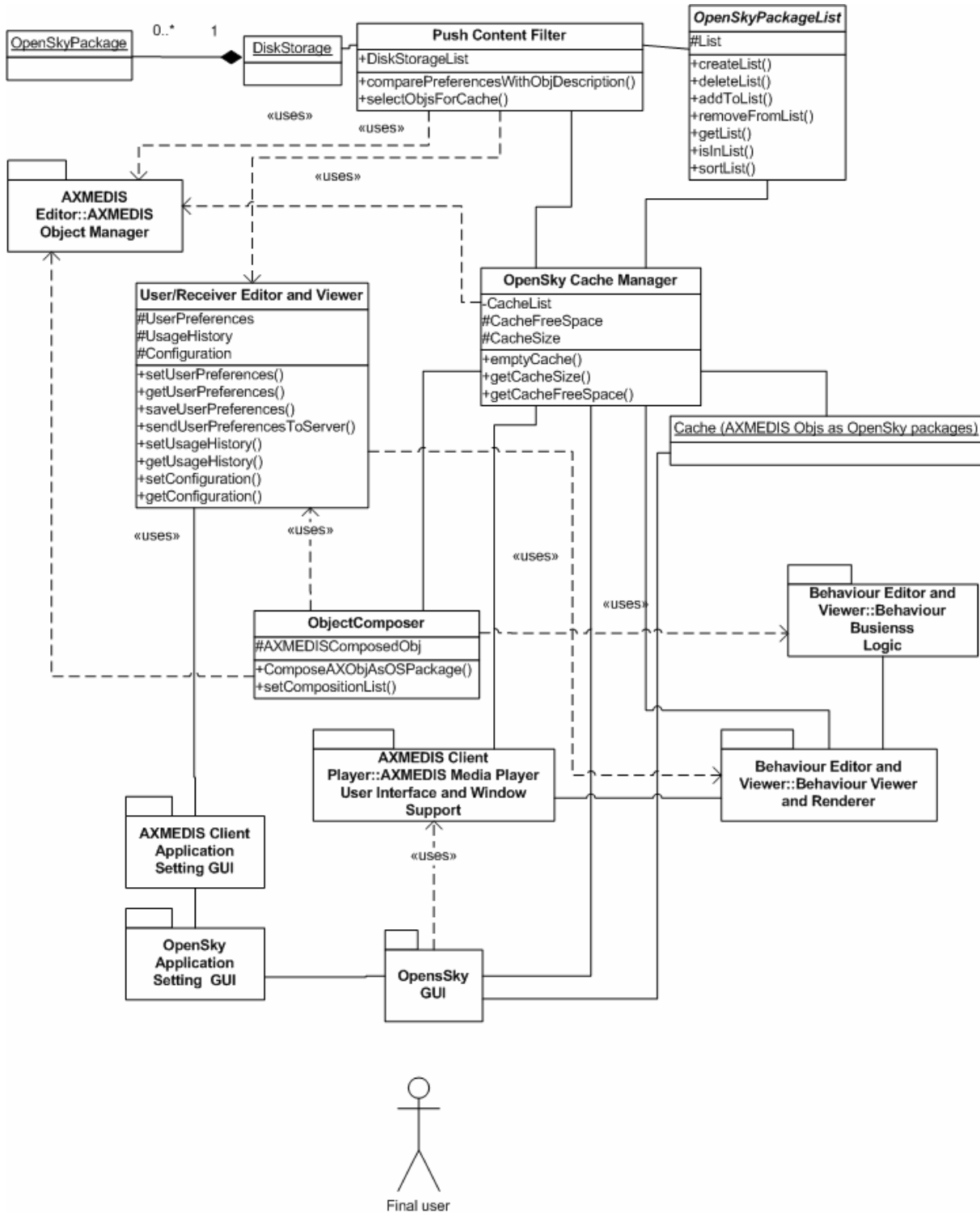
The Satellite Reception Application (which is a package of OpenSky Client Application) receives as input all the OpenSky packages which carry AXMEDIS Objects and puts them in the Disk Storage. The Push Content Filter of the AXMEDIS Client Application allows only those AXMEDIS Objects which are compliant to the User Profile to be stored in the Cache: it reads the User Profile (through the User Editor and Viewer), reads the content of the Cache (through the OpenSky Cache Manager) and the informatio parts of the AXMEDIS Objects (by means of the AXOM), and comparing these with the space available in the cache chooses what content is to be put in the Cache. The Cache is managed by the OpenSky Cache Manager (e.g. for reading its content, searching, add objects, delete objects, empty all the content, etc.). It contains AXMEDIS Objects

inside an OpenSky tree (each AXMEDIS Object is inside a directory along with an OpenSky information file, called `pk_info.os`).

The following package diagram shows the overall functionalities of the Client Application.



The following class diagram shows a deeper behaviour of the main functionalities of Push Content Filter, OpenSky Cache Manager, and Object Composer.



5.3.1 The Virtual Channels composition[CRS4]

The Object Composer gets the list of AXMEDIS Objects present in the Cache (by means of the OpenSky Cache Manager), reads their information part (using the AXOM functionalities), and compare these metadata with the User Profile (through the User/Receiver Editor and Viewer) to compose new AXMEDIS Objects made of references to the AXMEDIS Objects in the Cache.

The SMIL file with the rules needed to manage the temporal and/or spatial sequel of the AXMEDIS Objects which compound the Virtual Channel is made by means of the Behaviour Business Logic (of the Behaviour Editor and Viewer sub-system).

This newly created AXMEDIS Object is stored as OpenSky package in the Cache, together with all the other AXMEDIS Objects.

5.3.2 The Player [CRS4]

The GUI of the OpenSky Client Application is able to show the list of AXMEDIS Objects in the Cache. When the Final User selects an AXMEDIS Object from the list, the AXMEDIS Client Player is activated. This Player actually activates the SMIL player described in the Part B of this document. The SMIL Player is also able to read the behaviour SMIL file of rules to play the Virtual Channels.

5.3.3 User Profiling in the client side [CRS4]

The User Editor and Viewer manages the functionalities related to the User Profile and its updating, accordingly to the Usage History. The User Profile has a static part (such as, name, age, profession, etc.) and a variable part about the preferences (such as, kind of programs, genre, actors, etc.). Both the User Profile and the Usage History are XML files which are based on the metadata set of the ETSI standard TV-Anytime [TVA] (which are the same defined by MPEG-7 standard).

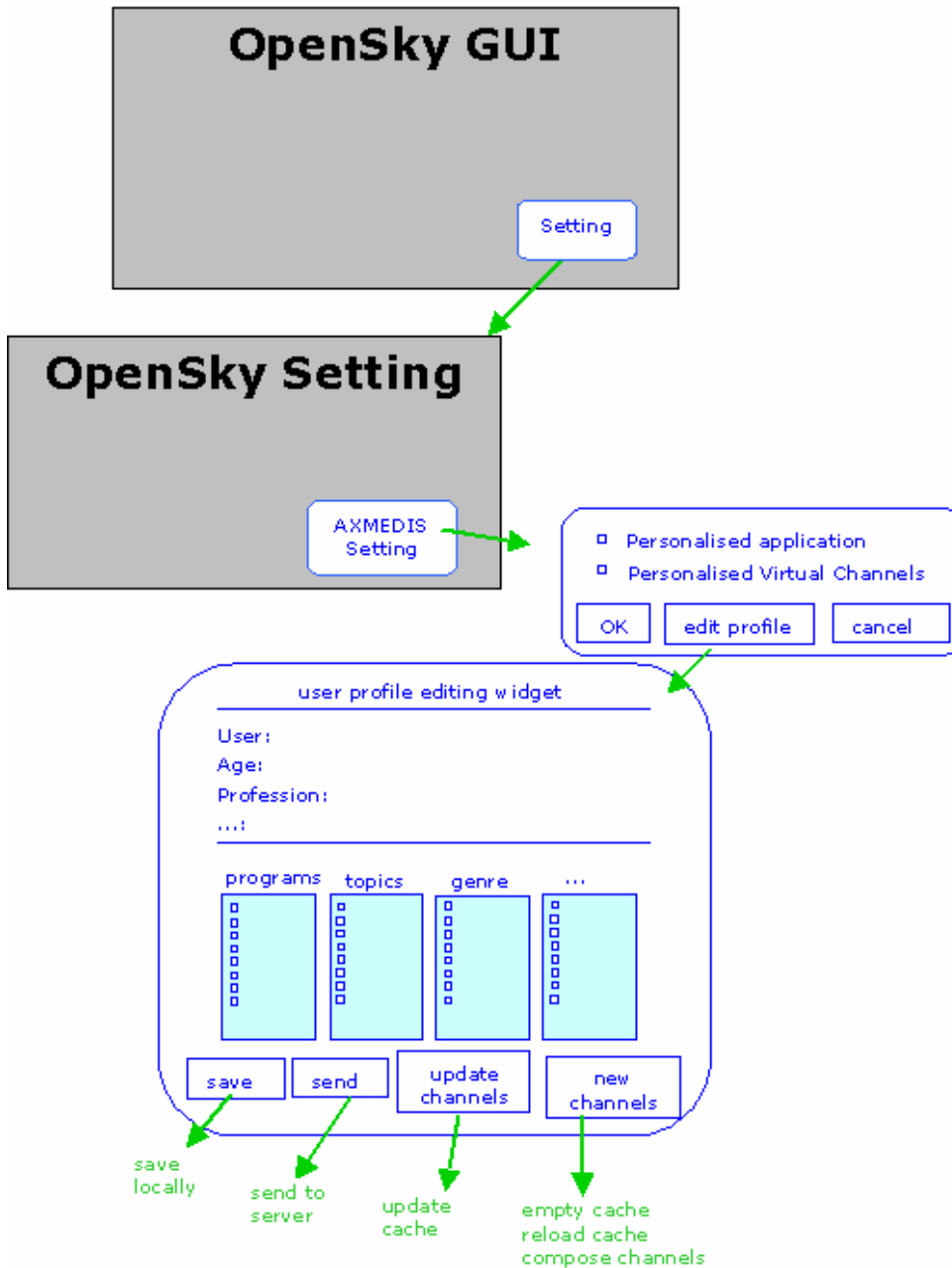
The Usage History file has information about the choices made by the user through the Player. This file can be used to update the User Profile file. Any changes in the User Profile shall be validated by the user.

The User is allowed to edit directly his/her profile by means of a GUI which is activated by the OpenSky Client Application GUI. In this way the user can change his/her preferences, store them locally or send them to the Server.

5.3.4 Graphic User Interface for the Application setting[CRS4]

The Final User manages the Application setting through a window of the OpenSky Client Application GUI. In this way, the User is able to activate the Push Content Filter in order to have only the AXMEDIS Objects compliant to his/her preferences. He/she can also select the Application which automatically builds the personalised virtual channels made of a list of AXMEDIS Objects present in the Cache.

The User is not able to directly manage the AXMEDIS Objects in the Cache, but he/she can choose to have the Cache emptied and refilled, and to have new personalised virtual channels.



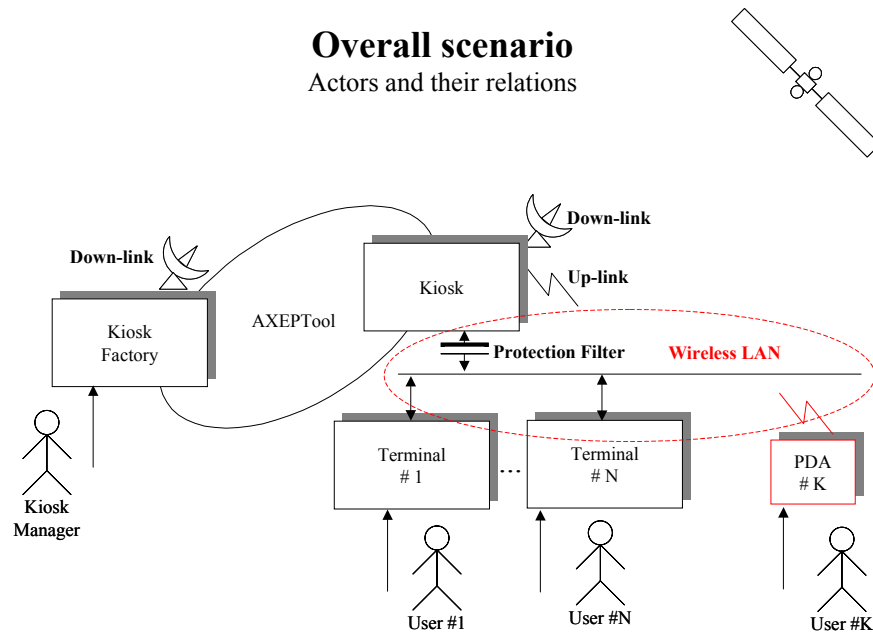
5.4 References

Reference	Document Name	File Name	Availability
[APS]	API Push Server – Public Functions	PushServerAPI_functions_Partners.pdf	public Attached to this document
[OCS]	OPENSKEY Client Settings	OPENSKEY Client Settings.pdf	Available upon request

[OCC]	OPENSKY Content Configs	OPENSKY Content Configs.pdf	Available upon specific request
[PP]	Push Protocol	Push Protocol.pdf	Attached to this document
[OWP]	OPENSKY Push Web Interface Publisher Guide	OPENSKY Push_Web_Interface_Publisher_Guide.pdf	Attached to this document
[TVA]	TV-Anytime phase 1, part 3 metadata, sub-part 1: Metadata schemas. ETSI TS 102 822-3-1	TV-Anytime-Metadata-schemas.pdf	Attached to this document

6 Distribution towards PDA via Kiosks (WP9.6: ILABS, DSI, EXITECH)

What follow is the description of the architecture of the kiosk distributor specified in terms of blocks, classes, interfaces, data structures, methods and relations; possible GUI are presented too. In the following picture is represented the overall scenario, involved actors and relations.



Actors

In this scenario the foreseen actors are the following:

- ❑ **Kiosk Manager:** interacts with the Kiosk Factory, which is connected to the P2P infrastructure of AXEPTool and prepares, publishes & distributes content for the Kiosk instances of AXMEDIS.
- ❑ **User #x:** a generic user that can interact with the Kiosk either directly or via PDA. The user will need to register and certify own device prior to be granted the possibility to use own PDA. If PDAs will be provided locally by the Kiosk Manager they will already be equipped and configure to properly interact with kiosks and therefore the user will be able to use them as if he was using the kiosk

System components

At a very high level the system components are summed up as follows. Later one we will enter in much more details.

- ❑ **Kiosk factory:** this is the part of the system that holds a local instance of AXMEDIS and is interconnected to the P2P infrastructure. Here the Catalogue is produced and contents are aggregated, produced and distributed.
- ❑ **Kiosk:** this is the part of the system that is interconnected to the P2P infrastructure and has local terminals (including PDA). It assures management of satellite downstream and modem upstream (can also be used in downstream). Provides security feature to the local LAN. The kiosk architecture will enable wireless communication with local mobile devices (PDA...).

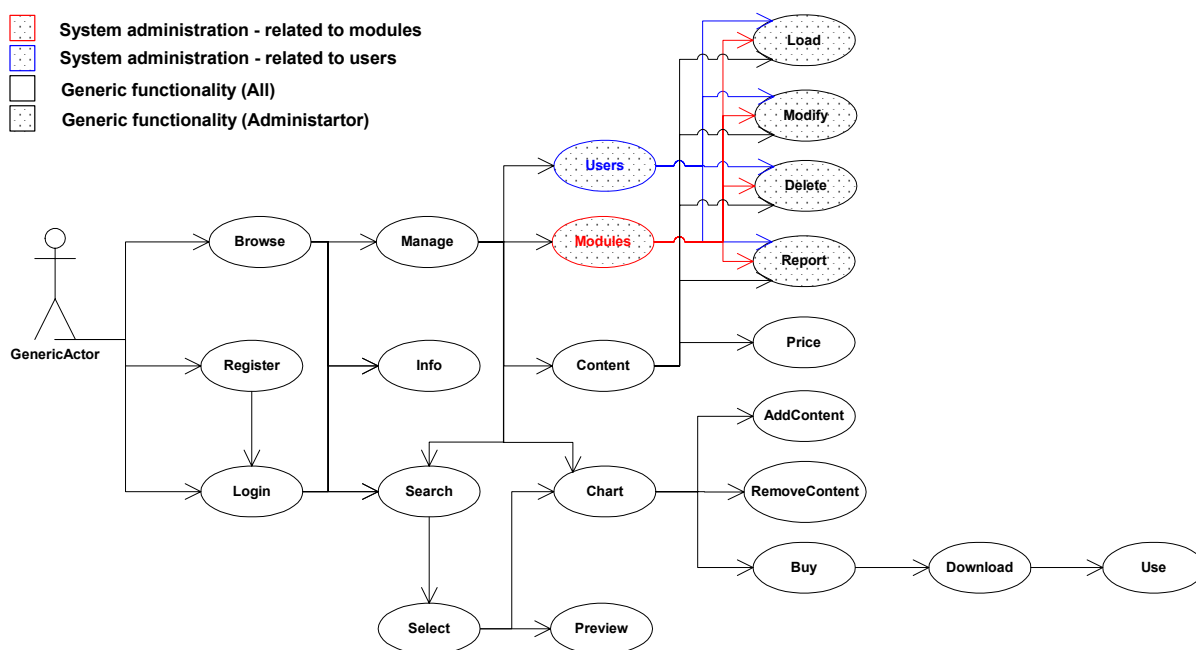
- ❑ **Terminal**: is a simple Point Of Presence (POP) for accessing to services provided by the Kiosk (mainly browsing and previewing of content listed into a catalogue for local fruition). With this device only rental is available.
- ❑ **PDA (local)**: this is a mobile version of the kiosk POP granting the user the kiosk functionalities. It is provided by the kiosk manager and requires cleaning after usage. With this device only rental is available.
- ❑ **PDA (user)**: is a user own device that will need to be identified and registered to the Kiosk in order to be used locally. Once registered the user will be provided the needed SW for using own PDA as a POP for the kiosk services.

It is worth noting that Kiosk's terminals and local PDA should be certified separately and autonomously from the user PDA. They do belong to the kiosk and not to the user, therefore even if the user logs-in is not meant to certify the device as this step should occur when the kiosk starts-up.

6.1 Overall use case

Starting from what just stated before and taking into account all the possible actions that will be performed at the kiosk is possible to define the following overall use case in the context of the kiosk. Summing up all into a single diagram may turn into a somehow difficult to read diagram, therefore to give a first glance impression of all operation for all actors in a single diagram a colour code has been adopted and will be detailed later on. Furthermore to simplify it we have also adopted a further simplification; foreseen actors are two: nominally the “*end-user*” and the “*administrative user*”. Actions have been divided too in terms of impacted objects; more specifically there are actions working on “*user*” objects (we refer here to data describing users and related to user management); actions working on “*modules*” (the system components) and generic actions working on every other object of the system.

In the following diagram blue has been used to identify actions related to “*user*” objects and red for those related to “*modules*”. Shaded nodes refer to functionalities devoted to administrative users while non-shaded ones apply to every actor.



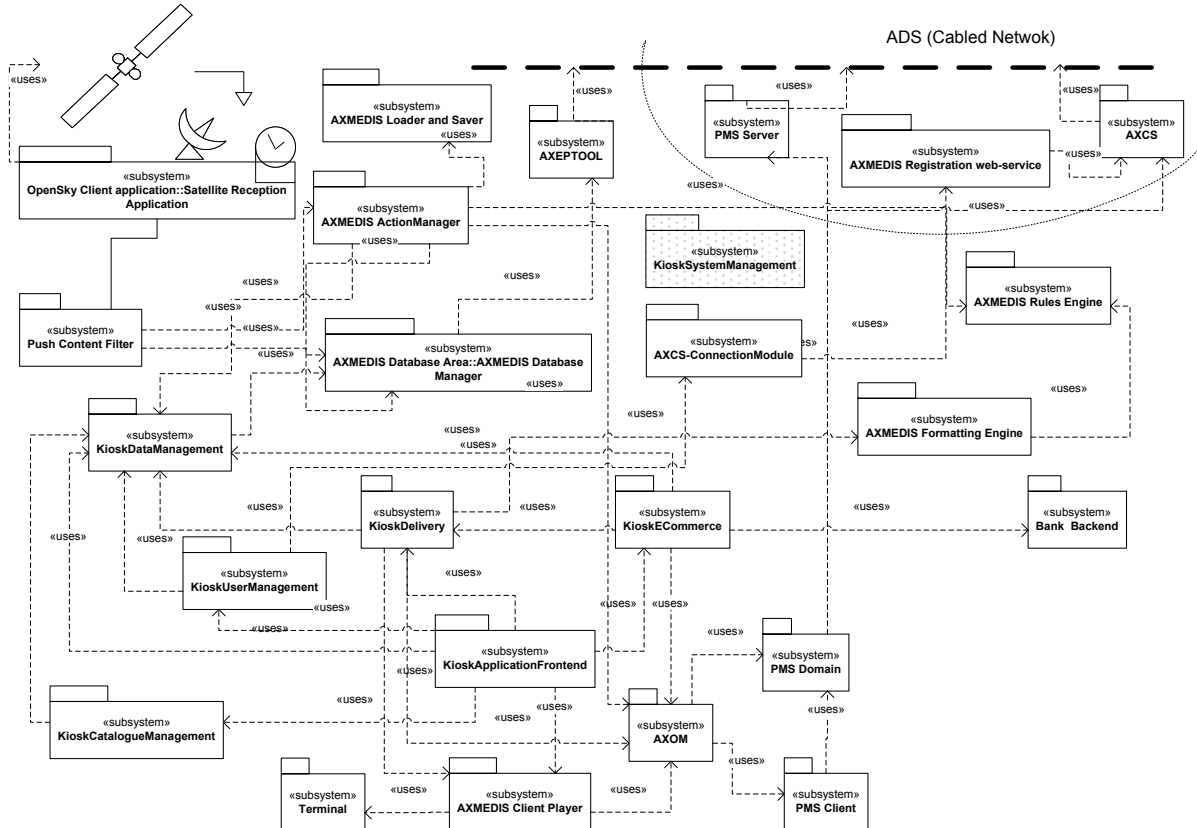
6.2 Overall architecture

In the following diagram all components of the kiosk system are presented in their essential role and with a clear indication of the usage relations among them. For the sake of clarity some modules that are external to the kiosk but extremely important for its correct functioning are also reported but beyond a dashed line (PMS server and AXCS). There is also a module that is shaded and not connected to others.

This is a rough simplification to avoid messing up the diagram with too many connections. Actually this “special” module is the one acting as system monitor (taking care to check overall functionality of the system through a constant monitoring of all its components).

There is also an other module specifically highlighted, the AXCS Connector module, as this is intended to solve a issue specific of the kiosk environment as far as connection with the AXCS is concerned in the user registration phase and in other very limited occasions.

The block indicated as “*terminal*” refers to the operating system of the generic terminal device, a PC, a tablet, a PDA or a smart-phone. The module tagged “*bank*” refers to the back system of the entity managing the actual payment.

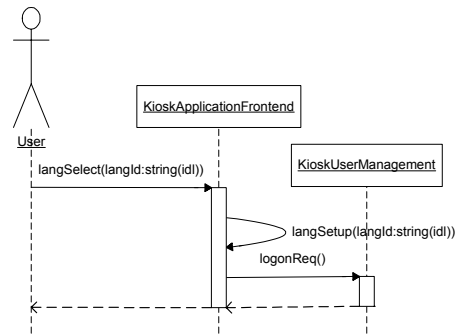


6.3 Temporal diagrams & related GUI aspects

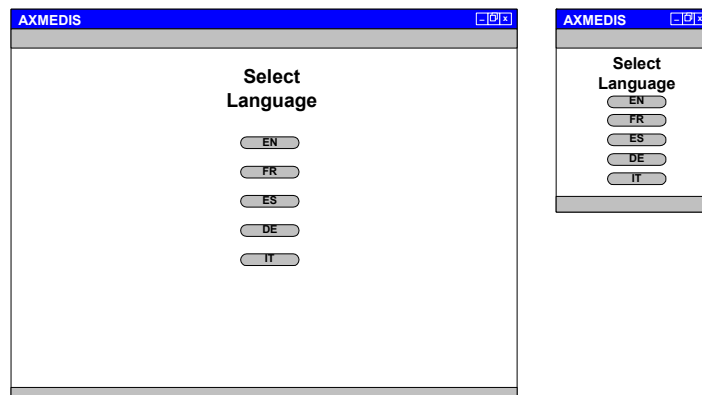
In the present section are reported the most relevant temporal diagrams describing interactions between the user and the system or among modules. The intended logic is to provide the most detailed info on the system structure and functioning. The followed approach starts by presenting user operations (like selecting the user interface, authenticating or registering, browsing, selecting, previewing, purchasing, downloading and using content. There are also other operation to be taken into account like profile update, content update and system management, all of them have been described in the use and test cases. Here will be definitely reported the related data structures and methods, but not all temporal diagrams in order to keep the dimension of the present section (and also of the document) to a reasonable size granting also an easy

reading. The present section is divided in subsections, each related to a specific operation or set of operations. As already mentioned such sub-sections will map the temporal diagrams presented. The starting point will be the GUI language selection followed by the user login / registration, to pass to content browsing and selecting and previewing. The next ones will deal mainly with content acquisition, purchase, download and fruition. The last ones will be referring to system maintenance, and in detail to content update. Here for content we intend whatsoever content from actual content that the user may use to application components that may be under upgrade.

6.3.1.1 User interface language selection



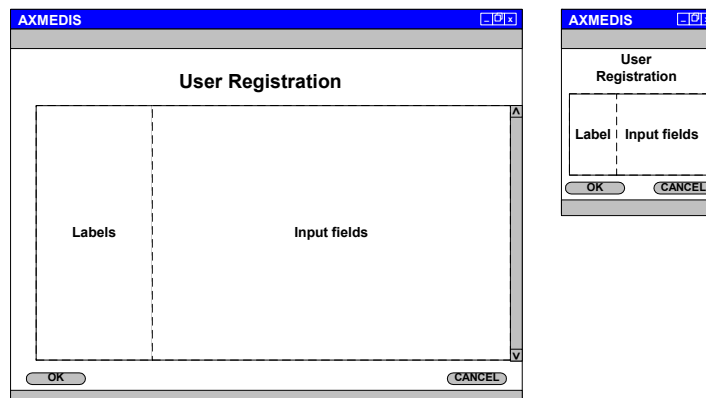
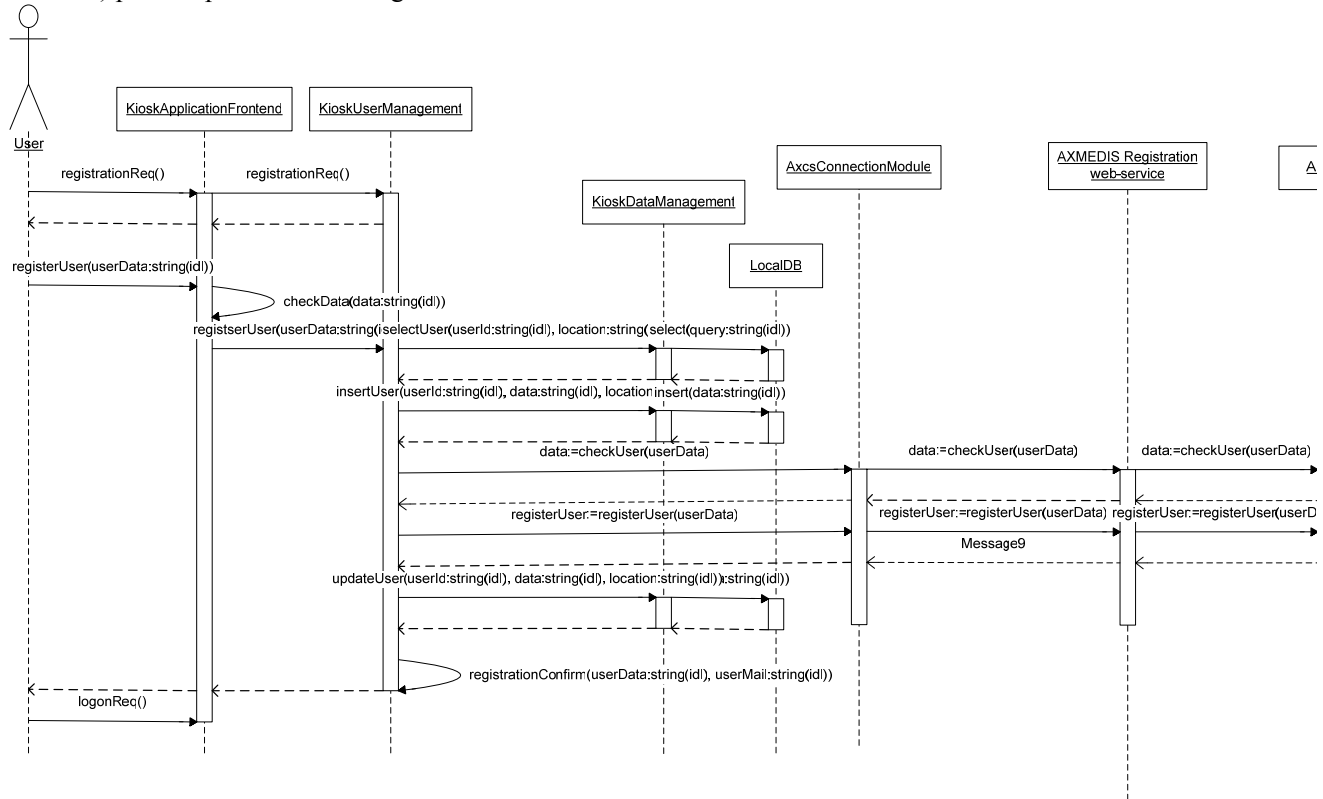
From this diagram is apparent that the user selects as the first thing the GUI language. The Application front-end intercepts such request and sets-up the appropriate environment variables to achieve expected results. Once this is achieved the application front-end requests to the user management to provide the required data structures to perform the logon. The user management module returns the requested data structure (see later on the section related to entity relation diagram and all related data structure descriptions). Once this operation is performed the application front-end presents the user the logon interface (see the following image) in the chosen language (for the demo is expected to provide only English and Italian).



6.3.1.2 User registration

The next operation to be examined in detail is the user registration. We choose to start from this rather than from the login as statistically is more probable that a user will need to register prior to be able to actually login the system. From the diagram is easy to notice that user will interact with the application front-end asking for initiating the registration procedure. This request will be passed to the user management module in order to retrieve the needed information structure. Once this is done the front-end will display the user the appropriate page to perform user registration data collection. The user will fill in the form (see the picture following the time diagram for a possible aspect of the registration form). Once this is done the user will send the data back to the front-end that will perform a consistency check prior to pass all to the user management for processing. The user management will store received data onto the local database via the data management module. Once this step will have been accomplished the user management will interact with the AXCS (via the AXCS connector Module) to check user data (in case the user is already an AXMEDIS user

despite being a new kiosk user) and retrieve the related data. If the user is not registered in AXMEDIS is given for granted that provided info will be sufficient to ensure proper user registration by the AXCS. Data received from the AXCS will then be stored locally prior to return control to the application front-end. The last operation that the user management will perform is to return the application front-end all AXMEDIS user related data for proper display to the user. The user will have to take note of part of the data (the sensible one) prior to proceed accessing the main kiosk GUI.



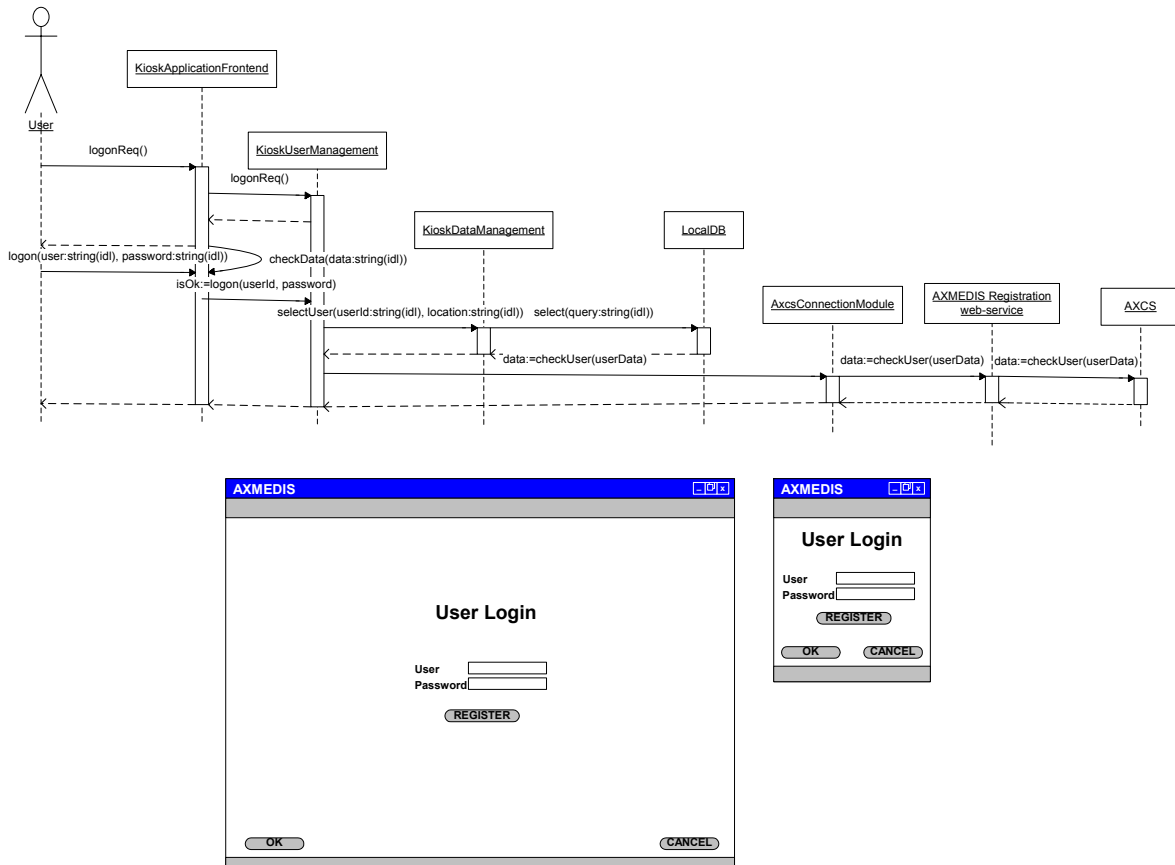
6.3.1.3 User login

The next operation to be examined in detail is the user login. From the diagram is easy to notice that user will interact with the application front-end inserting the login data and initiating the logon procedure. Data consistency will be performed and then the request is passed to the user management module. Once this is done the user management will look for user provided logon data in the local database via the data management module and then, in case of positive compare result of local authorisation (we always suppose a positive result in the reported diagrams) it will interact with the AXCS (via the AXCS connector Module) to

AXMEDIS Project

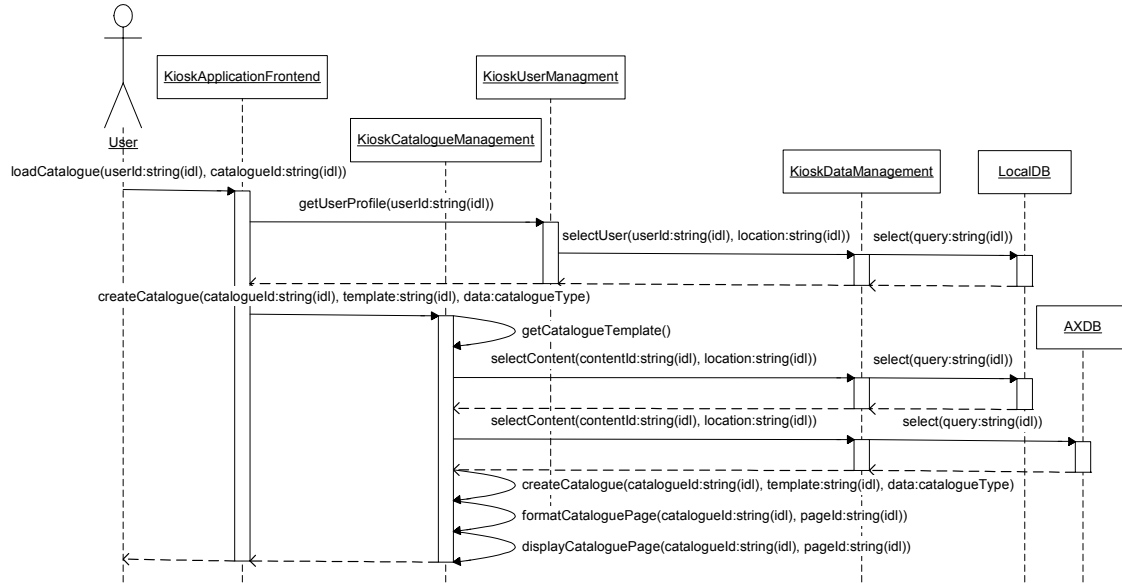
check user data (in case the user is already an AXMEDIS user) and log the operation. Once all this has been achieved the control will be returned to the application front-end that will grant the user access to kiosk resources and load the main kiosk GUI.

What follows is a possible instance of the user login page. Hereafter is reported a possible aspect of the related GUI.

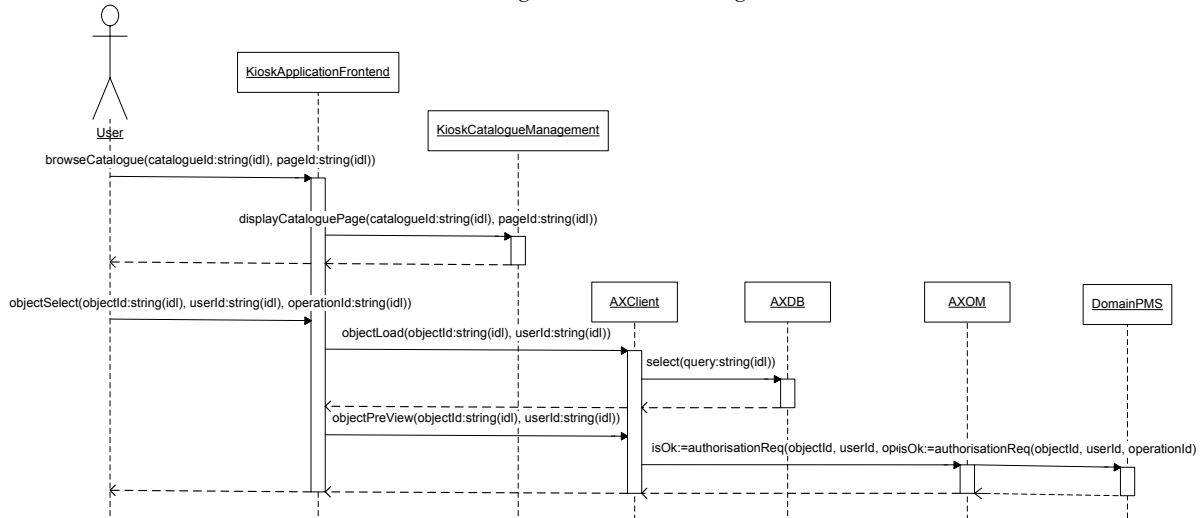


6.3.1.4 Content load, selection & preview

What follow is the set of operations that will occur after the user has logged into the system. For the sake of clarity the sequence diagram is split into several different phases each corresponding to one of the major steps described hereafter. Some of these steps may be repeated several times, like the content browsing, previewing, selecting, adding to chart and acquiring; this latter step can be performed for a single object or a collection. As a choice we will present just the case of a single object, as this will also apply to rental. So as already stated at this point the user interface will be loaded depending on the user profile that will be fetched from the local database. The catalogue management module will take care of preparing and loading the first page of the catalogue by combining available data, user profile and related catalogue template. In this phase info about AXMEDIS object will be retrieved. The user will be presented the current page of the catalogue and will be able to browse and select objects. Whenever a new page needs to be loaded a similar process will be followed as the one so far achieved to load the first page. When an AXMEDIS object is selected the application front-end will retrieve the full AXINFO so that if the user selects to preview object content or to render the related detailed information the AXMEDIS client will be able to properly perform required operations.

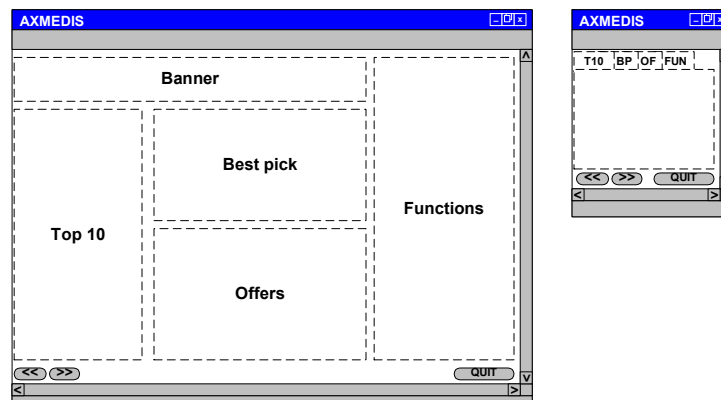


Catalogue creation and loading



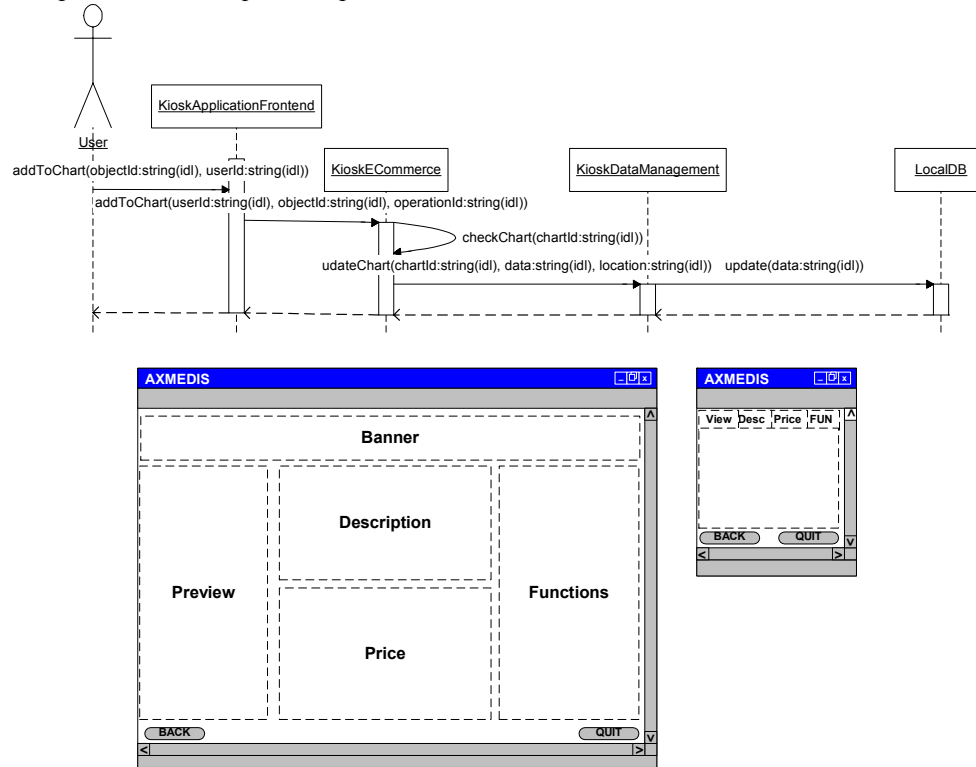
Catalogue browsing and content previewing

A possible aspect for the main browsing page is reported in the following image where functional areas are placed in evidence.



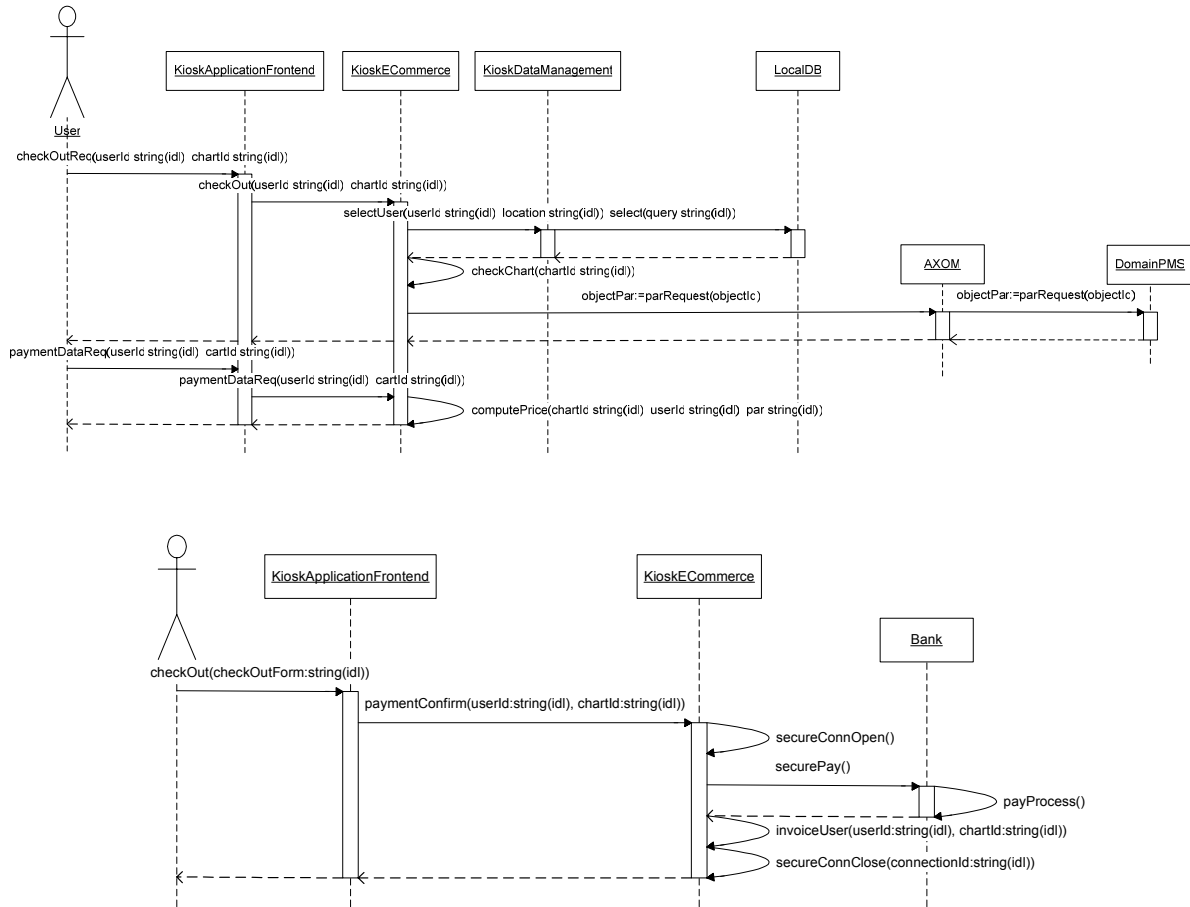
6.3.1.5 Content adding to chart

What follow is the set of operations that will occur once the user has decided to add a specific content to the chart. Basically the user will need first to select the object, just like we have already seen, then will need to specify that the selected object will need to be added to the chart. In this case we suppose that the chart may host one or more contents. In any case we will present only the case of a single object addition (just like later on, we will present only the case of a single object purchase). The chart addition operation is very simple in essence as the front-end module will simply need to inform the e-commerce module of the request passing the id of the selected object. This info in turn will be stored in the chart data structure into the local database by the data manager module, control will then be returned consequently by the data manager to the e-commerce module and by this one to the application front-end that will consequently inform the user of the successful completion of the requested operation.



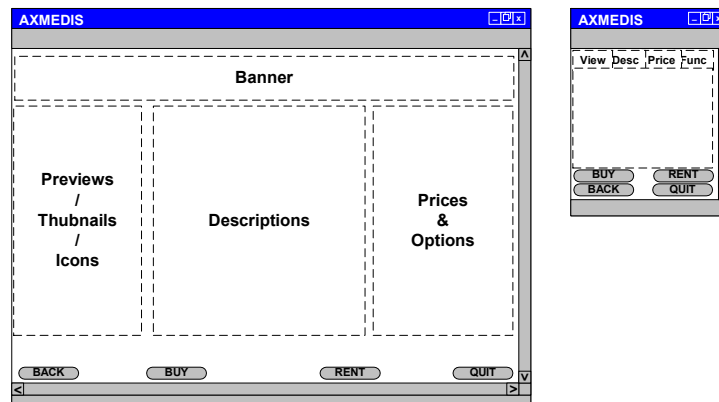
6.3.1.6 Content purchase

Once content has been selected, previewed and added to the chart, the user can decide to close the procedure either by quitting the system or proceeding to a checkout. In the latter case the following operations will have to be performed. The user will request to initiate the check out procedure. Therefore the application front-end will pass the request to the e-commerce module that will retrieve the current chart from the local database via the data manager. Once this is achieved (here we assume the chart is holding a single AXMEDIS object) the e-commerce module will request the Domain PMS to provide the potential available rights for the object whose ID is in the chart. Once retrieved the PAR the e-commerce module will verify what has been requested by the user and (eventually) request the formatting engine to properly format the desired object for delivery; then will determine the due amount and send the application front-end the potential billing information to be presented to the user for approval. If the user approves (and here we suppose so) he will fill in the needed payment information. Such data will be kept only for the transaction time and then will be discarded in order to avoid security issue. Once this data will be available and the user will conform the acquisition will, the acquisition process will end and the checkout process will be initiating. The e-commerce module will set-up a secure connection with the entity managing the payment and send the data structure filled by the user in order to the payment.



The front-end forwards all to the e-commerce module that proceeds to finalise the payment procedure with the third party managing financial issues; once the payment procedure is successfully concluded the e-commerce module send the e-mail invoice to the user and closes the secure connection prior to return control to the application front-end that will inform the user of the successful completion of the purchase.

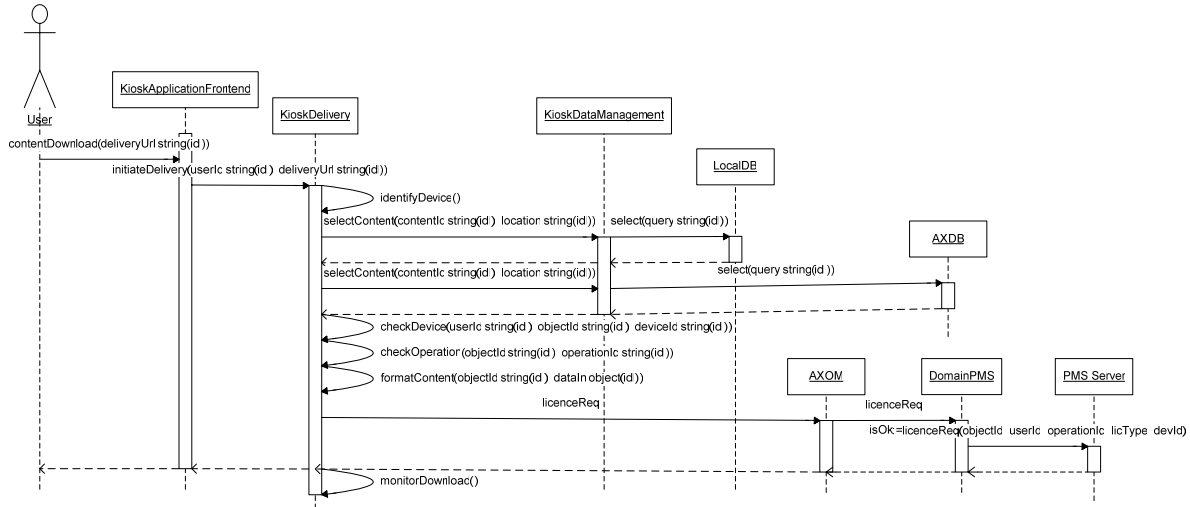
A possible aspect for the GUI related to this operation is reported in the following image where functional areas are placed in evidence.



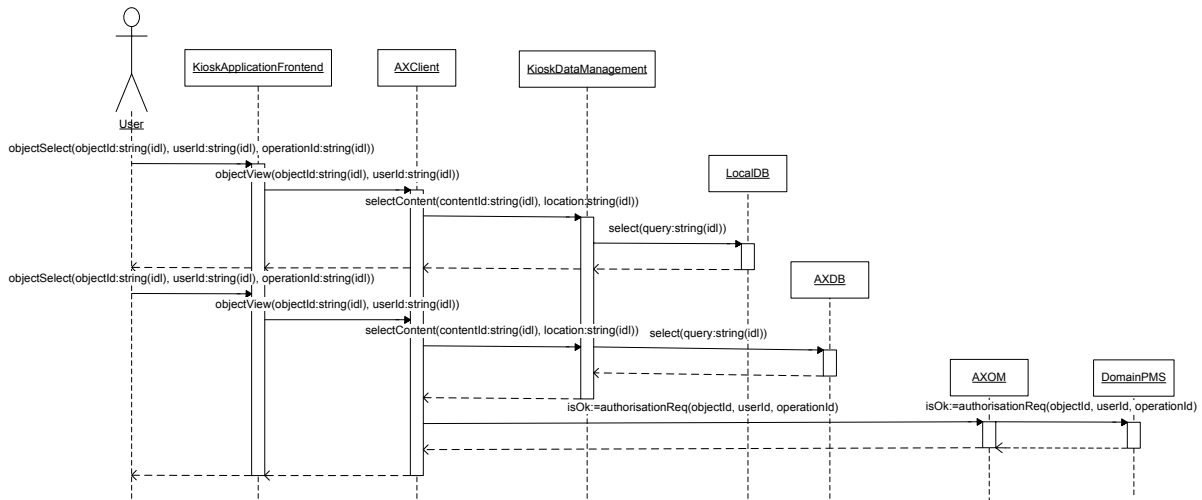
6.3.1.7 Content delivery

The delivery module to properly deliver data has to identify the delivery target device (terminal, PDA...), and then act consequently. The delivery module adapts the content to the fruition device (if necessary), retrieves device data (kind, storage, certificate...), and performs required checks, if these are positive the

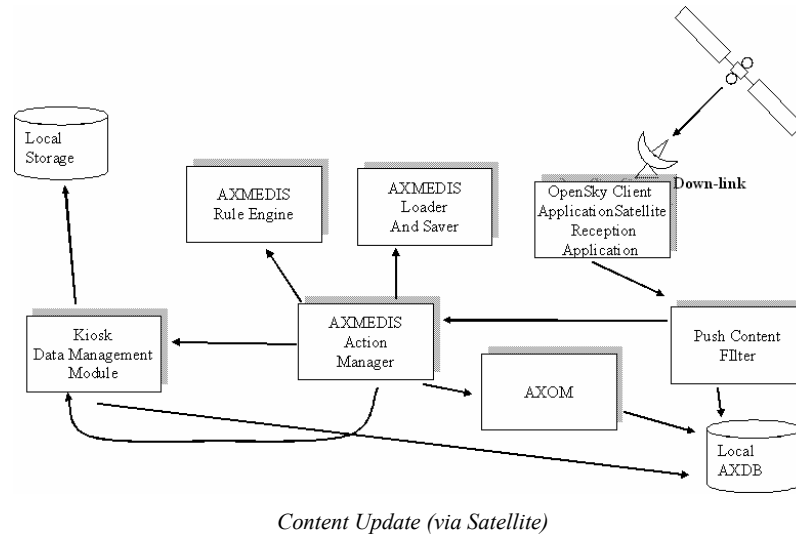
delivery module returns the front-end the download URL for activation. The user activates the download (a positive result to previous step is assumed here and the user should be free to decide the local storage position on the PDA). The delivery module takes the content from the local storage and the kind of operation requested. As the most complex case is the one of a delivery onto a user own device we will describe this instance.



The delivery module requires the domain PMS to generate a “device based” license, once this is returned, the delivery module requires the AXCS to generate the due keys. Once the AXCS returns the delivery module loads data onto the PDA monitoring the download; once this ends the delivery module notifies the application front-end of successful closure of the check out procedure; the user can now use the content according to acquired rights via the AXMEDIS viewer (as schematically represented hereafter).



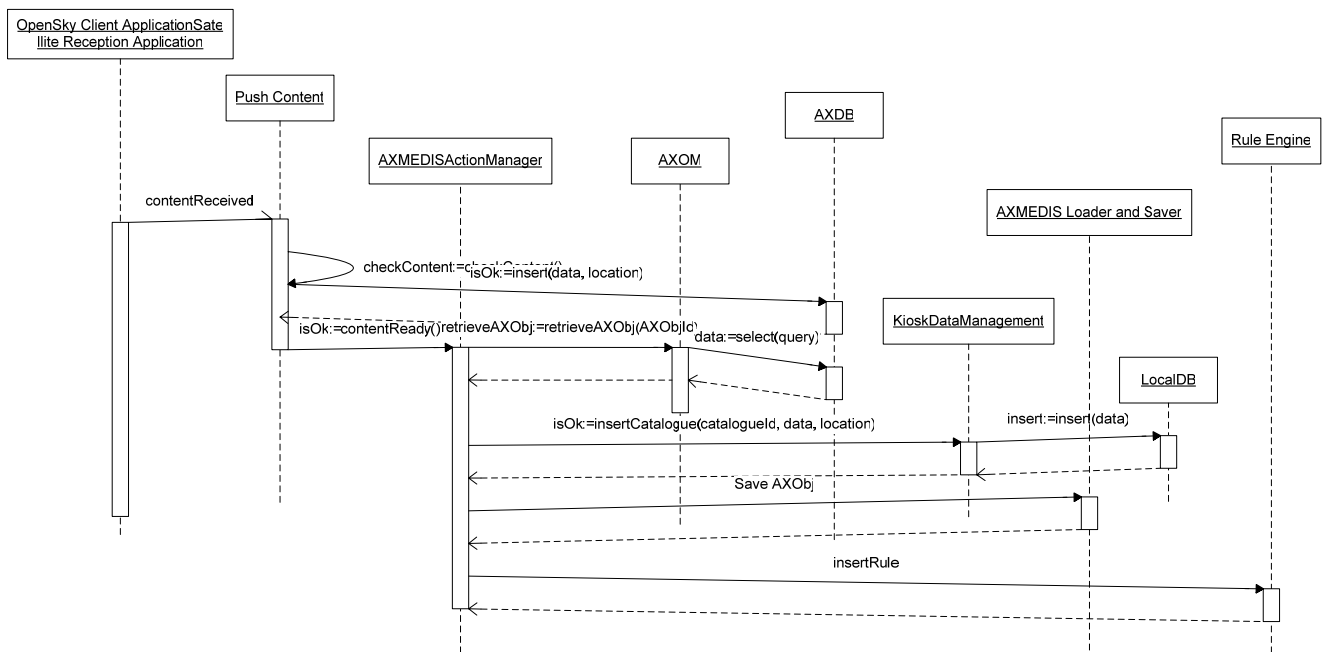
6.3.1.8 System maintenance



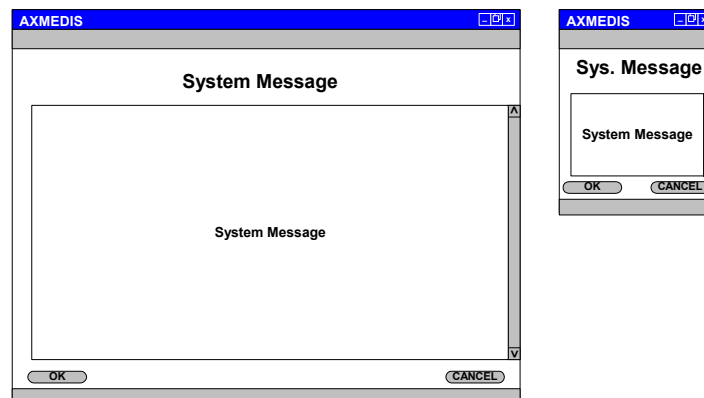
We assume that the starting point is when the checking time is over a Down-Link channel check has to be performed, the OpenSky Client Application Satellite Reception checks for data availability, we assume that data is available and therefore is downloaded and progressively cached locally till when all data to be received is stored locally (were it not a further check would be scheduled and the application would enters wait mode). All the data are AXMEDIS objects

The OpenSky Client Application Satellite Reception activates the Push Content Filter to decide how to proceed, the Push Content Filter verify consistency check on received data and if result is positive returns ACK, stores objects in the AXDB and notify the presence of new objects to the AXMEDIS Action Manager, if result is negative requires the distribution server to resend the damaged packages via Up-link. To have more information about the satellite functionalities , refers to chapter 4 of this document. Once this step is over the AXMEDIS Action Manager retrieves the data from the local AXDB, extracts the content from the AXMEDIS Object using an AXOM instance, checks the received data to determine what it is and perform the related action:

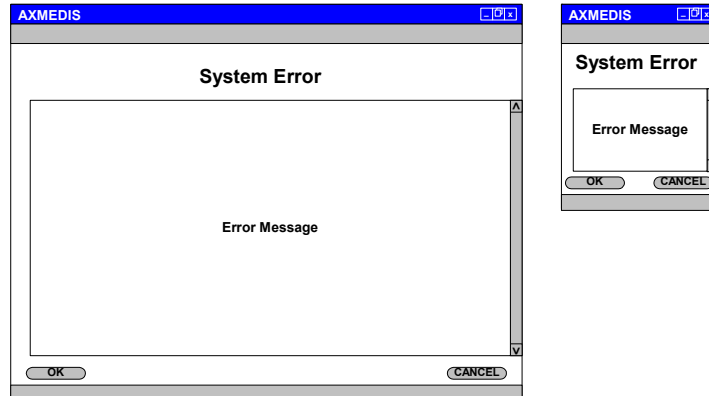
- if it is an AXMEDIS object, store it on AXMEDIS DB using loader and saver
- if it is a kiosk catalogue store it on the kiosk local DB
- if it is a formatting rule, store it using rule engine
- if it is a module update, proceed to its installation



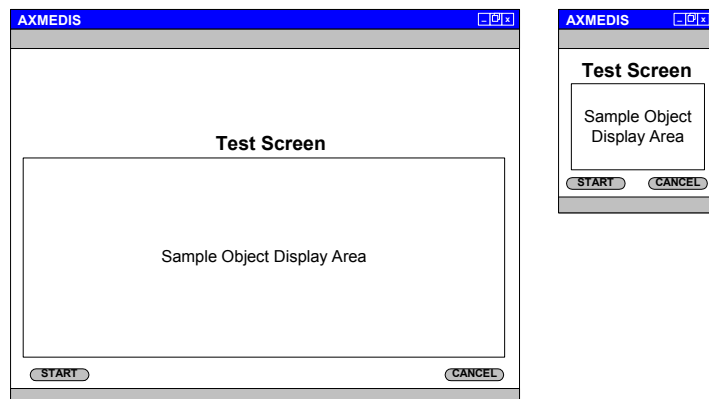
As previously mentioned there is also much side info that will need to be conveyed to the user during operation like messages, error notifications, test pages and so on. A possible aspect for the related GUI pages is reported in the following images.



Possible aspect of a system message page for both regular & PDA GUI

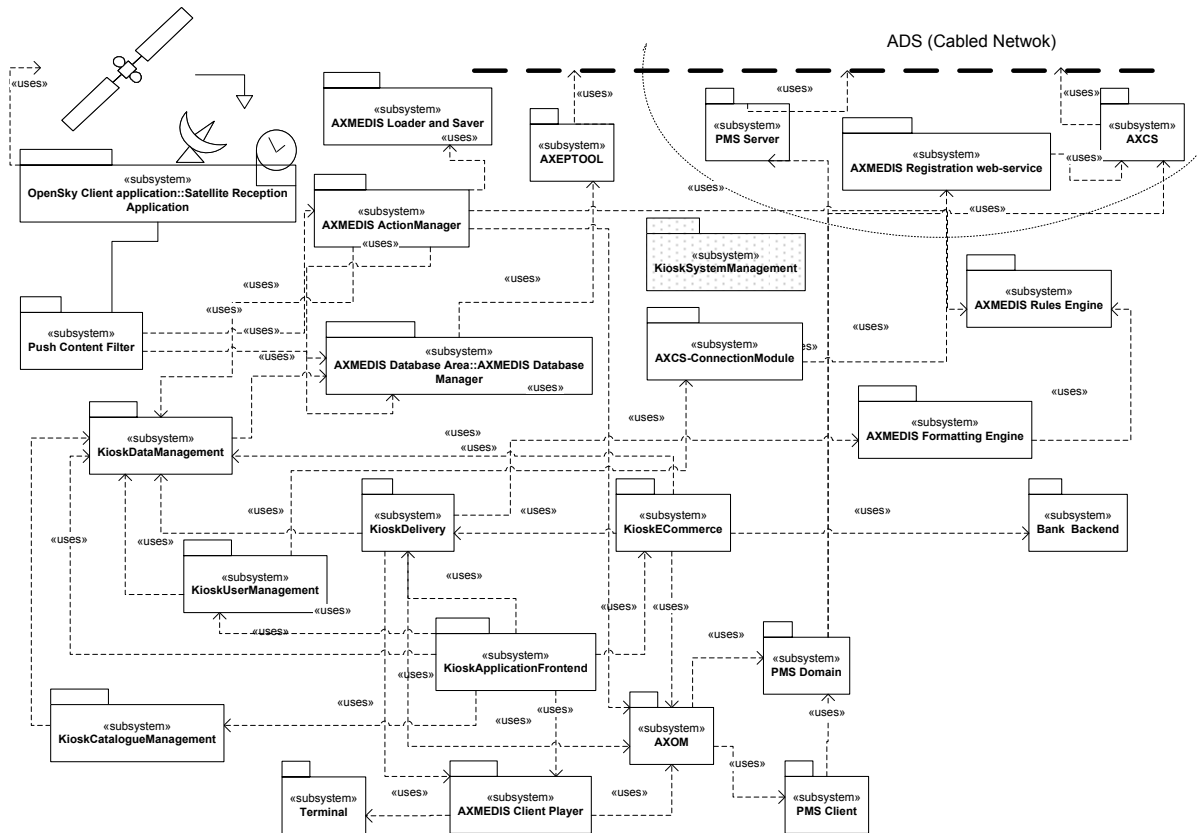


Possible aspect of an error message page for both regular & PDA GUI



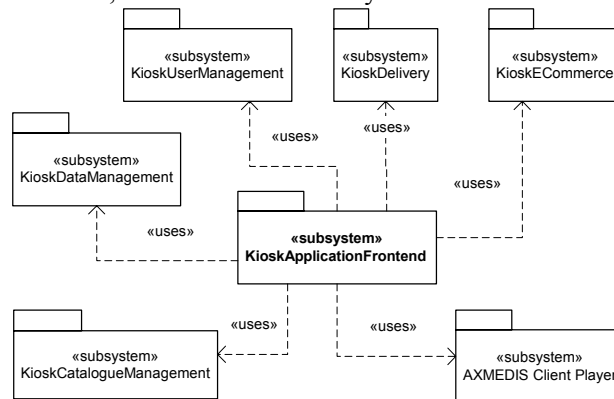
Possible aspect of a test page for both regular & PDA GUI

6.4 Kiosk Server Architecture



6.4.1 Kiosk Server Architecture: Application Front-end Management

Core of the kiosk application server, handles all kiosk subsystems and interfaces with the AXMEDIS player.



Module Profile	
Kiosk Application Front-end Module	
Executable or Library(Support)	Static Library
Single Thread or Multithread	Multithread
Language of Development	.NET framework (ASP.NET)
Responsible Name	Andrea Lorenzon
Responsible Partner	ILABS

Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux (MONO)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDISClientPlayerConnector	AXMEDIS Client Player	Web Services or API
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Web User Interface	ASP.NET, c#	.NET framework libraries
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

6.4.1.1 Kiosk Application Front-end API

```

interface KioskApplicationFrontend
{
    string langSelect(in string langId);
    boolean langSetup(in string langId);

    string logonReq();
    string logon(in string user, in string password);
    boolean logoff(in string user);

    string registrationReq();
    boolean registerUser(in string userData);
    string setUserProfile(in string userId);
    string getUserProfile(in string userId);

    boolean checkData (in string data);

    boolean loadCatalogue(in string userId, in string catalogueId);
    string browseCatalogue(in string catalogueId, in string pageId);

    boolean objectSelect(in string objectId, in string userId, in string operationId);
    string addToChart(in string objectId, in string userId);
    string checkOutReq(in string userId, in string chartId);
    string paymentDataReq(in string userId, in string chartId);
    string checkOut(in string checkOutForm);
    object contentDownload(in string deliveryUrl);
};

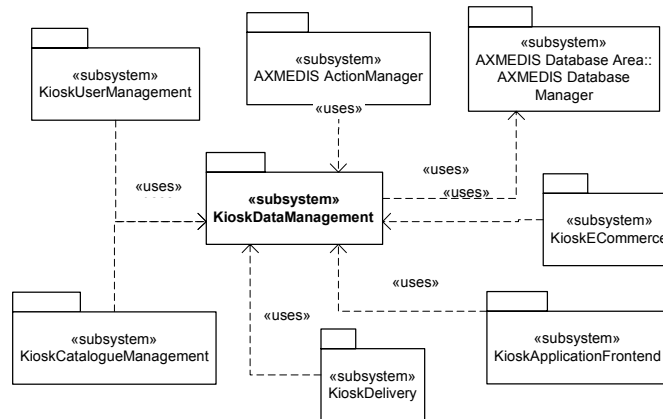
```

Function	Description	Method	Parameters	Reply
Language selection	Allows the user to request a specific language for the GUI	langSelect	string langId	string selectPage
Language selection	Performs the GUI language selection requested by the user	langSetup	string langId	boolean isOk
Logon	Initiates the logon process and retrieves the logon page	logonReq	-	string logonForm
Logon	Allows the user to login into the system, sets user privileges.	logon	string user string password	string sessionId

Function	Description	Method	Parameters	Reply
Logoff	Revokes user privileges.	logoff	string userId	boolean isOk
User registration	Requires the user management to provide the registration form	registrationReq	-	string registrationForm
User registration	Requires the user to provide the registration data	registerUser	string userData	boolean isOk
User registration	Stores user profile data	setUserProfile	string userId string userProfile	boolean isOk
User registration	Retrieves user profile data	getUserProfile	string userId	string userProfile
Data check	Performs basic validity check on inserted data.	checkData	string data	boolean isOk
Load a catalogue	Retrieves a Catalogue object, in terms of its stored data.	loadCatalogue	string userId string catalogueId	boolean isOk
Browse catalogue	Allows the user to browse the catalogue pages according to a pre-defined template	browseCatalogue	string catalogueId string pagId	string cataloguePage
Object selection	Allows to select an object specifying a requested operation	objectSelect	string objectId string userId string operationId	boolean isOk
Adding objects to chart	Allows the user to add a specific object to the chart	addToChart	string objectId string userId string operationId	string chartId
Asking for payment	Allows to initiate a checkout procedure	checkOutReq	string userId string chartId	string checkOutForm
Acquire objects	Allows to input data needed for initiating a checkout	checkOut	string checkOutForm	string deliveryUrl
Asking for payment data	Allows to require payment for objects currently in the chart	paymentDataReq	string userId string chartId	string paymentDataForm
Download objects	Allows the user to start content delivery procedure	contentDownload	string deliveryUrl	object content

6.4.2 Kiosk Server Architecture: Data Management

The present section describes the data management module providing a single point of interface to the local database and to the AXMEDIS system for all other kiosk subsystems.



Module Profile	
Kiosk Data Management Module	
Executable or Library(Support)	Static Library
Single Thread or Multithread	Multithread
Language of Development	.NET framework (c#)
Responsible Name	Andrea Lorenzon
Responsible Partner	GILABS
Status (proposed/approved)	Proposed
Platforms supported	Microsoft Windows, Linux (MONO)

Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXActionManagerConnector	AXMEDIS Action Manager	Web Services or API
AXDBConnector	AXMEDIS DataBase Manager	Web Services or API
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

6.4.2.1 Data Management API

```

interface KioskDataManagement
{
    string selectUser(in string userId, in string location);
    boolean checkUser(in string userId, in string data, in string location);
    boolean insertUser(in string userId, in string data, in string location);
    boolean updateUser(in string userId, in string data, in string location);
    boolean deleteUser(in string userId, in string location);

    string selectContent(in string contentId, in string location);
    boolean checkContent(in string contentId, in string data, in string location);
    boolean insertContent(in string contentId, in string data, in string location);
    boolean updateContent(in string contentId, in string data, in string location);
    boolean deleteContent(in string contentId, in string location);

    string selectModule(in string moduleId, in string location);
    boolean checkModule(in string moduleId, in string data, in string location);
    boolean insertModule(in string moduleId, in string data, in string location);
    boolean updateModule(in string moduleId, in string data, in string location);
    boolean deleteModule(in string moduleId, in string location);

    string selectCatalogue(in string catalogueId, in string location);
    boolean checkCatalogue(in string catalogueId, in string data, in string location);
    boolean insertCatalogue(in string catalogueId, in string data, in string location);
    boolean updateCatalogue(in string catalogueId, in string data, in string location);
    boolean deleteCatalogue(in string catalogueId, in string location);

    string selectChart(in string chartId, in string location);
    boolean checkChart(in string chartId, in string data, in string location);
    boolean insertChart(in string chartId, in string data, in string location);
    boolean updateChart(in string chartId, in string data, in string location);
    boolean deleteChart(in string chartId, in string location);

    string searchByKeyword(in string keywordList, in string location);
};

```

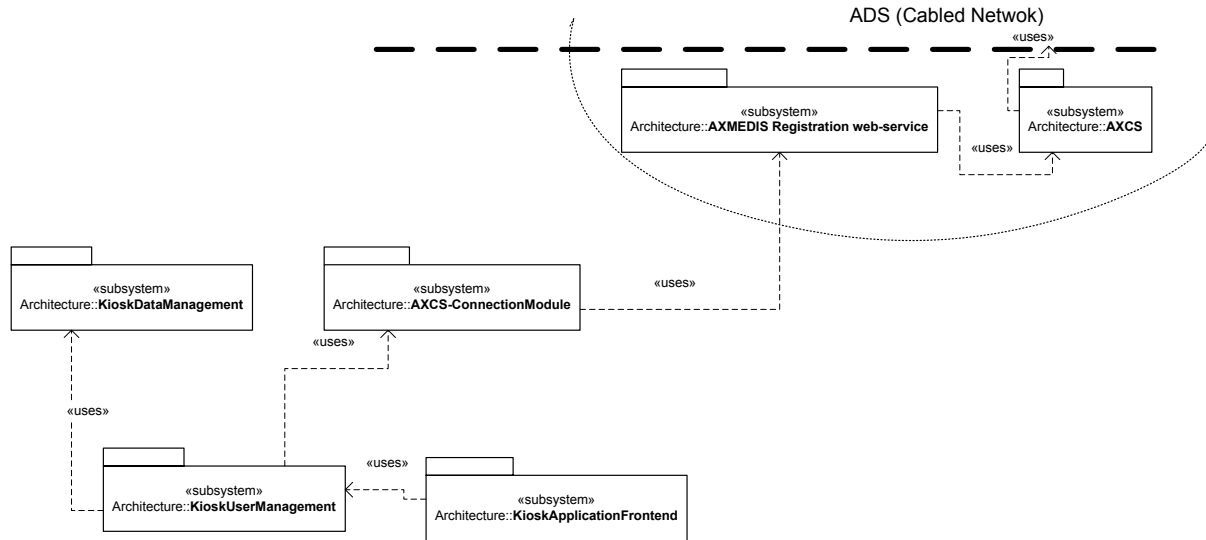
Function	Description	Method	Parameters	Reply
Select user data	Allows to select and retrieve user data from the specified location	selectUser	string userId string location	string data

Function	Description	Method	Parameters	Reply
Check user data	Allows to check user data in the specified location	checkUser	string userId string data string location	boolean isOk
Insert user data	Allows to insert user data in the specified location	insertUser	string userId string data string location	boolean isOk
Update user data	Allows to update user data in the specified location	updateUser	string userId string data string location	boolean isOk
Delete user data	Allow to delete user data from the specified location	deleteUser	string userId string location	boolean isOk
Select content data	Allows to select and retrieve content data from the specified location	selectContent	string contentId string location	string data
Check content data	Allows to check content data in the specified location	checkContent	string contentId string data string location	boolean isOk
Insert content data	Allows to insert content data in the specified location	insertContent	string contentId string data string location	boolean isOk
Update content data	Allows to update content data in the specified location	updateContent	string contentId string data string location	boolean isOk
Delete content data	Allow to delete content data from the specified location	deleteContent	string contentId string location	boolean isOk
Select module data	Allows to select and retrieve module data from the specified location	selectModule	string moduleId string location	string data
Check module data	Allows to check module data in the specified location	checkModule	string moduleId string data string location	boolean isOk
Insert module data	Allows to insert module data in the specified location	insertModule	string moduleId string data string location	boolean isOk
Update module data	Allows to update module data in the specified location	updateModule	string moduleId string data string location	boolean isOk
Delete module data	Allow to delete module data from the specified location	deleteModule	string moduleId string location	boolean isOk
Select catalogue data	Allows to select and retrieve catalogue data from the specified location	selectCatalogue	string catalogueId string location	string data
Check catalogue data	Allows to check user catalogue in the specified location	checkCatalogue	string catalogueId string data string location	boolean isOk
Insert catalogue data	Allows to insert catalogue data in the specified location	insertCatalogue	string catalogueId string data string location	boolean isOk
Update catalogue data	Allows to update catalogue data in the specified location	updateCatalogue	string catalogueId string data string location	boolean isOk
Delete catalogue data	Allow to delete catalogue data from the specified location	deleteCatalogue	string catalogueId string location	boolean isOk
Select chart data	Allows to select and retrieve chart data from the specified location	selectChart	string chartId string location	string data
Check chart data	Allows to check user chart in the specified location	checkChart	string chartId string data string location	boolean isOk
Insert chart data	Allows to insert chart data in the specified location	insertChart	string chartId string data string location	boolean isOk
Update chart data	Allows to update chart data in the specified location	updateChart	string chartId string data string location	boolean isOk
Delete chart data	Allow to delete chart data from the specified location	deleteChart	string chartId string location	boolean isOk
Search by keyword	Initiates a search by specifying a list of keywords.	searchByKeyword	string keywordList string location	string data

6.4.3 Kiosk Server Architecture: User Management

In the following diagrams are reported the main operations related to the user management (user side) along with the related foreseen graphic user interfaces.

All reported operations are foreseen also in the use-case diagram reported at the beginning of this section describing the overall functioning of the system.



Module Profile		
Kiosk User Management Module		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Multithread	
Language of Development	.NET framework (c#)	
Responsible Name	Andrea Lorenzon	
Responsible Partner	GILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux (MONO)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXCSAPI	AXCS	Web Services or API
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

6.4.3.1 User Management API

```

interface KioskUserManagement
{
    string logonReq();
    boolean logon(in string user, in string password);
    boolean logoff(in string user);

    string registrationReq();
    boolean registerUser(in string userData);
    boolean registrationConfirm(in string userData, in string userMail);

    string getUsersList();
    string getUser(in string userId);
    boolean checkUser(in string userId, in string userData);
    boolean createUser(in string userId, in string userData);
    boolean modifyUser(in string userId, in string userData);
    boolean removeUser(in string userId);

    string setUserProfile(in string userId);
    string getUserProfile(in string userId);

    boolean createUserCatalogueAssociation(in string userId, in string catalogueId);
    boolean removeUserCatalogueAssociation(in string userId, in string catalogueId);
    string getUserCatalogueAssociation(in string userId);
    string getUsersAssociatedToCatalogue(in string catalogueId);
    string getCataloguesAssociatedToUser(in string userId);
};

```

Function	Description	Method	Parameters	Reply
Logon	Initiates the logon process and retrieves the logon page	logonReq	-	string logonForm
Logon	Allows the user to login into the system, sets user privileges.	logon	string user string password	string sessionId
Logoff	Revokes user privileges.	logoff	string userId	boolean isOk
User registration request	Allows to retrieve the registration form needed to provide the system all needed user data	registrationReq	-	string registrationForm
User registration	Allows to retrieve the user registration data	registerUser	string userData	boolean isOk
User registration	Confirms the user registration and notifies the user the registration data (user, password...)	registrationConfirm	string userData string userMail	boolean isOk
Retrieve users list	Retrieves the list (userId) of user registered.	getUsersList	-	string userList
Retrieve user	Retrieves a user object, in terms of its stored data.	getUser	string userId	string userData
User data check	Allows to check user data provided a user ID	checkUser	string userId string userData	boolean isOk
Create user	Creates a new user to be handled by the kiosk	createUser	string userId string userData	boolean isOk
Modify user	Modifies a database user object associated with a given user ID.	modifyUser	string userId string userData	boolean isOk
User data check	Allows to check user data provided a user ID	removeUser	string userId	boolean isOk
User profile	Retrieves the user profile provided a user ID	setUserProfile	string userId string userProfile	boolean isOk
User profile	Retrieves the user profile provided a user ID	getUserProfile	string userId	string userProfile
Remove user	Removes a user form those to be handled by the kiosk	removeUser	string userId	boolean isOk
Create user -catalogue association	Establishes an association between a catalogue and a user (this is an extension)	createUserCatalogueAssociation	string userId string catalogueId	boolean isOk

Function	Description	Method	Parameters	Reply
Remove user - catalogue association	Remove user-catalogue association.	removeUserCatalogueAssociation	string userId string catalogueId	boolean isOk
Retrieve user - catalogue association	If there is the association retrieves the catalogue (catalogueId) associated with the user (userId), else retrieves a null object.	getUserCatalogueAssociation	string userId	string catalogueId
Retrieve users - catalogue association	Retrieves all users associated with a given catalogue (catalogueId).	getUsersAssociatedToCatalogue	string catalogueId	string userList
Retrieve user - catalogues association	Retrieves all catalogues associated with a given user (userId).	getCataloguesAssociatedToUser	string userId	string catalogueList

6.4.3.2 User Management external connection

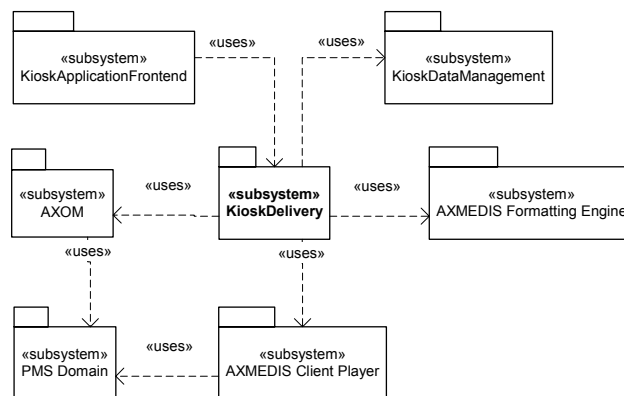
This module interfaces with the AXMEDIS AXCS component using the following interfaces:

```
interface AxcsConnectionModule
{
    boolean connect();
    boolean checkUser(in string userData);
    boolean registerUser(in string userData);
};
```

Function	Description	Method	Parameters	Reply
Connect module	Try to connect to the module to check if it is on	connect	-	boolean isOk
Check user registration	Check if the user is already registered in the AXMEDIS system	checkUser	string userData	boolean exist
Register User	Register the user on the AXMEDIS system	registerUser	String userData	boolean isOk

6.4.4 Kiosk Server Architecture: Delivery Module

This section describes the delivery module, the one in charge of delivering content to the user either on the kiosk terminal or on a mobile device (PDA or smart-phone). The module will also take charge of retrieving content from the data manager regardless of its actual location (local or remote) as this latter point will be masked by the data management functioning.



Module Profile	
Kiosk Delivery Module	
Executable or Library(Support)	Static Library
Single Thread or Multithread	Multithread

Language of Development	.NET framework (c#)	
Responsible Name	Andrea Lorenzon	
Responsible Partner	GILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux (MONO)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDISClientPlayerConnector	AXMEDIS Client Player	Web Services or API
AXMEDISFormattingEngineConnector	AXMEDIS Formatting Engine	Web Services or API
AXOMConnector	AXOM	Web Services or API
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

6.4.4.1 Delivery Module API

```

interface KioskDelivery
{
    boolean initiateDelivery(in string userId, in string deliveryUrl);

    string identifyDevice();
    boolean checkDevice(in string userId, in string objectId, in string deviceId);
    boolean checkOperation(in string objectId, in string operationId);

    object fomatContent(in string objectId, in object dataIn);

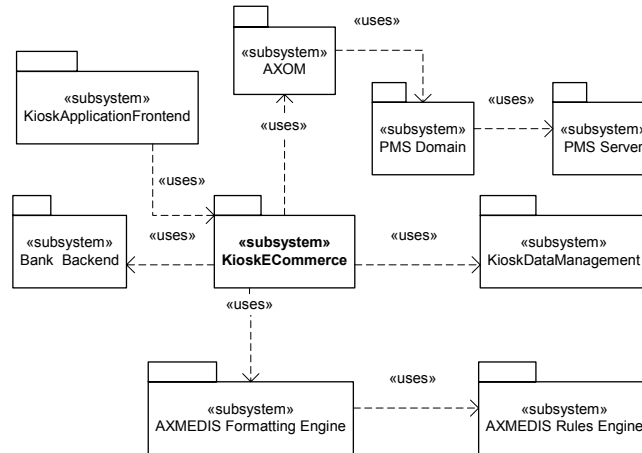
    boolean monitorDownload();
};

```

Function	Description	Method	Parameters	Reply
Delivery	Allows the system to start the delivery process	initiateDelivery	-	boolean isOk
Identify device	Allows to identify the delivery target device	identifyDevice	string userId string deliveryUrl	string deviceId
Check device	Allow to verify delivery target device characteristics	checkDevice	string userId string objected string deviceId	boolean isOk
Check operation	Allow to check operation requested on selected content that will be then delivered	checkOperation	string objected string operationId	boolean isOk
Format content	Allow to properly format content according to delivery target	fomatContent	string objected object dataIn	object dataOut
Monitoring download	Allows to monitor the download operation	monitorDownload	-	boolean isOk

6.4.5 Kiosk Server Architecture: e-Commerce Module

This section describes the e-commerce module that will be dealing with all operations related to purchase, acquisition, rental and payment based fruition of content.



Module Profile		
Kiosk eCommerce Module		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Multithread	
Language of Development	.NET framework (c#)	
Responsible Name	Andrea Lorenzon	
Responsible Partner	GILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux (MONO)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
BankConnector	Bank system	Web Services or API
AXOMConnector	AXOM	Web Services or API
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

6.4.5.1 e-Commerce Module API

```
interface KioskECommerce
{
    string secureConnOpen();
    boolean secureConnClose(in string connectionId);

    string addToChart(in string userId, in string objected, in string operationId);
    boolean removeFromChart(in string charted, in string objected, in string operationId);

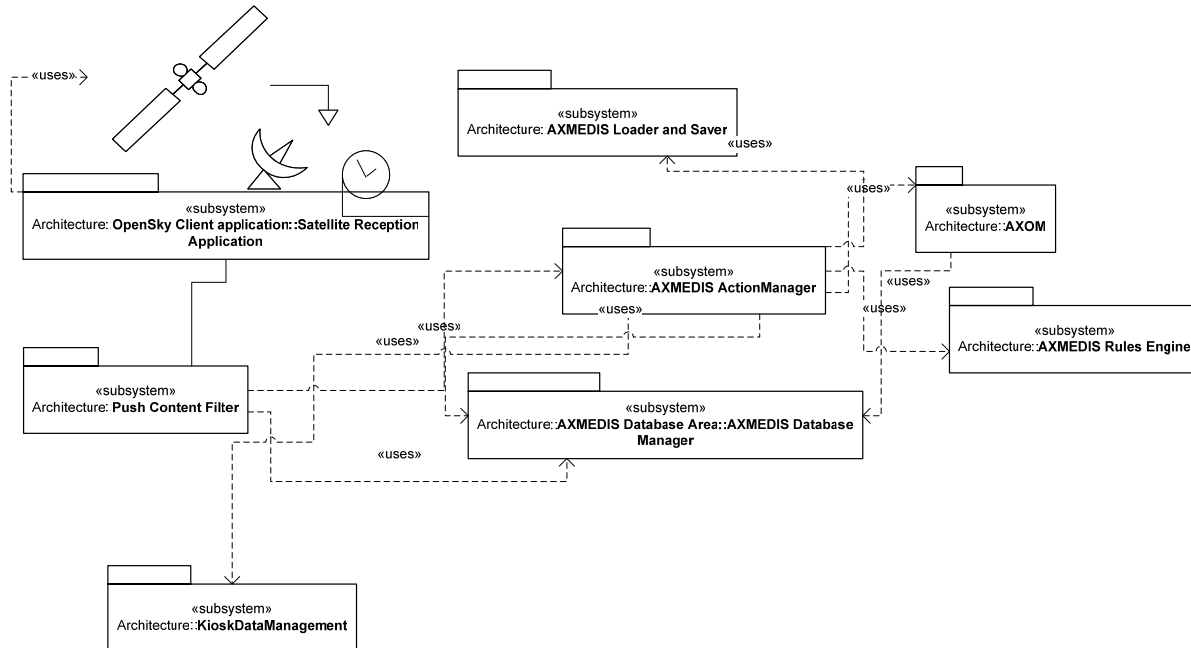
    string getChart(in string userId, in string chartId);
    boolean checkChart(in string chartId);

    boolean checkOut(in string userId, in string chartId);
    string computePrice(in string chartId, in string userId, in string par);
    string paymentDataReq(in string userId, in string chartId);
    boolean paymentConfirm(in string userId, in string chartId);
    boolean userInvoice(in string userId, in string chartId);
};
```

Function	Description	Method	Parameters	Reply
Connect	Allows the system to perform a secure connection to the entity managing the payment	secureConnOpen	-	string connectionId
Disconnect	Allows the system to close a secure connection with the entity managing the payment	secureConnClose	string connectionId	boolean isOk
Add object to chart	Add an AXMEDIS object to the user chart	addToChart	string userId string objectId string operationId	string chartId
Remove object from chart	Remove an AXMEDIS object from the user chart	removeFromChart	string chartId string objectId string operationId	boolean isOk
Retrieve Chart	Retrieve the chart associated to the user	getChart	string userId string chartId	string chartData
Check cart	Allow the system to perform a check on a given chart id	checkChart	string chartId	boolean isOk
Check out	Allows the system to perform the actual payment of what held in the chart	checkOut	string userId string chartId	boolean isOk
Compute prices	Allows the system to compute the total due amount that the user should pay during the check-out phase	computePrice	string chartId string userId string par	string chartData
Purchase chart objects	Retrieve all billing information of the AXMEDIS object contained in the chart	paymentDataReq	string userId string chartId	string paymentForm
Payment confirm	Allows the system to finalise the payment procedure for a specified chart	paymentConfirm	string userId string chartId	boolean isOk
Invoice user	Allows the system to invoice the user according to what has been requested in the payment request	userInvoice	string userId string chartId	boolean isOk

6.4.6 Kiosk Server Architecture: Satellite Reception

This module is the one that will handle the reception via satellite in push of content and updates (both applicative and in terms of rules...).



Module Profile		
Satellite Reception (ActionManager)		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Multithread	
Language of Development	.NET framework (c#)	
Responsible Name	Andrea Lorenzon	
Responsible Partner	GILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux (MONO)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDISSatelliteReceptionConnector	Push Content Filter	Web Services or API
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

6.4.6.1 Satellite Reception API

```

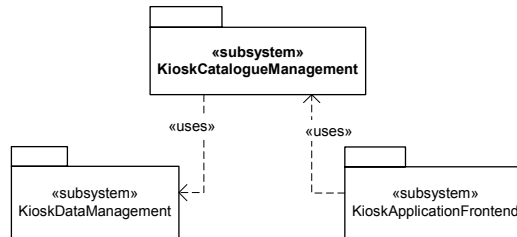
interface AXMEDISActionManager
{
    string contentReceived();
};

```

Function	Description	Method	Parameters	Reply
Content reception	Content sent in push to the kiosk exploiting satellite broadcast has been received and is ready for usage	contentReceived	-	boolean contentId

6.4.7 Kiosk Server Architecture: Catalogue Management

In the present section is presented the module in charge of the kiosk catalogue management. The module is the one implementing the business logic related to catalogue management and will be the one where kiosk specific customisation will be applied.



Module Profile		
Kiosk Catalogue Management Module		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Multithread	
Language of Development	.NET framework (c++/c#)	
Responsible Name	Andrea Lorenzon	
Responsible Partner	GILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux (MONO)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model, language, etc.	Library used for the development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or

		not

6.4.7.1 Catalogue Management interfaces description

```

interface KioskCatalogueManagement
{
    string listCatalogue ();
    string createCatalogueReq(in string userProfile, in string description, in string creationDate, in string validityDate);
    boolean createCatalogue(in string catalogueId, in string catalogueTemplate, in string catalogueData);
    boolean updateCatalogue(in string catalogueId, in string catalogueTemplate, in string catalogueData);
    boolean deleteCatalogue(in string catalogueId);
    string catalogueLoadSchedule(string loadSchedule);

    string getCatalogueTemplate();
    string formatCataloguePage(in string catalogueId, in string pageId);
    string loadCataloguePage(in string catalogueId, in string pageId);
    boolean displayCataloguePage(in string catalogueId, in string pageId);
};

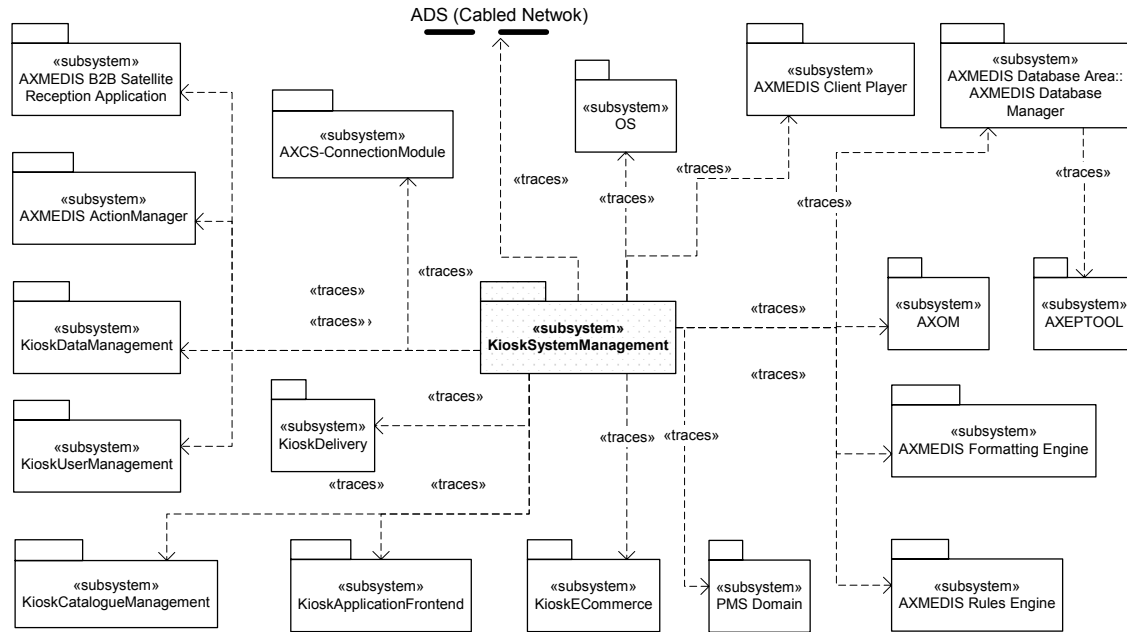
```

6.4.7.2 Catalogue Management API

Function	Description	Method	Parameters	Reply
List catalogue	Allows to retrieve the list of available catalogues	listCatalogue	-	string catalogueList
Catalogue creation	Allows to request a catalogue creation starting from available objects and catalogue template	createCatalogueReq	string userProfile string description string creationDate string validityDate	string catalogueId
Catalogue creation	Allows to create a catalogue starting from available objects and catalogue template	createCatalogue	string catalogueTemplate string catalogueId string catalogueData	boolean isOk
Catalogue update	Allows to create a catalogue starting from available objects and catalogue template	updateCatalogue	string catalogueTemplate string catalogueId string catalogueData	boolean isOk
Delete catalogue	Delete a catalogue object associated with a given ID.	deleteCatalogue	string catalogueId	boolean isOk
Schedule loading	Creates a “schedule” entry for a catalogue loading operation	catalogueLoadSchedule	string loadSchedule	string catalogueId
Template retrieval	Allows to retrieve the specific template according to which a catalogue should be formatted	getCatalogueTemplate	-	string catalogueTemplate
Catalogue formatting	Allows to format a catalogue for proper management onto the kiosk	formatCataloguePage	string catalogueId string pageId	string cataloguePage
Catalogue loading	Allows to load a specific catalogue page into memory for subsequent management	loadCataloguePage	string catalogueId string pageId	string cataloguePage
Catalogue display	Allows to display a selected catalogue page	displayCataloguePage	string catalogueId string pageId	boolean isOk

6.4.8 Kiosk Server Architecture: System Management

In the following section is described the kiosk system management module, whose role is to monitor and provide info on kiosk components overall functioning so to ensure the best possible user service during system up-time, signalling out-of-service or anomalous conditions.



Module Profile		
Kiosk System Management Module		
Executable or Library(Support)	Static Library	
Single Thread or Multithread	Multithread	
Language of Development	.NET framework (c++/c#)	
Responsible Name	Andrea Lorenzon	
Responsible Partner	GILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows, Linux (MONO)	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
BankConnector	Bank system	Web Services or API
AXMEDISDomainPMSConnector	AXMEDIS Domain PMS	Web Services or API
AXOMConnector	AXOM	Web Services or API
AXMEDISRuleEngineConnector	AXMEDIS Rule Engine	Web Services or API
AXMEDISFormattingEngineConnector	AXMEDIS Formatting Engine	Web Services or API
AXCSConnectorModule	AXCS	Web Services or API
AXMEDISClientPlayerConnector	AXMEDIS Client Player	Web Services or API
AXMEDISSatelliteReceptionConnector	AXMEDIS B2B Satellite Reception Application	Web Services or API
AXActionManagerConnector	AXMEDIS Action Manager	Web Services or API
AXDBCConnector	AXMEDIS DataBase Manager	Web Services or API
File Formats Used	Shared with	File format name or reference to a section
User Interface	Development model,	Library used for the

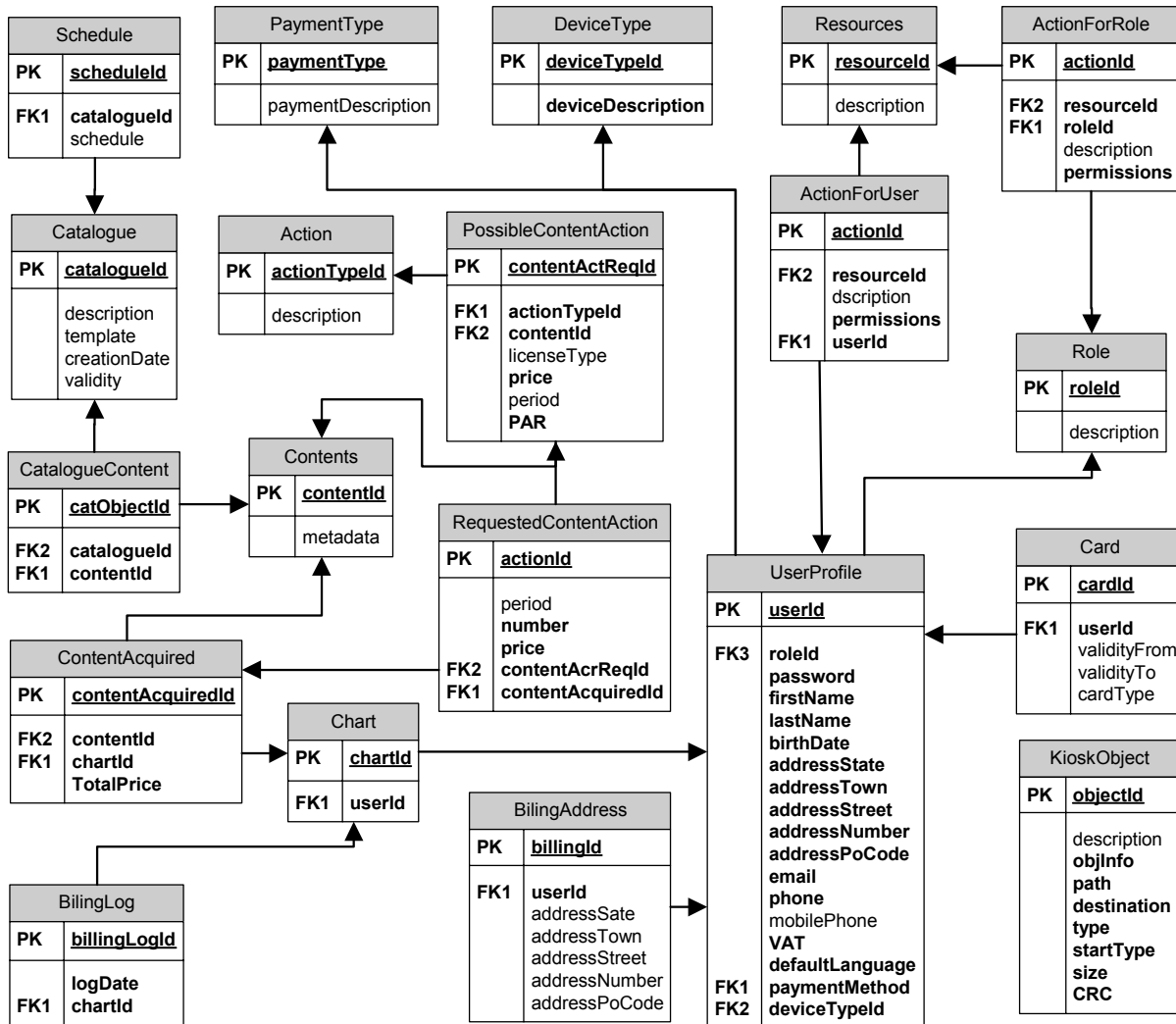
	language, etc.	development, platform, etc.
Used Libraries	Name of the library and version	License status: GPL. LGPL. PEK, proprietary, authorized or not

6.4.8.1 System Management API

```
interface KioskSystemManagement
{
    string startModule(in string moduleName);
    string checkModule(in string moduleId);
    boolean stopModule(in string moduleId);
};
```

Function	Description	Method	Parameters	Reply
Start a module	Allows the system manager to manually start a module or the system to perform an automatic component start-up	startModule	string moduleName	String moduleId
Check a module	Retrieves the status of a module to ensure proper system overall functioning monitoring	checkModule	string moduleId	String moduleStatus
Stop a module	Allows the system manager to manually stop a module or the system to perform an automatic component halt	stopModule	string moduleId	boolean isOk

6.5 Database Entity Relationship



6.6 Session object as value of the Attribute parameter

6.6.1 User object

The following table describes the user related data structure as devised within the kiosk application. Personal data are considered optional. Some billing related information is recorded within this data structure as they are inherently related to the user performing the operation. If the user would not like to store this info during the registration phase, yet will have to provide it during the billing phase, but data provided during the billing phase will not be permanently stored unless provided during registration phase or during personal profile management.

Attribute	Type	Description
firstName	String	User first name
lastName	String	User last name
userId	String	Username
type	String	{role, role...} Set of allowed roles for the specified user encompassing: Administrator, End-user...
userPermissions	String	{{permissions} {permissions}...} The set of permissions granted to the user

Attribute	Type	Description
defaultLanguage	Int	The default language for the GUI and content (the default value will be English)
birthDate	String	The user date of birth
AddressState	String	User preferred mail address (state)
addressTown	String	User preferred mail address (town)
addressStreet	String	User preferred mail address (street)
addressNumber	String	User preferred mail address (number)
addressPoCode	String	User preferred mail address (postal area code)
phone	String	User phone number
mobile	String	User mobile phone number
email	String	User e-mail address
VAT	String	User VAT code number
paymentMethod	Int	Preferred payment method: 0 = Pre-paid-cards / 1 = Credit card / 2 = Other
cardRefernce	String	Reference to a temporary structure holding user chosen credit card(s)
billingRefernce	String	Reference to a temporary structure holding user chosen billing info
device	Int	0 = PDA / 1 = Smartphone / 2 = Other

6.6.2 User card object

The following table describes the temporary data structure used to hold user credit card info during the billing and check-out operations..

Attribute	Type	Description
cardId	String	User card number
validFrom	String	Date of validity (From)
validTo	String	Date of validity (To)
cardType	String	User card type

6.6.3 User billing address object

The following table describes the temporary data structure used to hold user billing info during the billing and check-out operations.

Attribute	Type	Description
billingAddress	String	User preferred billing mail address
billingPhone	String	User billing phone number
billingEmail	String	User billing e-mail address
billingPoCode	String	User preferred mail address (postal area code)
billingVAT	String	User billing VAT code number

6.6.4 User list object

The following table describes the object list related data structure as devised within the kiosk application. This is a special data structure devised just to handle specific operations inside the kiosk management. Other specific structures have been defined to manage objects in other context.

Attribute	Type	Description
userObjectArrayList	ArrayList	Array list of User s

6.6.5 Catalogue object

The following table describes the catalogue related data structure as devised within the kiosk application. The catalogue will be the primary object that will be used by the end user in the interaction with the kiosk front-end application.

Attribute	Type	Description
catalogueID	string	Unique ID of the Catalogue

description	string	Textual description of the Catalogue
creationDate	string	Date of creation of the catalogue
validityDate	string	Date of validity of the catalogue
template	string	Template to apply to catalogue structure
axObjectList	arrayList	List of AXMEDIS objects contained in the catalogue

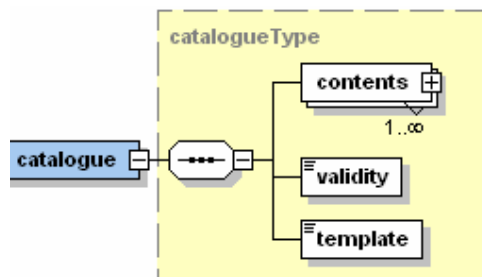
6.6.5.1 XML Representation of a catalogue

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="catalogue" type="catalogueType"/>
  <xs:complexType name="catalogueType">
    <xs:sequence>
      <xs:element name="catalogueId" type="xs:string"/>
      <xs:element name="content" type="contentType" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element name="description" type="xs:string"/>
      <xs:element name="creationDate" type="xs:string"/>
      <xs:element name="validity" type="xs:string"/>
      <xs:element name="template" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="contentType">
    <xs:sequence>
      <xs:element name="contentId" type="xs:integer"/>
      <xs:element name="metadata" type="xs:string"/>
      <xs:element name="action" type="actionType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="actionType">
    <xs:sequence>
      <xs:element name="actionId" type="xs:integer"/>
      <xs:element name="period" type="xs:string"/>
      <xs:element name="price" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

diagram



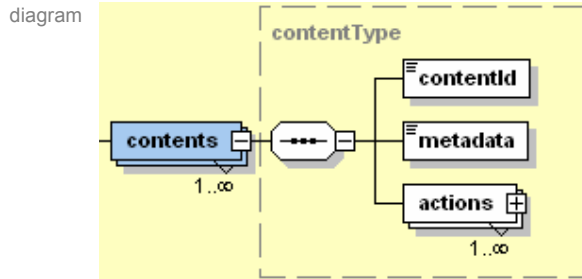
namespace <http://www.AXMEDIS.org/kiosk-schema>

type **pin:CatalogueType**

children **contents validity template**

description This element is the root element containing information of the kiosk catalogue. It contains all necessary information to manage a kiosk catalogue. The fields are:

- **validity** date of validity of the catalogue
- **template** template to apply to catalogue structure



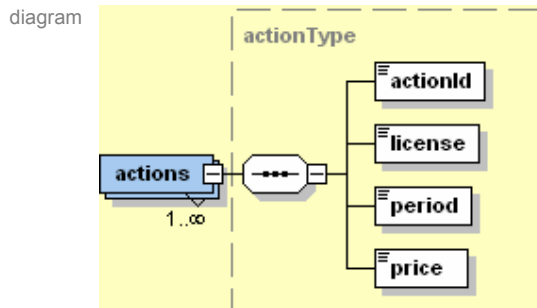
namespace <http://www.AXMEDIS.org/kiosk-schema>

type **pin:contentsType**

children **contentId metadata actions**

description This element contains all information about the contents of a catalogue. The fields are:

- **contentId** the identifier of the content. This is the unique identifier of the content in the kiosk system
- **metadata** metadata of the content. They are used to give all information about the content to the user



namespace <http://www.AXMEDIS.org/kiosk-schema>

type **pin:actionsType**

children **actionId license period price**

description This element contains all information about the possible actions on the content in the kiosk catalogue. The fields are:

- **actionId** the identifier of the action. This is the unique identifier of the action in the kiosk system
- **license** the license type linked to the action
- **period** the eventual period of validity of the action
- **price** the price of the action

6.6.6 Catalogue list object

The following table describes the catalogue object list related data structure as devised within the kiosk application. This is a special data structure devised just to handle specific operations inside the kiosk management. Other specific structures have been defined to manage objects in other context.

Attribute	Type	Description
catalogueObjectArrayList	ArrayList	Array List of Catalogue Object

6.6.7 Kiosk object

The following table describes the generic object related data structure as devised within the kiosk application. This structure will be used both by the kiosk application front-end and the kiosk data management module.

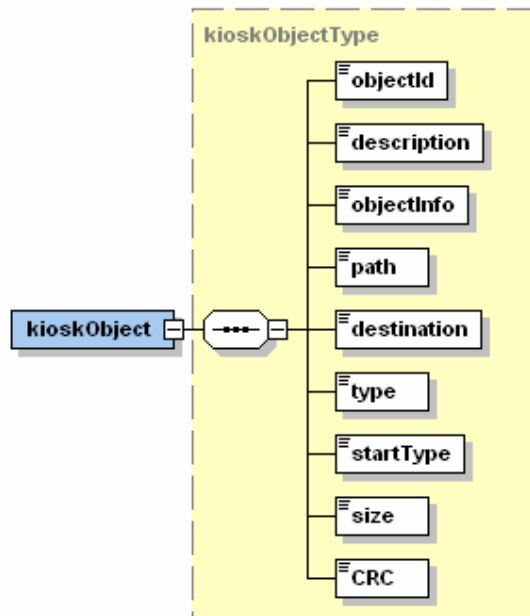
Attribute	Type	Description
objectId	String	Unique ID of the Kiosk Object
description	String	Textual description of the object
objInfo	String	XML field describing the object metadata
path	String	The URL, inside the kiosk environment, of the object
destination	String	The URL of the physical location of the object
typeId	Int	1 = Content / 2 = Update / 3 = Other
startType	String	Describes how the module will be used (via a loader, and exe...)
size	Long	Object size
CRC	Long	Object CRC to ensure data integrity

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="kioskObject" type="kioskObjectType"/>
  <xs:complexType name="kioskObjectType">
    <xs:sequence>
      <xs:element name="objectId" type="xs:string"/>
      <xs:element name="description" type="xs:string"/>
      <xs:element name="objectInfo" type="xs:string"/>
      <xs:element name="path" type="xs:string"/>
      <xs:element name="destination" type="xs:string"/>
      <xs:element name="type" type="xs:string"/>
      <xs:element name="startType" type="xs:string"/>
      <xs:element name="size" type="xs:float"/>
      <xs:element name="CRC" type="xs:float"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

diagram



namespace <http://www.AXMEDIS.org/kiosk-schema>

type **pin:kioskObjectType**

children **objectId description objectInfo path destination type startType size CRC**

description This element is the root element containing information of the kiosk object. It contains all necessary information to manage a kiosk object. The fields are:

- **objectId** unique id of the kiosk object
- **description** textual description of the kiosk object

- **objectInfo** object metadata
- **path** the URL, inside the kiosk environment, of the object
- **destination** the URL of the physical location of the object
- **type** the type of object (content, update, ...)
- **startType** how the module will be used (via a loader, and exe...)
- **size** object size
- **CRC** object CRC to ensure data integrity

6.6.8 Loading schedule object

The following table describes the catalogue loading schedule related data structure as devised within the kiosk application.

Attribute	Type	Description
scheduleId	String	The ID of the schedule.
catalogueId	String	The ID of the related catalogue.
schedule	String	CRON like schedule specifier (if empty, defaults to NOW). It specifies when to start a harvesting process, through the Mobile Agent

6.6.9 Billing object

The following table describes the billing related data structure as devised within the kiosk application. These data is complementary to the one devised inside user data structure. The present data structure will be filled exploiting a specific view on several other tables described in the DB entity relation diagramme.

Attribute	Type	Description
billingId	String	The ID of the billing information object.
chartId	String	The Id of the chart
totAmount	Float	The total amount of the billing
contentPriceList	Arraylist	The list of couple: AXMEDIS object title, price

6.6.10 Chart object

The following table describes the chart related data structure as devised within the kiosk

Attribute	Type	Description
chartID	String	The ID of the chart.
userId	String	The ID of the user
contentPurchaseList	ArrayList	The list of couple: object, operation type...

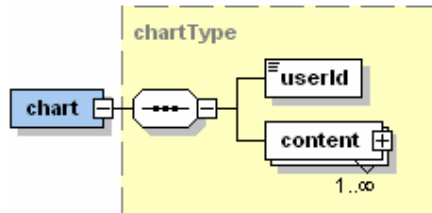
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="chart" type="chartType"/>
  <xs:complexType name="chartType">
    <xs:sequence>
      <xs:element name="userId" type="xs:string"/>
      <xs:element name="content" type="contentType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="contentType">
    <xs:sequence>
      <xs:element name="contentId" type="xs:string"/>
      <xs:element name="actions" type="actionsType"/>
      <xs:element name="totalPrice" type="xs:float"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="actionsType">
    <xs:sequence>
```

```

<xs:element name="action" type="actionType" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="actionType">
  <xs:sequence>
    <xs:element name="actionId" type="xs:string"/>
    <xs:element name="period" type="xs:string"/>
    <xs:element name="price" type="xs:float"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

diagram



namespace <http://www.AXMEDIS.org/kiosk-schema>

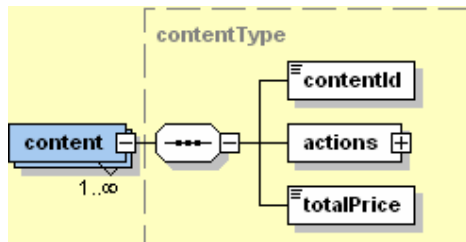
type **pin:chartType**

children **userId content**

description This element is the root element containing information of the chart object. It contains all necessary information to manage the chart of a user. The fields are:

- **userId** unique id of the user in the kiosk system

diagram



namespace <http://www.AXMEDIS.org/kiosk-schema>

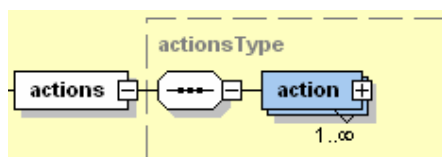
type **pin:contentType**

children **contentId actions totalPrice**

description This element contains all information about the contents of a chart. The fields are:

- **contentId** unique id of the content in the kiosk system
- **totalPrice** the total price of the acquired actions on the content

diagram

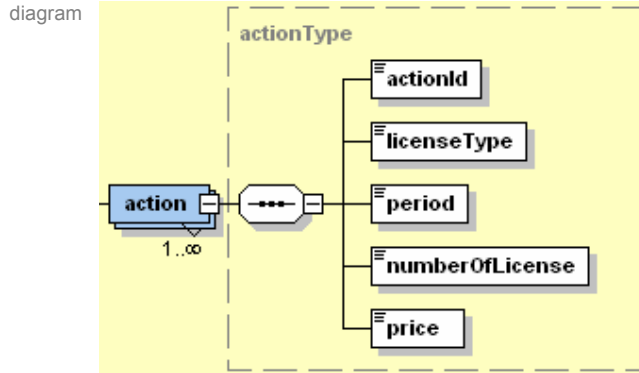


namespace <http://www.AXMEDIS.org/kiosk-schema>

type **pin:actionsType**

children **action**

description This element contains all information about the actions related to the content in the chart.



namespace <http://www.AXMEDIS.org/kiosk-schema>

type **pin:actionType**

children **ActionId licenseType period numberOfLicense price**

description This element contains all information about an action related to the content in the chart.. The fields are:

- **actionId** unique id of the action in the kiosk system
- **licenseType** license type related to the content
- **period** period of validity of the action
- **numberOfLicense** number of license to acquire
- **price** total price of the acquired action on the content

6.6.11 Billing log object

The following table describes the billing log related data structure as devised within the kiosk

Attribute	Type	Description
billingId	String	The ID of the billing operation to be logged.
userId	String	The ID of the user
operationDate	String	Date of actual billing operation
contentPurchaseList	ArrayList	The list of couple: object, operation type...

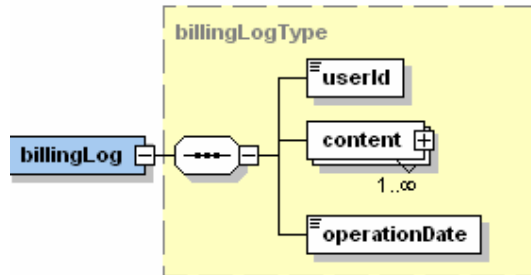
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="billingLog" type="billingLogType"/>
  <xs:complexType name="billingLogType">
    <xs:sequence>
      <xs:element name="userId" type="xs:string"/>
      <xs:element name="content" type="contentType" maxOccurs="unbounded"/>
      <xs:element name="operationDate" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="contentType">
    <xs:sequence>
      <xs:element name="contentId" type="xs:string"/>
      <xs:element name="actions" type="actionsType"/>
      <xs:element name="totalPrice" type="xs:float"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

```

<xs:complexType name="actionsType">
  <xs:sequence>
    <xs:element name="action" type="actionType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="actionType">
  <xs:sequence>
    <xs:element name="actionId" type="xs:string"/>
    <xs:element name="period" type="xs:string"/>
    <xs:element name="price" type="xs:float"/>
    <xs:element name="licenseReference" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

diagram



namespace <http://www.AXMEDIS.org/kiosk-schema>

type **pin:billingLogType**

children **userId content operationDate**

description This element is the root element containing information of the billing logt object.The fields are:

- **userId** unique id of the user in the kiosk system
- **content** refer to the previous contentType definition (page 78)
- **operationDate** date of the operation

7 AXMEDIS Portal (EXITECH)

7.1 Overview

As stated in the Annex I, the AXMEDIS portal is the general service front-end to users, playing a fundamental role in the technical infrastructure and in the implementation of synergies among distribution channels for classical music digital content.

The web portal will be the main front-end for the customers (those that can be interested to join the AXMEDIS framework). It will be designed and organized to attract potential users through an appealing and efficient user interface. The usability characteristics of the web portal will be well studied so that the visual aspect of the web pages can be easy to use and efficient. During the development of web pages, the different market segments and user profiles will be taken under consideration so that the appearance of the web portal is interesting for frequent web users and easy and attractive for people who are less familiar with this media. This is very important since the project, which is mainly based on editorial production and distribution, must take care about the brand and the image on the market. So logo, icons, colours, shapes and sound have to be appealing and unique in such a way that they can be a strong element in the marketing and dissemination phase.

The **main roles of the AXMEDIS portal** are the following:

- Providing technical information to contractors, affiliated partners, user group members etc.: software components, test cases, discussion lists, documents, guidelines, etc.;
- Providing information and access point to who is interested to join the AXMEDIS initiative
- Providing administrative information for the contractors;
- Providing support for research institutions interested to join AXMEDIS and to contribute to its development,;
- Disseminating the information of AXMEDIS and providing some demonstrator;
- Providing support to companies and institutions that will be involved in the take up actions;

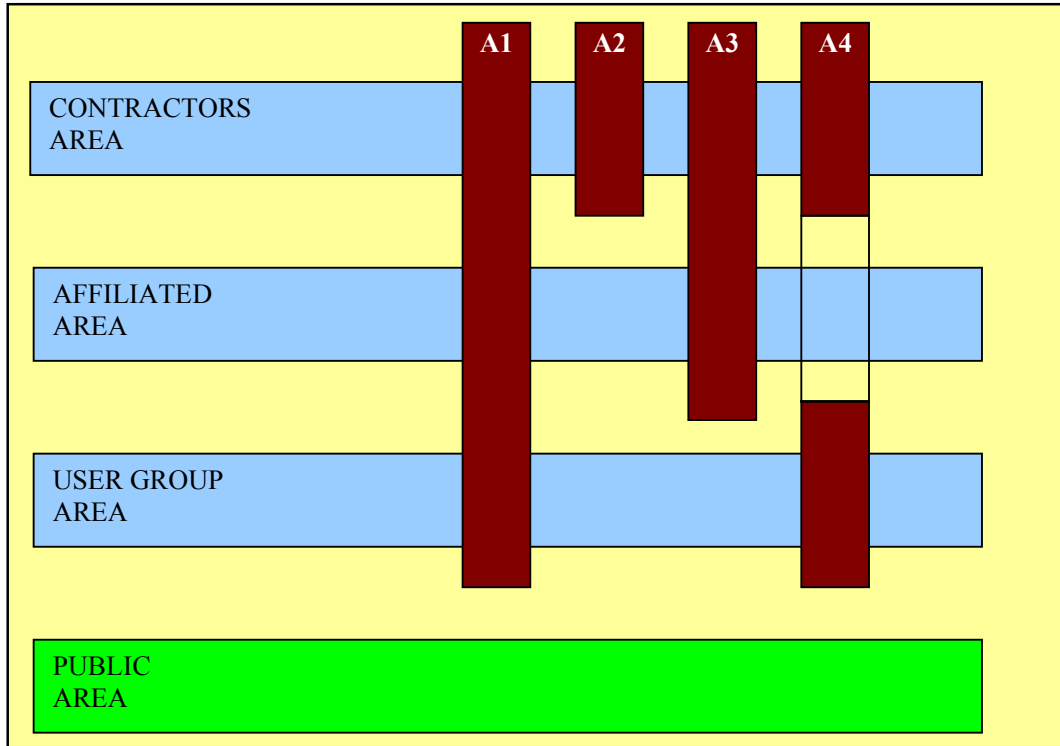
The Annex I notes that: “the AXMEDIS Portal is NOT a super database of the contents. The content is stored into the databases of the Content Providers and distributors”.

The AXMEDIS web site will be implemented using PHP modules and MySQL database on an Apache server. The web site will be compatible with any other web server and should be moved on a Microsoft IIS web server as well. The style sheet technology was used to guarantee a uniform model and style of the site layout. The entire code will be projected, implemented, and tested by Exitech. The main reason for doing so is that we like to have a very high level of site personalization. Anyway if one or more commercial or third party tool will be found guaranteeing all the web site requirements in a better manner it may substitute our code.

The AXMEDIS web site will be divided in four main areas:

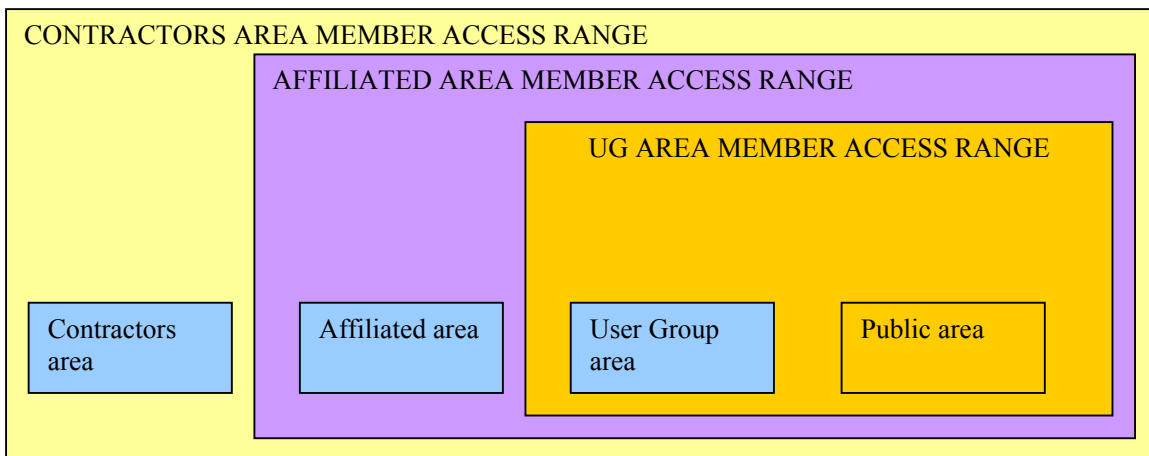
- (i) public area,
- (ii) user group area, (private)
- (iii) affiliated area, (private)
- (iv) contractors. (private)

The portal structure will be completed with several activities. The activity is a transversal structure, created and managed from the contractors area. An activity will contains generic hypertext, a set of documents and a set of access rights. One activity can be viewed in all the private areas but not all the area subscribers will grant the right to access it.



The PORTAL activities can be visible in one (A2) or more (A1, A3, A4) private areas, but not into the public side of the site.

The web page access will be free for the public area and controlled for the other. A contractor will have access to the whole web site; the affiliated area member cannot access the contractors area and the UG area members cannot access the other private areas.



The web site will manage the following type of access:

- Anonymous
- User Group member
- Affiliated area member
- Contractors area member
- Public area administrator
- User group area administrator
- Affiliated area administrator
- Contractors area administrator
- Contractors activity administrator
- Webmaster

The following services were identified for the web site:

- Information on AXMEDIS (multilingual – 6 languages)
- Information on UG activity
- Information on Affiliated activity
- Information on Contractors activity
- Mailing list for discussion of problems
- Upload and download of documents
- Contractors and affiliated web pages
- Event and fairs announcements and registration
- Test cases database
- Trial applications and tests results
- Identification of major sources of information
- WEB site search engine (html, .pdf, .ps, .doc file format and database content)
- WAI accessibility for some selected parts of the web site
- Newsletter service and newsletter archive browsing
- News service, top ten news, top ten downloaded documents, etc.
- Web site statistics, documents, logs, etc.
- Collaborative area
- CVS availability
- Opening and closing operation for activities
-

The activities will be organized as described in follow:

ACTIVITY: test admin Coordinator(s): mati (a.mati@exitech.fi.it)	Services Download doc Upload doc News	Contents + main html node 1 c outside html node 2  - empty node 1  subnode A  subnode B 
---	---	--

The activity page will display the activity services (the same for all the activities) and the activity contents tree.

Into the tree there are 3 types of elements:

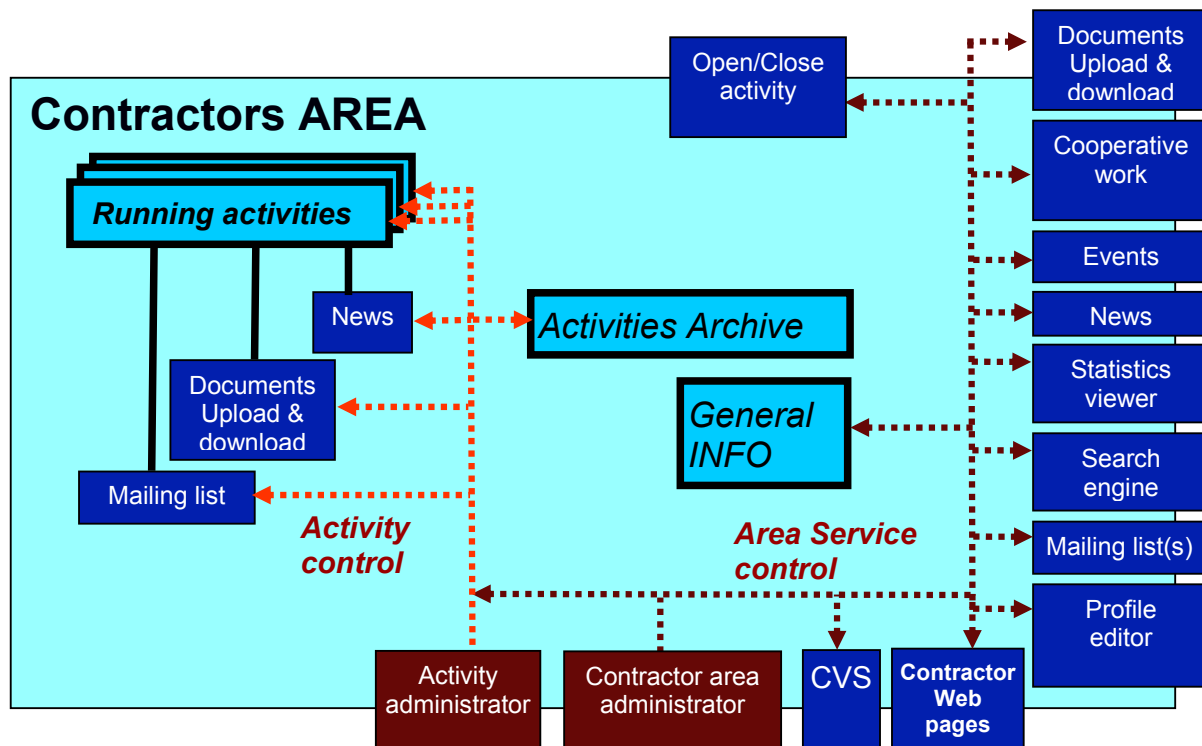
- Html node – a normal hypertext page
- Empty node – just a kind of folder name, does not contain any data but can be used for grouping other elements

- External node – a link to a file (images, documents, etc) – by clicking a new browser window will be opened

7.2 CONTRACTORS area, (private access):

The contractors area will be accessible only to the contractors. In order to become a contractors area member the user has to contact the webmaster and after the project coordinator confirmation the access will be activated. The main characteristics of the contractors' area are:

- A structure divided in ACTIVITIES with news and documents for each activity. New activities can be added dynamically by one contractors area administrator.
- A contractor service access point
- Access for the contractors area administrators
- An access for the activity administrators
- An access to Area for cooperative work on documents, (like BSCW or NUXEO)
- Access to the AXMEDIS CVS for cooperative work on sources of the AXMEDIS framework
- Hypertext content



Each running activity will provide

- Documents upload tool, automatically will be accessible for all the CONTRACTORS in the global list. For each upload a message is automatically added and displayed into the activity news and posted on the mailing list indicated by the responsible, see below.
- List of the available ACTIVITY documents by alphabetic or date ordering with the evidence in which ACTIVITY has been posted
- Download of document of the activity
- Specific blobs posted
- A responsible that has to administrate the documents cataloguing, has to maintain the main HTML page of the activity and to state which mailing list(s) has to receive an email for each news or document posted.
- A main html activity page with
 - last posted public documents
 - most requested public documents

The contractors main page will also contains:

- Messages posted on the internal NEWS/BLOGS
- Access to the Contractors web pages, only internally accessible, but granting the possibility to perform autonomous changes by each individual contractor
 - One home page for each contractor for names, emails, fax, of people involved, roles, link to their web pages and activities, any links to relevant document into the AXMEDIS database, see below, etc.
- Access to a page for adding new events in the list
- Profile changing tool,
- Access to send a email to the mailing list: CONTRACTOR, AFFILIATED, PARTICIPANTS, USER GROUPS
- List of the available area documents by alphabetic or date ordering with the evidence in which ACTIVITY has been posted
- Search in the database of documents and html pages
- Access to the list of ACTIVITIES Archive
- Access to the list of Running ACTIVITIES

For the CONTRACTOR AREA administrator more tools will be available:

- Creation of a new ACTIVITY and related framework
- Close an Activity (pass it from running Activities to the Activities Archive)
- Control for polishing lists and the above services
- Monitoring user access and producing statistics
- Change the access to document granting the access of a document to CONTRACTORS, AFFILIATED, USER GROUP or PUBLIC. In those cases the related list of document is updated.
- See the list of registered people at the several levels
- Allow to a Contractors area member to administrate an activity (create a Contractors activity administrator)
- Change the activities accessibility

The Activity administrator will access the tool for:

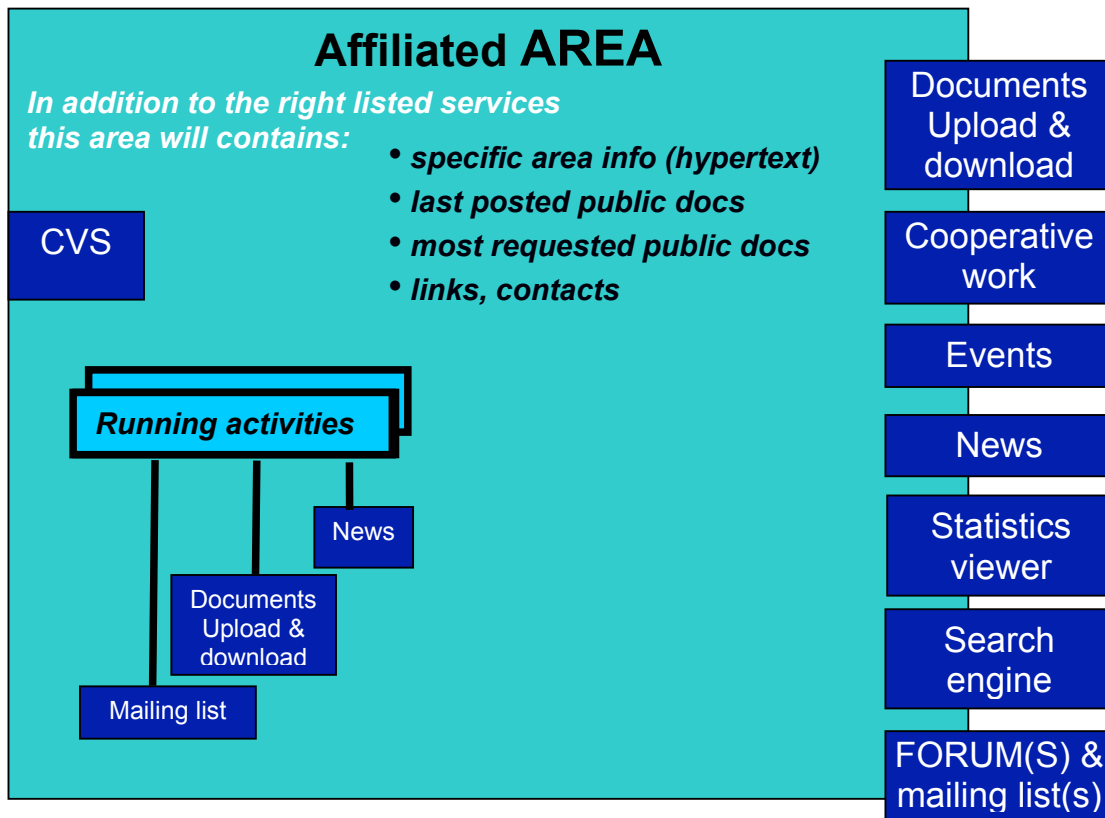
- Manage the activity documents (organise, delete or move documents)
- Maintain the activity html pages
- View the activity statistics (web page access, activity mailing lists)
- Close the Activity (pass it from running Activities to the Activities Archive)

7.3 AFFILIATED Area (private)

The access to the affiliated area will be grant to the affiliated area members and to the contractors area member.

For the affiliated area the following services will be provided:

- Messages to be viewed on the internal Affiliated NEWS
- Access to part of AXMEDIS CVS
- Access to the Affiliated web pages, only internally accessible, but granting the possibility to perform autonomous changes by each individual affiliated
(One home page for each affiliated for names, emails, fax, of people involved, roles, link to their web pages and activities, any links to relevant document into the AXMEDIS database, etc.)
- Change of profile, non change of mailing list subscription
- Affiliated mailing list
- Documents upload/download
- Search in the database of documents and html pages (not into the contractors area)
- Area for cooperative work on documents

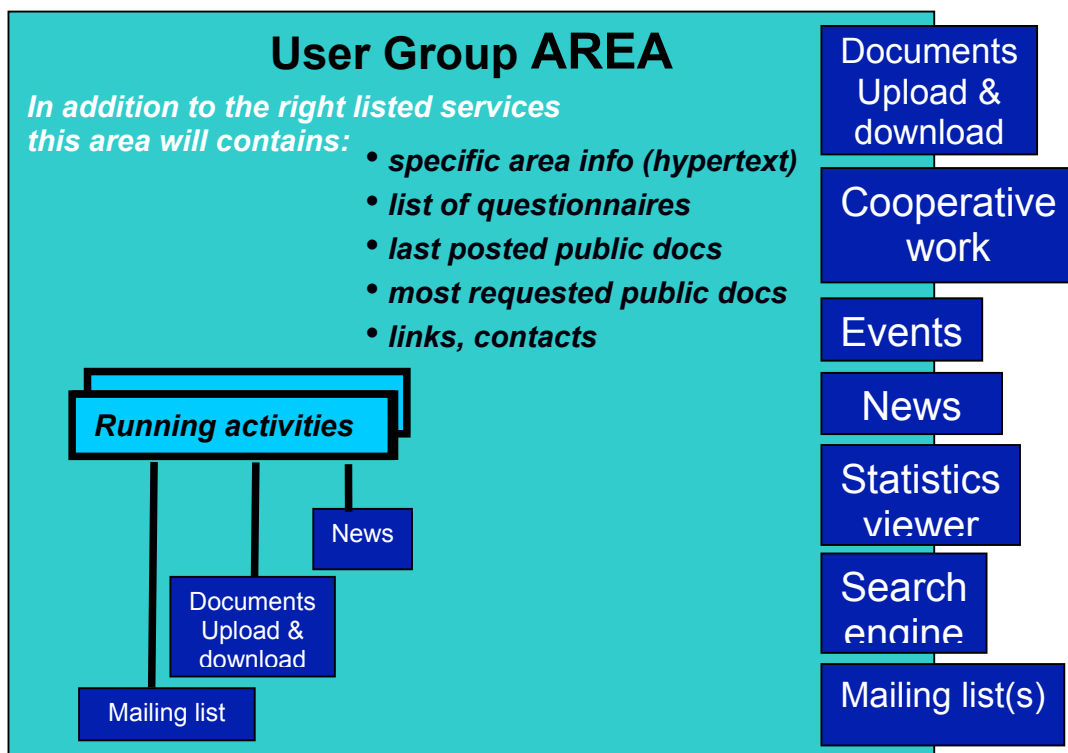


- For the affiliated area administrators more services will be available:
 - Control for polishing lists and the above services
 - Monitoring access and producing statistics

- Change the access to document granting the access of a document to CONTRACTORS, AFFILIATED, USER GROUP or PUBLIC. In those cases the related list of document is updated.
- See the list of registered people at the several levels

7.4 USERGROUP area (private):

- For the UG area the following services can be included:
 - a specific mailing lists
 - Upload and download of documents
 - The access to the contractors and affiliated documents will be as decided during upload by the Contractors or Affiliated
 - Area for cooperative work on documents
 - A questionnaires repository
 - Search on all documents, but access to only those marked as USERGROUP or PUBLIC

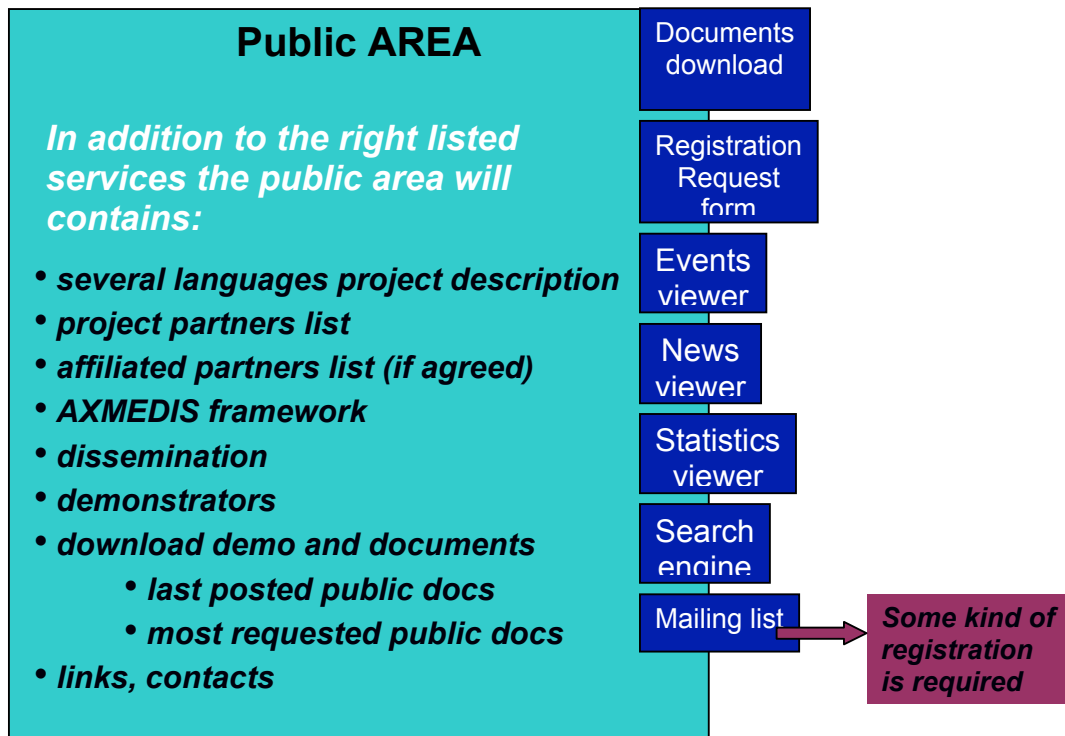


- As usual for the UG area administrators more services have to be available:
 - Control for polishing lists and the above services
 - Monitoring access and producing statistics
 - Change the access to document granting the access of a document to CONTRACTORS, AFFILIATED, USER GROUP or PUBLIC. In those cases the related list of document is updated.
 - See the list of registered people at the several levels

7.5 PUBLIC area:

The public area is also the home page for the whole portal. The page organisation is very important in order to help the user to quickly understand the web site structure.

- Menu on the left
 - Description
 - general information of the project
 - AXMEDIS Framework
 - Architecture, purposes
 - Dissemination
 - Press release, flyer, project presentations
 - Demonstrations
 - Some download possible, link to demos
 - Downloads
 - For documents and tools
 - Events and conferences
 - Contacts
 - contacts and main people
 - Partners
 - list of the Contractors with their logos
 - Affiliated partners (those that have signed during the registration to be visible)
 - Registration rules
 - How to become affiliated, real access to the AXMEDIS information
 - How to become participants, simple registration on the newsletter
 - Links
 - Statistics



- Services visible on the first page
 - Immediate description in several languages
 - news
 - last posted public documents
 - most requested public documents
 - registration to the mailing list (please refer to the “Newsletter” paragraph in follow)
 - mailing lists for all the affiliated
 - request of information, send an email to me
 - Search in PUBLIC documents and web pages

The public available pages should be HTML compliant.

7.6 PORTAL administration area:

The administration area provide all the services needed for the remote web portal administration. The services are available following the user rights. In this document all the service available in the administrative area are briefly specified.

- Access definition:

Access type	Access range		
admin	faq	Edit	Delete
admin	general	Edit	Delete
admin	mlists	Edit	Delete
admin	news	Edit	Delete
admin	newsletter	Edit	Delete
admin	partners	Edit	Delete
admin	statistics	Edit	Delete
admin	users	Edit	Delete
admin	webpages	Edit	Delete
admin events	general	Edit	Delete
admin events	partial	Edit	Delete
coordinator activity	Assessment	Edit	Delete
coordinator activity	AXMEDIS Specification	Edit	Delete
coordinator activity	Call for Take UP	Edit	Delete

The project administrator can define how many access types he need and the relative access range.

- Document organisation.

Used for defining the folder where the documents of the several activities are showed.

AXMEDIS Specification/

New group	
generic (0)	New subgroup Edit Delete
AXMEDIS Framework (21)	New subgroup Edit
State of art (4)	New subgroup Edit
Subcontract-crawler (2)	New subgroup Edit

The single documents can be renamed, moved, replaced or removed from the selected folder.

AXMEDIS Specification/State of art/

IPR Help Desk of EC and DRM (8_digitalrightsmanagementsystems[0000006090_00].pdf) Nov. 2, 2004 paolo	Edit In groups Replace Visibility Delete
Content Reference Forum Documentation (crf.zip) Oct. 12, 2004 paolo	Edit In groups Replace Visibility Delete
ACEMEDIA documentation (acemedia.zip) Oct. 12, 2004 paolo	Edit In groups Replace Visibility Delete
an communication of NESI, old article on modeling and management for development (s4050-nesi.pdf) Sep. 28, 2004 paolo	Edit In groups Replace Visibility Delete

■ Activity list

All the portal activities can be viewed and managed by the allowed administrator accounts:

activity	creation date	coordinator(s)	mailing list	state	closing date			
Assessment	Sep. 1, 2004	e.melchior	axmedis@axmedis.org	in progress		Edit	Coordinator	Mlist
AXMEDIS Specification	Sep. 1, 2004	paolo	axmedis@axmedis.org	in progress	Dec. 31, 2008	Edit	Coordinator	Mlist
Call for Take UP	Oct. 28, 2004	c.giachetti	axmedis@axmedis.org	in progress	Jun. 15, 2006	Edit	Coordinator	Mlist
Content Specification	Oct. 28, 2004	d.fuschi	axmedis@axmedis.org	in progress	Dec. 31, 2008	Edit	Coordinator	Mlist

Also new activities can be created.

■ News, newsletters and events

News, newsletters and events can be inserted, edited or removed:

insert new news

2004-12-14 e.melchior	New doc: Files with target values
<input type="button" value="Edit"/> <input type="button" value="Delete"/>	axmedis-de-10-4-1-app-v13.zip
2004-12-14 e.melchior	New doc: AXMEDIS-DE-10-4-1-Assessment-and-Evaluation-v1-3.ZIP
<input type="button" value="Edit"/> <input type="button" value="Delete"/>	axmedis-de-10-4-1-assessment-and-evaluation-v1-3.zip
2004-12-14 e.melchior	New doc:
<input type="button" value="Edit"/> <input type="button" value="Delete"/>	axmedis-de-10-4-1-assessment-and-evaluation-v1-1.zip
2004-12-14 e.melchior	New doc: AXMEDIS-DE-10-4-1-Assessment-and-Evaluation-v1-2.ZIP

■ Content

All the web site contents can be remotely edited by the administrators:

title:

description:

content: [view default styles](#)

```
<br>
<table border="0" bgcolor="Silver" align=center>
<tr>
    <td align="center"><font color="Maroon"><b>For the full document list of
this area please see the download documents link on the activity menu on the top
    </b></font></td>
</tr>
</table>
<br>
<span class="title1">AXMEDIS Consortium Reference People: </span><br>
<?php
```

update date:

Also the activity contents can be edited:

Into the administration area the user has to choose the activity/contents link on the left menu, choose an activity (more than one should be displayed if you are administrator or more activities) and you will see the tree elements:

accessi

[vedi/modifica](#)

[aggiungi nuovo](#)

activity

[documents](#)

[activity](#)

[news](#)

[new activity](#)

[contents](#)

aree

[area](#)

contents

[contents](#)

documenti

[documenti](#)

[edit file](#)

Contents [\[help\]](#)

New content				
main	New content	Type	Edit	
sub main inside	New content	Type	Edit	Delete
html node 1	New content	Type	Edit	Delete
c outside	New content	Type	Upload file	Edit Delete
html node 2	New content	Type	Edit	Delete
empty node 1	New content	Type		
subnode A	New content	Type	Edit	Delete
subnode B	New content	Type	Edit	Delete

It is possible to add new contents at the several tree levels, remove nodes without children or edit the non empty nodes.

The upper “new content” button will add nodes at the root level.

When the Edit button is pushed the content can be modified by using the following form:

[Contents](#) [\[help\]](#)

title:

description:

content: [view default styles](#)

```
<div align=\ "center\ ">
<br>
<span class=\ "title1\ ">AXMEDIS test admin main</span>
<br>
</div>
```

update date:

The description field is saved into the database but it is dedicated to future expansions.

Into the “content” field you have to insert the html code for the content.

The “update date” is saved into the database and it is used to display the “new” label near the node if the changes were performed in the last three days.

When an external node is added a document has to be uploaded. By pushing the node a new window will be displayed if the document has a mime type, otherwise the download page will be displayed.

■ Areas tool

For all the web areas is possible to rename, change the access:

Acronym:

Title:

Description:

All user can subscribe to: ☒ yes ☐ no

Visible for:

☐ all

☒ registered users with access:

registered user — general ☐

membership area — Contractors ☒

membership area — Affiliated ☐

membership area — User Group ☐

■ The mailing lists can be created or edited:

[create new mlist](#)

title	prefix	email	
General	[AXMEDIS]	axmedis@axmedis.org	Edit
Administration	[AXADMIN]	axadmin@axmedis.org	Edit
Dissemination	[AXDISS]	axdiss@axmedis.org	Edit
Exploitation	[AXEXPLOIT]	axexploit@axmedis.org	Edit
Usergroup	[AXUG]	axug@axmedis.org	Edit

- The partners list can be viewed and managed:

Partners [\[help\]](#)

partner	country	profile manager	webpage on Axmedis			
ACIT	Germany	e.melchior	ASSIGNED	Edit	profile manager	webpage
AFI	Italy	f.giangrandi	ASSIGNED	Edit	profile manager	webpage
ANSC	Italy	r.grisley	ASSIGNED	Edit	profile manager	webpage
COMVERSE	Israel	a.ashkinazi	ASSIGNED	Edit	profile manager	webpage
CPR	ITALY	g.cesaretti	ASSIGNED	Edit	profile manager	webpage
CRS4	ITALY	d.carboni	ASSIGNED	Edit	profile manager	webpage
DIPITA	ITALY	m.fabbri	ASSIGNED	Edit	profile manager	webpage
DSI-DISIT	ITALY	paolo	ASSIGNED	Edit	profile manager	webpage
EPFL	Switzerland	g.zoia	ASSIGNED	Edit	profile manager	webpage
EUTELSAT	France	o.pulvirenti	ASSIGNED	Edit	profile manager	webpage
EXITECH	ITALY	marlus	ASSIGNED	Edit	profile manager	webpage
FHGIGD	GERMANY	m.schmucker	ASSIGNED	Edit	profile manager	webpage
FUPF	SPAIN	j.prados	ASSIGNED	Edit	profile manager	webpage
HP	Italy	c.marangoni	ASSIGNED	Edit	profile manager	webpage
ILABS	Italy	d.fuschi	ASSIGNED	Edit	profile manager	webpage
IRC	UK	a.badii	ASSIGNED	Edit	profile manager	webpage
OD2	UK	NOT ASSIGNED	ASSIGNED	Edit	profile manager	webpage
SEJER	FRANCE	c.stehlin	ASSIGNED	Edit	profile manager	webpage
TISCALI	Italy	NOT ASSIGNED	ASSIGNED	Edit	profile manager	webpage
UNIVLEEDS	UNITED KINGDOM	k.ng	ASSIGNED	Edit	profile manager	webpage
XIM	UNITED KINGDOM	l.pearce	ASSIGNED	Edit	profile manager	webpage

- Statistics

Several statistics can be viewed from the administration area. The number of accesses are registered for each user, the accesses can be viewed also grouped by company, viewed for each area, country, the number of uploaded documents for each registered user, the number of news inserted for each user and the number of accesses to each activity.

TISCALI	68
UNIVLEEDS	368
XIM	224
Total:	5817

Access from: Users View					
Users	Affiliation	Con	Aff	Use	Pub
User Name 1	Company 1	3	0	0	0
User Name 2	EXITECH	319	22	22	67
User Name3	EXITECH	122	2	2	16

■ Users

The portal administration can use a set of tools for user list, user access creation, assign, edit or delete user access, activate or deactivate user fields (form to be listed into the user profile). A link for the list of the online user is also available.

7.7 Main service characteristics

In the follow the services characteristics are shown. For each service the main features are listed and explained. The showed figures was inserted for better indicate the scope of the service but there are indicative (not mandatory) regarding the final web site release.

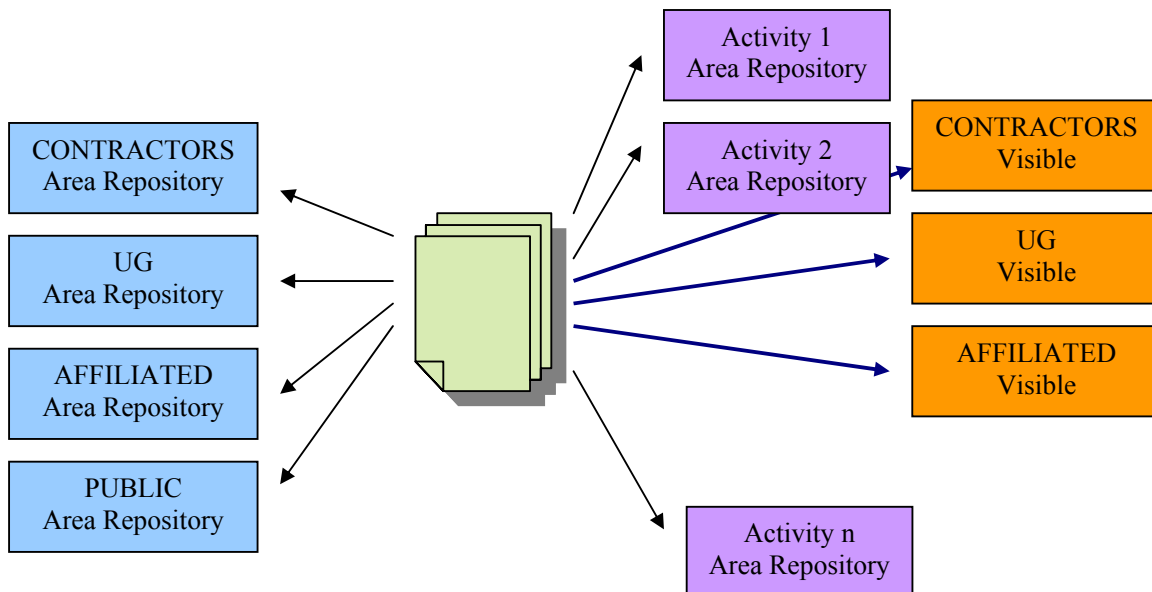
The administrative services was described into the dedicated paragraph “PORTAL administration area”

7.7.1 Documents Organisation

To be revised I don't know if the description is clear enough

To the documents uploaded on the web portal corresponds a unique file.

The documents can be viewed in several virtual repositories: one for each area and one for each activity.



Between the repositories acts also the activities visibility rules. That means that if a document is visible in the activity “k” but the K activity is not visible for the UG area it will be not visible in the UG activities.

The orange permission act as a second level visibility switch and allow to set documents as visible for the activity but not visible for an area also if the rest of activity remain visible.

Let consider for example the document “test.pdf” visible for the “activity 1” repository and the “activity 1” is visible for all the private areas. By setting the document UG visibility it can be not visible for the UG area.

If the document has to be viewed by the anonymous user it have to be set as visible into the PUBLIC Area Repository since no activities can be viewed in the public area.

There is no user available tool for viewing the document visibility. Into the download documents window and also into the full documents list, no information about the document visibility into other areas or activities are available.

7.7.2 Documents Download

For downloading a document the user has to use the download item into the services menu. Both the four areas documents and the activities documents index will be displayed. It is possible to browse into the structure and view/download any document.

Meetings [doc structure \(view full list\)](#)

[Activities/Meetings/generic/](#)

doc	date [desc]	uploaded by
axmedis-scenarios-wf-v3-irc-hp-xim-dsi-v4-25-11-04.ppt	Nov. 25, 2004	a.badii
axmedis-usergroupsession3-v-1-1.zip	Nov. 22, 2004	s.llorente
axmedis-call-for-papers-v0-2-univleeds.zip	Nov. 20, 2004	k.ng

7.7.3 Documents Upload

Any user who has an account on the AXMEDIS web site may upload a document in his own area or into the other areas like shown into the next table:

User is Member of:	Upload available into the
Contractors	Contractors area, activities, Affiliated area, User group area, Public area
Affiliated	Affiliated area, User group area, Public area
UG	User group area, Public area

The anonymous user cannot upload documents.

For accessing to the uploading service the user has to access the web site, perform the login and choose the upload doc into the Services menu.

Into the next window browse for the file to be uploaded (name without space), insert the document short description (to be shown into the top ten or other short lists), then the complete description. Choose the areas where the document has to be viewed and the relative destination group. The same document may be uploaded into all the four areas (no copies will be created but only the several accesses will be created).

Uploading documents in a area

It is enough to choose the “upload doc” in the service menu. In the opened window the full file name, the short and the long description have to be given. The document will gain visibility in the areas selected by the user.

The window will be similar with the one showed into the next figure.

Documents upload

[back to Contractors](#) [back to home](#)

Upload file (name without space):

Short description:

Description:

Select area:

Contractors ☒

Affiliated ☐

User Group ☐

Public area ☐

Select destination group:

generic

generic

generic

generic

Uploading documents in an activity

The activities documents can be uploaded by accessing any activity into the Contractors, Affiliated or UG Area. Use the upload document link for having the next window.

Upload doc

Upload file (name without space):

Short description:

Description:

Select activity:

Requirement analysis ☐

AXMEDIS Specification ☐

Management ☒

Meetings ☐

Assessment ☐

Dissemination ☐

Tech-watch ☐

test admin ☐

Call for Take UP ☐

Test Cases ☐

UserGroup ☐

Content Specification ☐

Research Guidelines and Spec ☐

Exploitation ☐

Select destination group:

generic

generic

generic

generic

generic

generic

generic

generic

generic

generic

generic

generic

generic

generic

Document visible for:
(each tick automatically means tick for every row atop)

Contractors ☒

Affiliated ☐

User Group ☐

Public area ☒

select destination group: generic

By choosing the right item in the right combo boxes the documents will be showed in the selected folder.

7.7.4 Web search engine

A service offered by the AXMEDIS web site is the search engine. The engine will be chosen in order to fulfill requirements like the capability to perform database search and to allow some protection on the result (in case the user is not registered), possibility of personalization, etc.

Search engine requirements:

- Perform search into the html file
- Perform search into the txt file
- No search into the php code
- Allows to eliminate some directory from the search path
- Allows search into a part of the database (messages)

7.7.5 Mailing lists

Several mailing lists will be set up at different levels. Any web site area will have one or more mailing lists. The members of contractors area, according to the CA, will be automatically added to one or more mailing lists. The members of the other areas will accept an agreement when the subscription will be done.

Some specific rules will be implemented:

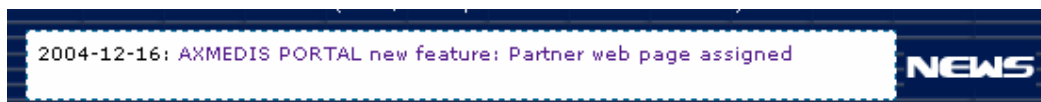
- Only subscribers can send emails to the mailing list
- Some antivirus scan will be perform on the messages
- Some controls will be performed to eliminate vacancy message propagation
- A user may be subscribed in more than one mailing list and the subscription can be controlled by the web administrator and the mailing list administrator
- The email addresses for members of contractor area mailing lists will be loaded from the database so the user can change at any time his/her address by accessing the edit profile service
- The mailing lists will allow attaches
- When a document is posted a mail message will be sent to the related mailing lists.
- When a document is posted a news message will be added to the related area or activity.

New mailing lists can be created at any moment with little system modification.

Any activity will be associated to one or more mailing lists where the activity news will be posted.

7.7.6 News

The News service allow to the area administrators to insert short text messages that will be displayed on the top of the AXMEDIS portal page:



Only the last five news are shown there, but a news archive is available.

If the news is older than 60 days it is no more displayed. The news area should remain empty if no news was inserted into the last 60 days.

From the administrator area it is possible to insert new news and give it the necessary visibility.

insert new news

2004-10-07

marius

CEATEC 2004 opened

CEATEC Japan 2004, one of the biggest exhibitions in Asia, opened Oct. 5 at the Makuhari Messe convention center outside Tokyo Sony unveils the new Vaio type X home server, which can store 1 terabyte of data, enough to record six TV channels for a week with MPEG-2 compression. It will be available next month for about \$4,700.

Edit

Delete

2004-10-07

marius

AXMEDIS kickoff meeting - October 5-9, 2004

The AXMEDIS kickoff meeting is planned for October 5-9, 2004 in Florence

Edit

Delete

2004-09-10

marius

AXMEDIS web site activated

first draft of the AXMEDIS web site activated

Edit

Delete

insert new news

In the text it's possible insert link: `link name`

Title: AXMEDIS PORTAL:

Text: Partner web page assigned - see the partners page

News visible for:

☐ all

☒ registered user with access:

membership area — Contractors ☒

membership area — Affiliated ☐

membership area — User Group ☐

Insert

Quit

Also for the each activity a news service was planned. An activity administrator can insert text news displayed on the news part of the activity.

The upload of a document became automatically news in that activity and is displayed on the top of the activity main page:

Activity: Content Specification
Coordinator(s): d.fuschi (d.fuschi@giuntilabs.it)

last news:		
Nov. 20, 2004	New doc: AXMEDIS-DE8.2.1	axmedis-de8-2-1-contenttselectionguidelines-v0-6.zip
Nov. 20, 2004	New doc: AXMEDIS DE3.1.3	axmedis-de3-1-3-contentaspectspecification-v0-9.zip
Nov. 18, 2004	New doc: AFI contribution to DE 3.1.3.	axmedis-de3-1-3-contentaspectspecification-v0-8afi.doc
Nov. 5, 2004	New doc: AXMEDIS DE8-2-1	axmedis-de8-2-1-contenttselectionguidelines-v0-5.zip

For the full document list of this area please see the download documents link on the activity menu on the top

7.7.7 Newsletter

A newsletter service was planned. The administrator can send a message to one or more destinations choosed between:

- Contractors area subscribers
- UG area subscribers
- Affiliated area subscribers
- Newsletter subscribers

An anonymous user can subscribe to the newsletter by typing the email into the newsletter registration form. Also the unsubscription can be performed from the same form.

It is possible to view all the posted newsletter or to perform filters and change the newsletter list order.

AXMEDIS Newsletter

Name:	<input type="text"/>	<input type="button" value="Subscribe"/>
Email:	<input type="text"/>	<input type="button" value="Unsubscribe"/>

Archive

From:	<input type="text" value="January"/>	<input type="text" value="2004"/>
To:	<input type="text" value="December"/>	<input type="text" value="2004"/>
Order by:	<input type="text" value="date"/>	<input type="radio"/> asc <input checked="" type="radio"/> desc
<input type="button" value="View"/>		

7.7.8 User Profile

The user profile allows the web site registered users to update the personal information.

The only field that cannot be changed by the user is the account. It can be changed only by the webmaster.

The profile will collect several information like:

- First name
- Second name
- Country
- Email

- Website
- Title
- Affiliation
- Type of affiliation
- Type of role
- Area of role
- Contact information (phone, fax, ecc.)

The user may choose if the data will be visible only to the web administration, to the registered user or the data will be public visible.

At this date no public data export is planned so the public visibility is considered as the one for registered users.

The Newsletter subscription or un-subscription form is available.

		visibility:
nickname:	marius	public
New password:	<input type="text"/>	nobody
Repeat new password:	<input type="text"/>	nobody
first name:	Marius	public <input type="button" value="v"/>
last name:	Spinu	public <input type="button" value="v"/>
contry:	ITALY <input type="button" value="v"/>	public <input type="button" value="v"/>
telephone(s):	055-470958	web administration <input type="button" value="v"/>
fax:	055-470958	public <input type="button" value="v"/>
email:	ms@exitech.it *	public <input type="button" value="v"/>
web site:	http://www.exitech.it	public <input type="button" value="v"/>
title:	eng. Ph.D.	public <input type="button" value="v"/>
affiliation:	EXITECH *	public <input type="button" value="v"/>
type of affiliation:	Information Technology providers, industry <input type="button" value="v"/> *	public <input type="button" value="v"/>
type of role:	technical <input type="button" value="v"/>	public <input type="button" value="v"/>
area of role:	(AXMEDIS web responsible, data	public <input type="button" value="v"/>
expertise keywords:	watermark	public <input type="button" value="v"/>
major interest about the Axmedis:	DRM	public <input type="button" value="v"/>
note:	<input type="text"/>	public <input type="button" value="v"/>

Newsletter registration:

☒ yes ☐ no

7.7.9 Contractors and Affiliated web pages

The members of the contractors area and the members of the affiliated area will have the possibility to fill a web page on the AXMEDIS web site.

The link (partner profile) will be available into the services menu and the user will have the possibility to change:

- the company web site
- the country
- the user assigned to the partner profile maintenance
- the company presentation page

Partners Edit profile

The information are visible on the list <http://www.axmedis.org/partners.php>
after short description the link '[...more]' open the webpage (if in the edit webpage form is set active).

partner:	<input type="text" value="EXITECH"/>
short description:	<input type="text" value="EXITECH srl"/>
country:	<input type="text" value="ITALY"/>
website:	<input type="text" value="http://www.exitech.it"/>
profile manager:	<input type="text" value="maris"/>

webpage on Axmedis:

The web pages are generated one for each affiliation (company). Any company has to define a person who will administer the web page.

Information like company contacts, overview on the activity, description of products can be inserted into these pages.

The pages will be activated on request.

The web pages will have a fixed structure and the user will upload some pieces of text and a number of images.

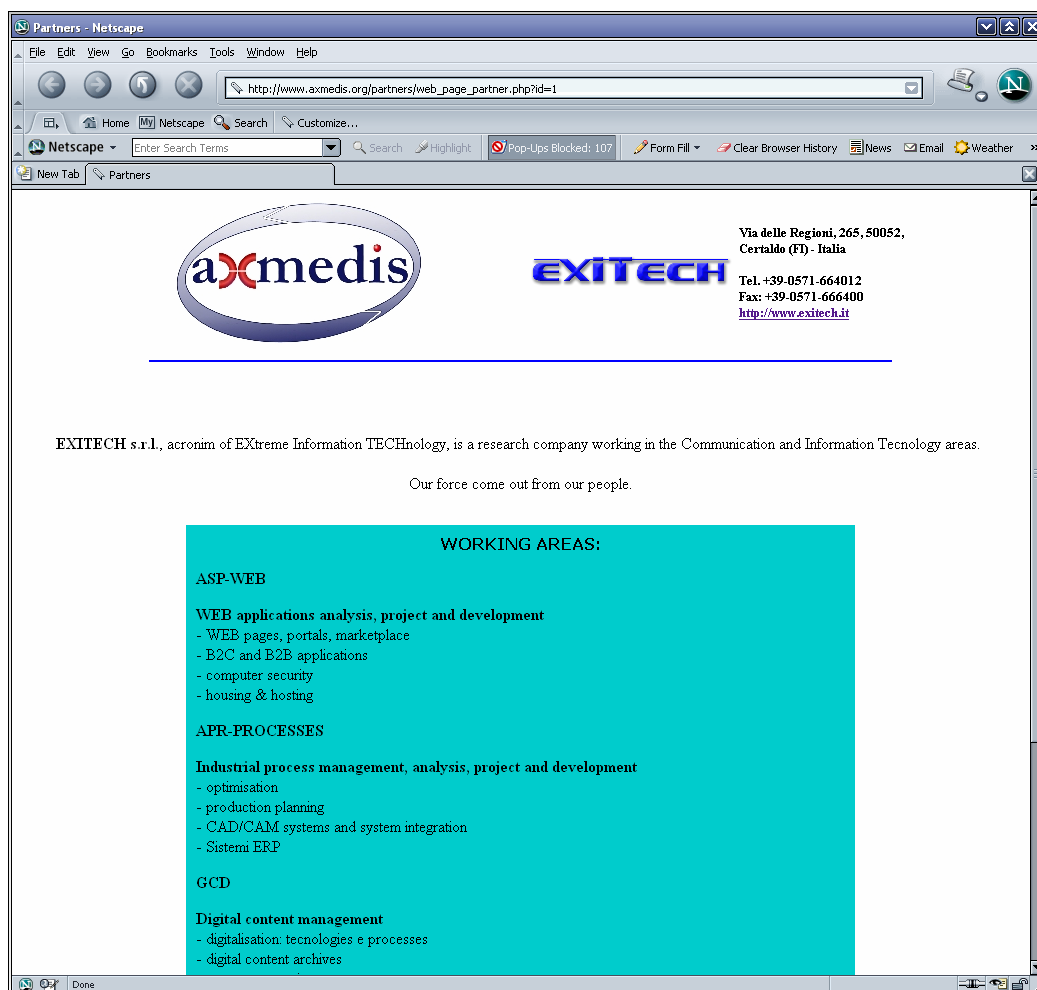
Web page structure:

logo	text 1
text 2	
image 1	
text 3	

The text can contain HTRM tags for page formatting. Since the access is controlled security controls on the uploaded text will be quite soft. The information will be visible on (public) partners page (<http://www.AXMEDIS.org/partners.php>). After the short description the link '[...more]' open the webpage (if in the edit webpage form is set active).

EUTELSAT	EUTELSAT S.A.	France	
EXITECH	EXITECH srl [...more]	ITALY	
FHGIGD	FRAUNHOFER GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V. [...more]	GERMANY	

A possible result is shown in the next figure:



7.7.10 Events

The events service is available from all the web portal areas. When a new event is posted the destination has to be chosen.

The same events can be viewed in more than one portal area. Into the event insertion form an area selection tool will be available.

The events can be organised by types. New event types can be added if necessary. Some events type will be available into the first version of web portal:

- Event organised by AXMEDIS
- Event where AXMEDIS will be present with special workshops or stands
- Event where AXMEDIS expert will be present

Insert event in:

Contractors	<input type="checkbox"/>
event type:	AXMEDIS will be present with special workshops or stands at the following conferences and fairs ▼
Affiliated	<input type="checkbox"/>
event type:	▼
User Group	<input type="checkbox"/>
event type:	▼
Public area	<input type="checkbox"/>
event type:	AXMEDIS will be present with special workshops or stands at the following conferences and fairs ▼

7.7.11 CVS

For the CVS support see Part B.

7.7.12 Conference management

A special section will be dedicated to the AXMEDIS conferences management

As commonly used the conference pages will be Structured in two sections:

- Public section with reference to the:
 - Conference topics of interest
 - Call for papers
 - Conference program
 - Committee
 - Location, maps, tourist and accommodation information
 - Registration form
- Private Area reserved for:
 - Paper submission,
 - Technical committee,
 - Reviewers.

Cyberchair (<http://www.cyberchair.org>)

For managing the paper submission aspects a specific tool was identified and analysed.

Cyberchair (defined as an online paper submission and reviewing system) is a tool based on the paper of O. Nierstrasz, Identify the Champion: (<http://iamwww.unibe.ch/~oscar/champion/>), with which the Program Committee Members ('reviewers') can indicate their opinion about papers that were submitted by authors.

The program is written in Python and was already used for other conferences.

CyberChair has a great part of the capabilities needed for the conference administration:

- paper submission,
- paper assignments to reviewers,
- review submission and comparison,
- categorization of reviews,
- generating overviews of reviews,
- sending notifications
- generating proceedings.

The conferences main page will be available from the public area of the AXMEDIS web portal.