Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE9.6.4
Integrated prototype of content production and distribution to kiosks and new generation of PDAs

Version: 2.3
Date: 15/04/2007
Responsible: ILABS (d.fuschi@giuntilabs.it) (revised and approaved by the coordinator)

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: DE9.6.4 Contractual Date of Delivery: M30 Actual Date of Delivery: 15/04/2007 Title of Deliverable: Specification of content production and distribution to kiosks and local PDAs Work-Package contributing to the Deliverable: WP9.6 Task contributing to the Deliverable: WP9.6.1, WP2.1/2, WP31.1/3/4, WP8 Nature of the Deliverable: Report Author(s): ILABS

Abstract: the present document reports the detailed specifications for the development of the experimental test-bed for distribution towards kiosks and PDA **Keyword List:** kiosk, PDA, content, production, distribution

Table of Content

1 Executive Summary and Report Scope

The present document holds the detailed specification of the demonstrators focussed on production and distribution of content to kiosks and local PDA. Therefore the demonstrator is focussed on content production, protection, sharing, formatting, distribution considering the market segments and interest of the potential consumers and their needs, it will be validated and tested by using the content produced with WP8 and that will be available in the project digital archives. Final objective of the demonstrator is to validate project results and achievements in a real world application that may lead to the set up of some complete and ready to use AXMEDIS product and service. The document is structured in sections; a part from the index and summary the first relevant section aims at describing content production (basically the kiosk factory) and the second covers the distribution. References to used contents and link to glossaries are provided too.

1.1 Responsibilities

ILABS is responsible for the completion of the present document, yet other technical partners are responsible for providing contributions especially as the kiosk environment is primarily an integration one. Document sections responsible are reported in the section title within brackets. When a section, or subsection, has no explicit responsible is assumed that is in charge to the responsible of the previous one or is a general one to which all partners shall contribute.

2 Introduction

The starting point of the kiosk demonstrator is the kiosk factory. This latter represents an instance of the publishing value chain enriched with AXMEDIS solutions. In the following sections we will quickly recall the production structure especially from the point of view of the desktop/e-publishing so to make apparent the area of intervention and the modified process. The aim is to specify most of the work by difference. This approach has been adopted as it has proved to be the most suitable for inserting new technology in highly structured and rigid environment like the publishing one.

2.1 Content production in the publishing environment

Given the rather complex and varied set of content that is addressed in AXMEDIS and the specific structure of the kiosk demonstrator is worth starting this detailed specification with a brief introduction on the traditional value chain so to point out the affected area and related benefits. The content production chain follows well-codified and standardised process that can be described as follows:

<table>
<tr><th>Activity</th><th colspan="2">Task</th><th>Involved roles</th></tr>
<tr><td colspan="3">Idea (1)</td><td rowspan="3">Management
Authors
Chief Editor
Press Office</td></tr>
<tr><td colspan="3">Market survey (2)</td></tr>
<tr><td colspan="3">Title design (3)</td></tr>
<tr><td colspan="3">Go / No Go decision [1] (4)</td><td>Management</td></tr>
<tr><td>Research of (5)</td><td>Sources (6)
References (7)
Contacts (8)
Multimedia (9)
…
Similar titles (10)</td><td rowspan="2">IPR/© clearance</td><td>Management
Authors
Editorial board
Editorial staff
Press Office
Legal Department</td></tr>
<tr><td colspan="2">Draft acceptance [2]
(if positive the next step starts if not the previous is reiterated)</td><td>Management
Authors
Editorial board
Legal Department</td></tr>
</table>

[1] Such a decision is based on market data and production cost analysis to ensure the expected return on investment (ROI)
[2] At this step starts a parallel process for managing © and IPR clearance performed in an asynchronous parallel fashion.

<table>
<tr><th>Activity</th><th colspan="3">Task</th><th>Involved roles</th></tr>
<tr><td>Editing of (11)</td><td>Texts (12)
Notes (13)
Indexes (14)
…
Multimedia (15)
Captions (16)</td><td colspan="2" rowspan="3"></td><td>Authors
Chief Editor
Editorial board
Editorial staff
Instructional designer
Press Office
Legal Department</td></tr>
<tr><td colspan="2">Product final acceptance
(if positive the next step starts if not the previous is reiterated)</td><td>Authors
Chief Editor
Legal Department</td></tr>
<tr><td>Finalisation of (17)</td><td>Texts (18)
Notes (19)
Indexes (20)
…
Multimedia (21)
Captions (22)</td><td>Chief Editor
Editorial board
Editorial staff
Instructional designer
Production department
Press Office
Legal Department</td></tr>
<tr><td colspan="3">Formal authorisation to start production
(if positive the next step starts if not the previous is reiterated. In this case IPR/© clearance should have been completed if not process may be suspended / stopped / cancelled)</td><td>Management
Chief Editor
Editorial board
Legal Department</td></tr>
<tr><td rowspan="4">Production</td><td>Books & Magazines</td><td rowspan="6">Cost control</td><td rowspan="6">IPR/© Contracts management</td><td rowspan="4">Production department
Outsourced service
Press Office
Marketing manager
Legal department
Company accountant</td></tr>
<tr><td>CD/ROM
DVD</td></tr>
<tr><td>Web</td></tr>
<tr><td>TV, iTV, PDA mobile and other new media</td></tr>
<tr><td>Marketing</td><td>Promoting</td><td>Marketing manager
Legal department</td></tr>
<tr><td>Distribution & rights selling</td><td>Revenue management</td><td>Marketing manager
Legal department
Company accountant</td></tr>
</table>

The cost control, and overall, IPR and copyright management are usually managed in a parallel stream. As far as the impacts are concerned, cost control has to monitor that process development is kept in line with expectations and budgets retaining its profitability (ore even increasing it whenever possible) and therefore is a good part of the standard management process. On the other hand IPR and copyright management main interactions occur whenever some asset cannot be cleared and therefore has to be replaced. Such event may occur at any step and the impact may be marginal or relevant depending on a set of possible combinations of factors:

- The relevance for the publishing project of the object that could not be cleared;
- The reason for lack of clearance;
- The stage of the publishing process:
- The availability of a replacement / equivalent object…

2.2 Content production in the media environment

So far we have focused on traditional, desktop and multimedia publishing as they present the highest number of contact points and overlaps. In line of principle a similar approach applies to the editorial process of whatsoever product, yet digital TV, serials, movies and in general products requiring video production are somehow different as the design and planning phase is much longer and has several by-products (storyboards, drawings, scripts, books…) that in certain case will have their own existence and production cycle. Therefore, for pure reference, we will just give a very brief sketch of this process:

Activity	Involved roles

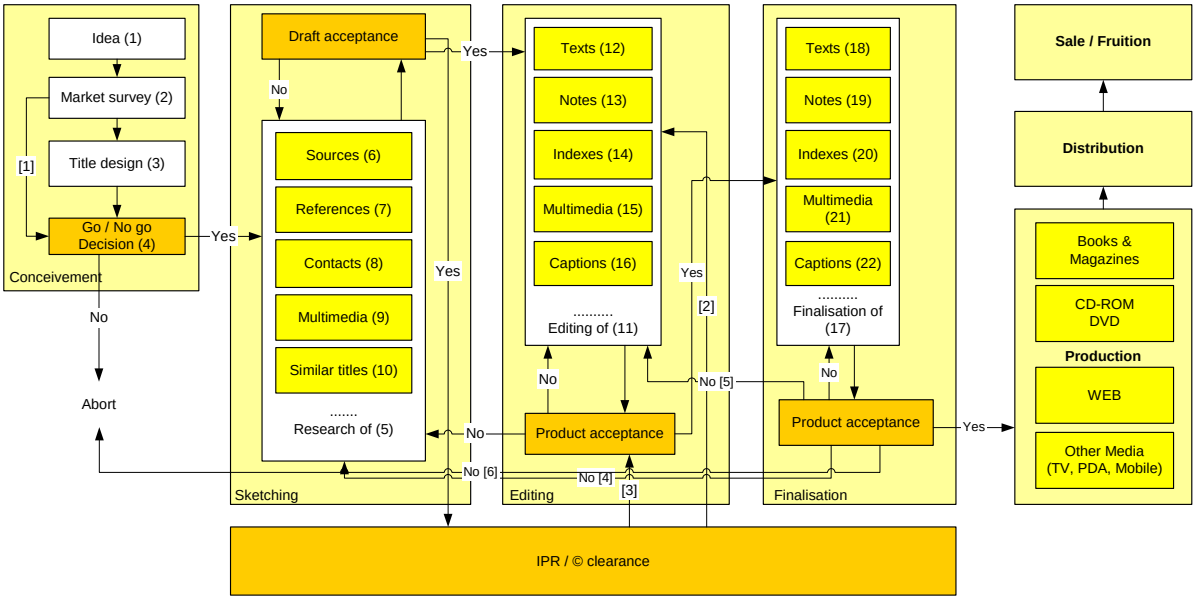
<table>
<tr><td>Idea</td><td colspan="3">Storyboard drafting</td><td>Author
Writer
Script editor
Art director
Director
Producer
Production accountant</td></tr>
<tr><td colspan="3">Cost estimation</td><td>Producer
Production accountant</td></tr>
<tr><td colspan="3">Go/No Go decision</td><td>Producer</td></tr>
<tr><td>Title design</td><td>Storyboard finalising
Scripting drafting
Casting preparation</td><td rowspan="2">Cost estima-tion refine-ment

Cost estima-tion refine-ment</td><td rowspan="4">Distribution & sale IPR/© Contracts management

Distribution & sale IPR/© Contracts management</td><td>Author
Script editor
Producer
Director
Casting director
Art director</td></tr>
<tr><td>Design & prepara-tion</td><td>Casting
Scripting
Lighting drafting
Shooting plan drafting
Effects drafting
Costumes drafting
Sound track drafting
Contracts finalising</td><td rowspan="3">Costume designer
Makeup designer
Composer
Audio Engineer
Graphic designer
Special effect designer
Fight arranger
Lighting cameramen
Production assistant
Location manager
Art director
Director
Casting director
Producer
Production lawyer
Production accountant</td></tr>
<tr><td>Development & shooting</td><td>Storyboard adapting
Lighting management
Shooting plan adapting
Effects management</td><td rowspan="3">Cost control</td></tr>
<tr><td>Post Production</td><td>Mounting
Sound addition
Effects addition
Packaging</td></tr>
<tr><td colspan="2">Marketing</td><td>Marketing manager</td></tr>
<tr><td colspan="2">Distribution & rights selling</td><td>Revenue management</td><td>Marketing manager
Production accountant</td></tr>
</table>

2.3 Content production flow

In terms of the present demonstrator what described so far can be summed up in the following schema; more details are provided in other deliverables describing user needs and requirements.



[1] There is no market
[2] There is a clearance problem impacting on editing
[3] There is a problem that may affect the Editing process and / or require product re-design
[4] There is a problem that requires re-design
[5] There is a problem that requires revising the finalisation process
[6] There is a clearance problem so serious that is worth stopping the process

2.4 The Kiosk factory case

Given the process described before (and largely based on the standard editorial value chain) we will address now the points where the integration with AXMEDIS will introduce significant changes. The present section is aimed specifically to cover the first part of the experimentation phase; that is: the content production. In more details, we will present here how we expect to integrate AXMEDIS into the existing production chain. As already mentioned we will operate on a by difference approach and therefore here we will focus only on the steps that are affected by AXMEDIS pointing out why they are affected and how. This is initially done by highlighting the affected process steps and then describing why. The revised proves will then look as follows:

<table>
<tr><th>Activity</th><th>Task</th><th></th><th>Involved roles</th></tr>
<tr><td colspan="3">Idea (1)</td><td rowspan="4">Management
Authors
Chief Editor
Press Office</td></tr>
<tr><td colspan="3">Market survey (2)</td></tr>
<tr><td colspan="3">Title design (3)</td></tr>
<tr><td colspan="3">Go / No Go decision [3] (4)</td></tr>
<tr><td rowspan="2">Research of (5)</td><td rowspan="2">Sources (6)
References (7)
Contacts (8)
Multimedia (9)
…
Similar titles (10)</td><td rowspan="3">IPR/© clearance</td><td>Management
Authors
Editorial board
Editorial staff
Press Office
Legal Department</td></tr>
<tr><td colspan="2">Draft acceptance [4]
(if positive the next step starts if not the previous is reiterated)</td><td>Management
Authors
Editorial board
Legal Department</td></tr>
</table>

[3] Such a decision is based on market data and production cost analysis to ensure the expected return on investment (ROI)

<table>
<tr><th>Activity</th><th colspan="3">Task</th><th>Involved roles</th></tr>
<tr><td>Editing of (11)</td><td>Texts (12)
Notes (13)
Indexes (14)
…
Multimedia (15)
Captions (16)</td><td colspan="2" rowspan="3">IPR/© clearance</td><td>Authors
Chief Editor
Editorial board
Editorial staff
Instructional designer
Press Office
Legal Department</td></tr>
<tr><td colspan="2">Product final acceptance
(if positive the next step starts if not the previous is reiterated)</td><td>Authors
Chief Editor
Legal Department</td></tr>
<tr><td>Finalisation of (17)</td><td>Texts (18)
Notes (19)
Indexes (20)
…
Multimedia (21)
Captions (22)</td><td>Chief Editor
Editorial board
Editorial staff
Instructional designer
Production department
Press Office
Legal Department</td></tr>
<tr><td colspan="3">Formal authorisation to start production
(if positive the next step starts if not the previous is reiterated. In this case IPR/© clearance should have been completed if not process may be suspended / stopped / cancelled)</td><td>Management
Chief Editor
Editorial board
Legal Department</td></tr>
<tr><td rowspan="4">Production</td><td>Books & Magazines</td><td rowspan="5">Cost control</td><td rowspan="5">IPR/© Contracts management</td><td rowspan="4">Production department
Outsourced service
Press Office
Marketing manager
Legal department
Company accountant</td></tr>
<tr><td>CD/ROM
DVD</td></tr>
<tr><td>Web</td></tr>
<tr><td>TV, iTV, PDA mobile and other new media</td></tr>
<tr><td>Marketing</td><td>Promoting</td><td>Marketing manager
Legal department</td></tr>
<tr><td>Distribution & rights selling</td><td>Revenue management</td><td>Marketing manager
Legal department
Company accountant</td></tr>
</table>

From the table is evident the impact of AXMEDIS on the overall and especially on the production process, the reason is reported hereafter also taking into account what previously mentioned about the traditional process.

Activity	Reason
Research … (5)	During the search process the P2P infrastructure, AXEPTool and Query support will reduce efforts related to searching content and address IPR/© clearance.
Editing of (11) Finalisation of (17) Production	During these process phases the composition and formatting engines will reduce efforts related to content production/aggregation while ensuring proper IPR/© management.
Marketing	P2P infrastructure, AXEPTool and Query support will reduce efforts related to content promotion and distribution (in terms of samples). The framework will help easing up addressing IPR/© issues enabling the definition of proper rules, licenses and distribution agreement in a simpler form.
Distribution & rights selling	P2P infrastructure, AXEPTool and Query support will reduce efforts related to content distribution (broadcast, active selections, publishing rules…). The framework will help easing up addressing IPR/© issues enabling the definition of proper rules, licenses and distribution agreement (including revenue management) in a simpler form.

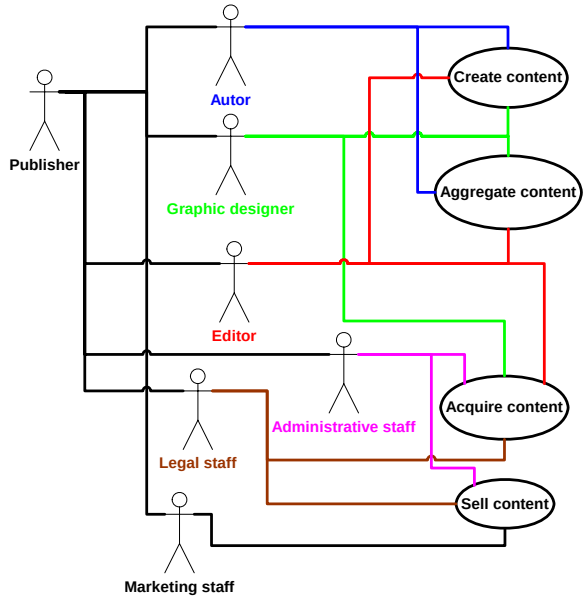
The content production process in the AXMEDIS framework for the kiosk environment has as a starting point the transposition of our archive on history of arts form its original format to an intermediate format that

[4] At this step starts a parallel process for managing © and IPR clearance performed in an asynchronous parallel fashion.

will represent the "raw" one for our experiment in AXMEDIS. This is just a part of a more complex process of content processing, filtering and adapting that covers mainly the last part of the value chain (even if it has direct impacts on some initial steps as if certain issues are not addressed at production time it will not be possible to solve them at this stage). What just stated would be valid per se in the publishing environment but is even more true in the case of the specific demonstrator as the "kiosk" is a un-attended environment dependent from a broadcasted updating process and with a fruition and delivery context that is quite unique as it is much more than the usual Info-kiosk or totem and basically comparable to a "v-ortal" (vertical portal). As a matter of fact any kiosk could be part of a set of kiosks (distributed on the territory) and devoted to a specific audience with tailored services and applications. Moreover the AXMEDIS kiosk will enable its user to access to services in a local area (defined by the WiFi support coverage). Having said this let's focus on the operations to be performed that basically are:

- Acquire content
- Create content
- Aggregate content
- Sell content

What just described is therefore translated in the following UML schemas (where for simplicity we have used a colour code to identify relations between actors and objects (please see the following picture).



In this scenario the first three operations are occurring at the kiosk factory and mainly foresee a set of actors operating with different level of competencies and responsibilities in the process. More specifically we have that:

Publisher – manages the overall process in terms of strategies and decision-making.

Author – creates the content (depending on the kind of "creation" could be an instructional designer designing a course, or a graphic designing a graphic template or layout…)

Graphic designer – takes care of content graphic finalisation process and of all QA aspects related to graphic.

Editor - takes care of content finalisation process and of all QA aspects related to content.

Administrative staff – manages the administrative aspect of the process (including but not limited to provisioning)

Legal staff – manages the legal aspect of the process (including but not limited to IPR, © clearance, protection, licensing…)

Marketing staff – manages the promotional aspect of the process as well as the distribution policies in cooperation with the publisher and the legal staff.

At the kiosk factory all these actors will have access to AXMEDIS framework, each for own role and activities. The result of their cooperation (given the previously mentioned impacts on the process) will lead to the generation of the content that will then be delivered (in broadcast) to the kiosks and used by them for their daily operation as described in other sections of the document.

2.5 Kiosk Factory and POPs usage

The overall scenario covers two levels, one called the kiosk factory dealing with content production and catalogue preparation, and one related to local kiosk POPs, this one comprising management (catalogue selection & loading, user management…) and content fruition .

In the kiosk factory the distributor (or the people preparing content for kiosk distribution) will take care of selecting the contents that will be presented into a catalogue, assign the proper category, rights and costs to each, prepare and distribute the catalogue.
At the POPs side the application has been designed taking into account the need to accommodate both terminal and PDA fruition, therefore a web-based interface has been designed and implemented. The user should register (and download the AXMEDIS viewer whenever needed) prior to be granted access to the application. In the registration phase some demographic data and some preferences are collected for subsequent usage (even though only very little data is mandatory). Once registered the user can access to the core application that will present the kiosk catalogue, where available contents are categorised and can be browsed or searched. The user can select content for preview and once the selection is performed the user can purchase the content and, upon process completion, also access to it using the specific player).

2.5.1 Where to find the material

The software for the application can be found on the project CVS (https://cvs.axmedis.org/newrepos/) under the "applications" area (https://cvs.axmedis.org/newrepos/Applications/) and more specifically inside the Kiosk Components section (https://cvs.axmedis.org/newrepos/Applications/kioskcomponents/) which is divided in Factory and Kiosk given the specific nature of the application. References for the two mentioned components are:

- Factory (https://cvs.axmedis.org/newrepos/Applications/kioskcomponents/Factory/)
- Kiosk (https://cvs.axmedis.org/newrepos/Applications/kioskcomponents/Kiosk/)

It is, nevertheless, necessary to take into account that the application demo makes usage of the underlying infrastructure either locally (that is having the AXMEDIS framework installed locally) or remotely (accessing to it via the provided web-services. To this purpose we will also recall that:

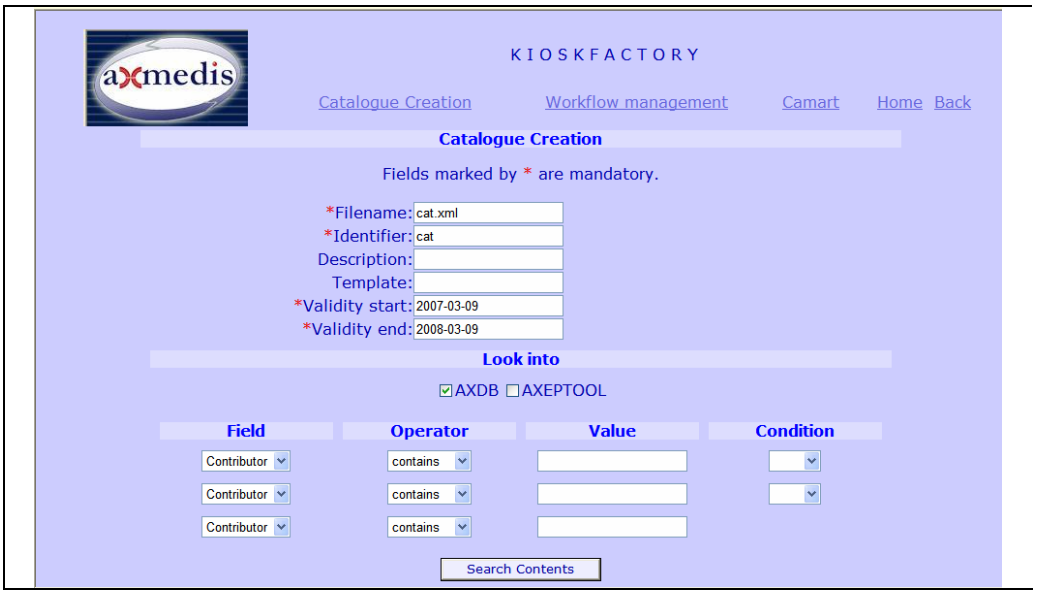
- AXMEDIS framework can be found at the address: https://cvs.axmedis.org/newrepos/Framework/
- AXMEDIS web-services can be found at the address: https://cvs.axmedis.org/newrepos/WebServices/

Last but not least it will be necessary to find ready made objects to be used for the testing, or to create them through the AXMEDIS tools.

2.5.2 Kiosk Factory

A publisher prepares content that wants to distribute on a particular target kiosk set (e.g. museums) via satellite. The publisher needs to select a list of objects, a catalogue template, an object template and specify when to deliver each object (including the catalogue) and to whom (i.e. the distribution kiosk). By using the AXMEDIS Query support the publisher searches the desired objects and selects the ones to be used to build the catalogue. By using the AXMEDIS CAMART he can explore the usage statics for the objects and automatically extract top and bottom ones. By using the P&P Editor the programme manager can create a programme to specify the time and destination channel of the objects to be distributed. On completion, the publisher activates the programme and during the programme life cycle the objects will be distributed until the programme has completed or the programme is removed from being activated.

Kiosk Publisher's home page (at the Kiosk factory)



Catalogue creation

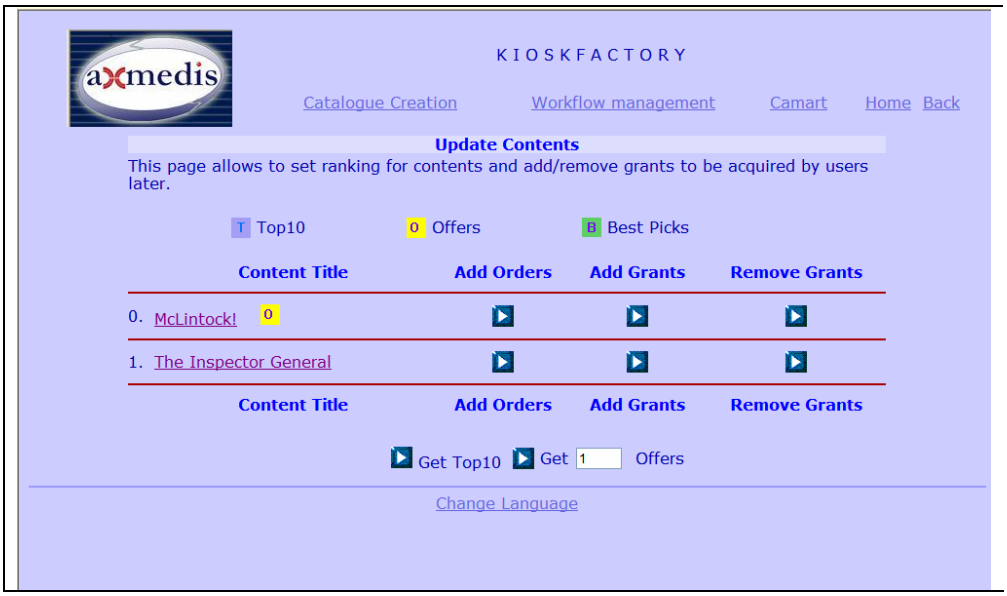
This is the interface that allows user to insert basic catalogue data (identifier, description, template, validity start date, validity end date) and a query for content retrieval.

Content search and selection for catalogue construction

The user can select among the results of the previous query the ones he wishes to add to the catalogue.

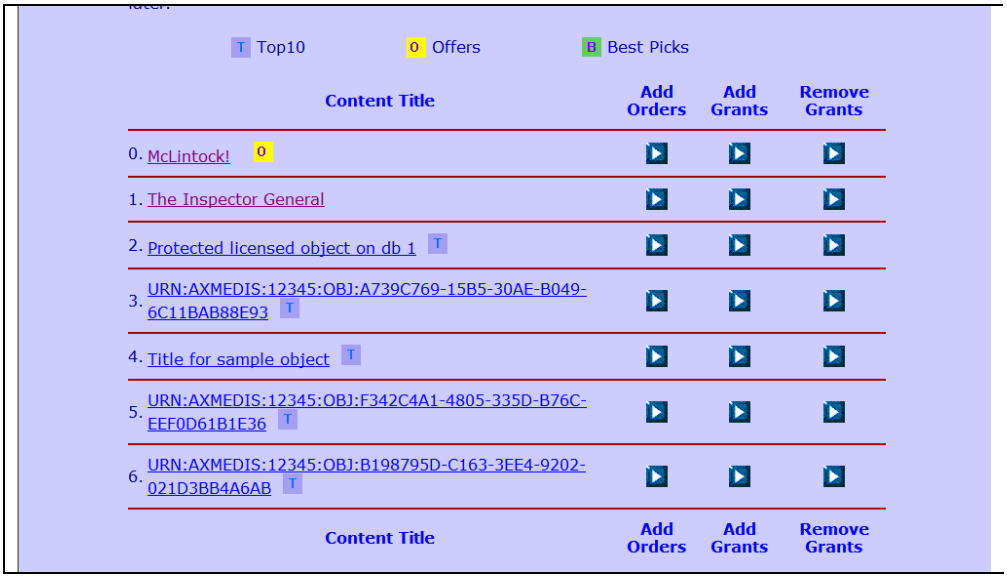
Content details

The user can also view the content metadata before adding to the catalogue.



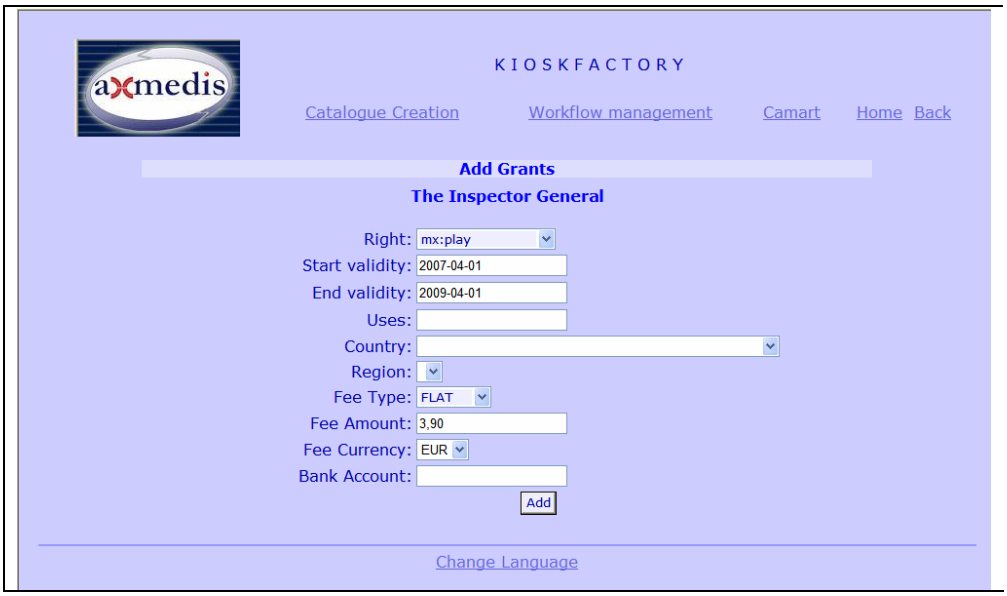
Once the user has selected some contents, he can insert them into the three catalogue categories: Top10, Offers, Best Picks. He can also retrieve automatically The Top 10 and the Offers contents (last ones are the "bottom" contents) querying the CAMART service through the buttons at the end of the page.

Content management for catalogue construction



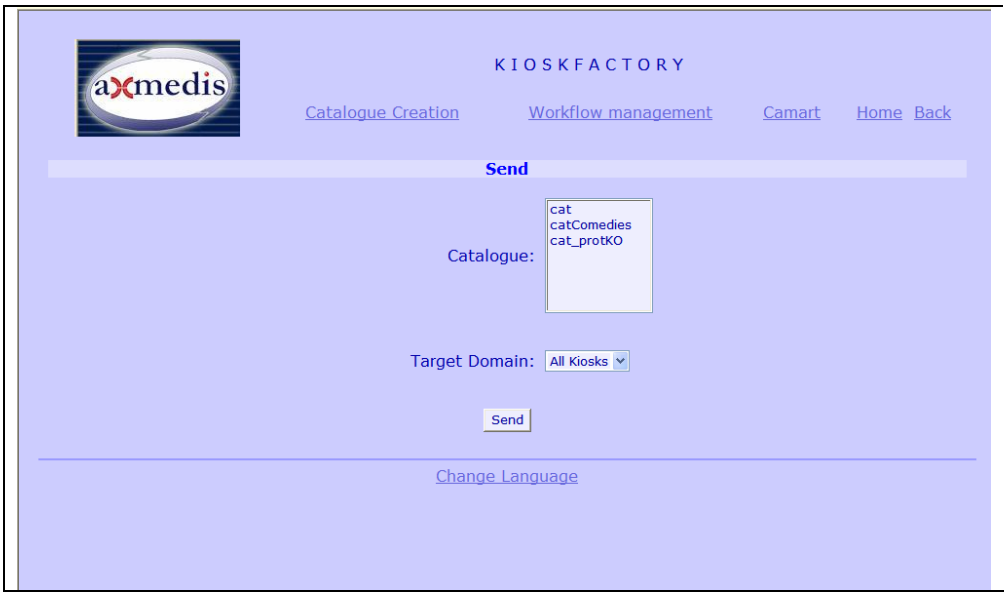
This is how a contents list is updated after automatic extractoion of Top10 contents. A little box appears near contents that have already been inserte dinto one or more categories: a violet T for Top10, a yellow O for Offers, a green B for Best Picks.

Contents arranging into categories



This is the interface for adding and removing grants that will be for sale to catalogue users. These grants are the ones that have been acquired by the KioskFactory from the contents' creators and for which a distributor license has been acquired.

Catalogue completion via grant attribution to content



Once the user decides that he has finished creating the catalogue he needs, he can send it to Kiosk POPs through the Satellite Channel.

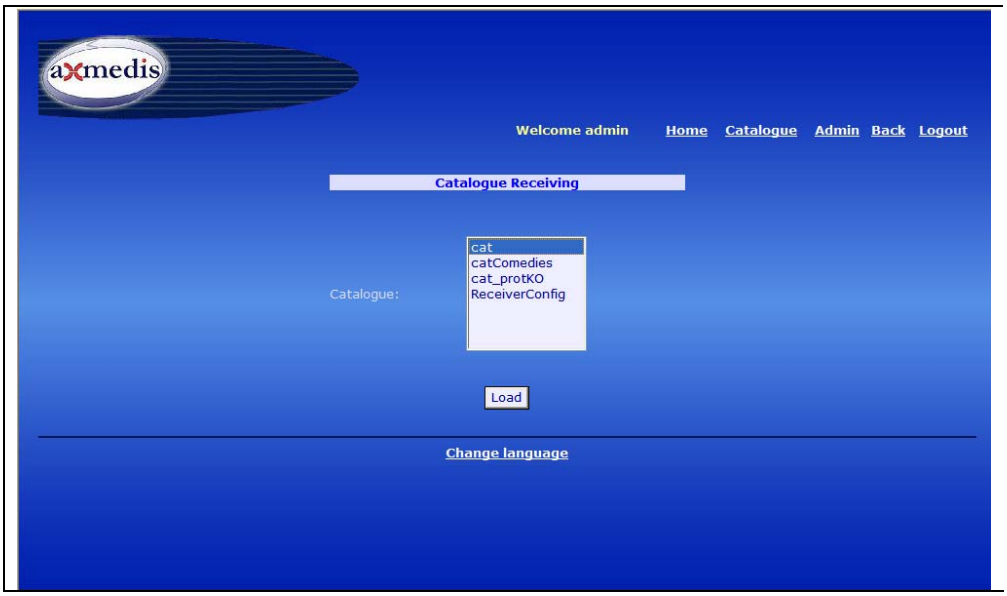
Catlaogue sending

2.5.3 Kiosk POP

At the Kiosk POP there are two sets of interfaces, one related to kiosk management (encompassing, catalogue user management from an "administrative" point of view) and one related to user content fruition. What follows is a guide to the POP functionalities. So what happens when a kiosk is used? The user shall register or log onto the kiosk infrastructure to be recognised and authorised to access to provided services. The user will be able to browse the catalogue and select content for delivery and fruition. In this latter phase the user will be able to select also the fruition model (acquisition, rental, pay per use…) and experiment what it looks like a DRM empowered system specifically designed to cover the whole value-chain from production to fruition. As a matter of facts the system will show how only allowed operation can be performed while all others are inhibited.

From the Administration section, an enabled user (the Administrator) can set up the current catalogue for the Kiosk and manage users' data.

Kiosk Administration



The Administrator can set up which catalogue can be browse by the users, choosing among the ones received through the Satellite channel.

Catalogue Loading

The Administrator can get the list of all the users of the Kiosk, view and modify their data and remove their account.

Users management

The Kiosk Administrator can also generate codes for the users, after payment. These codes will be used later in order to allow the users to acquire the grants over contents to which they are interested.

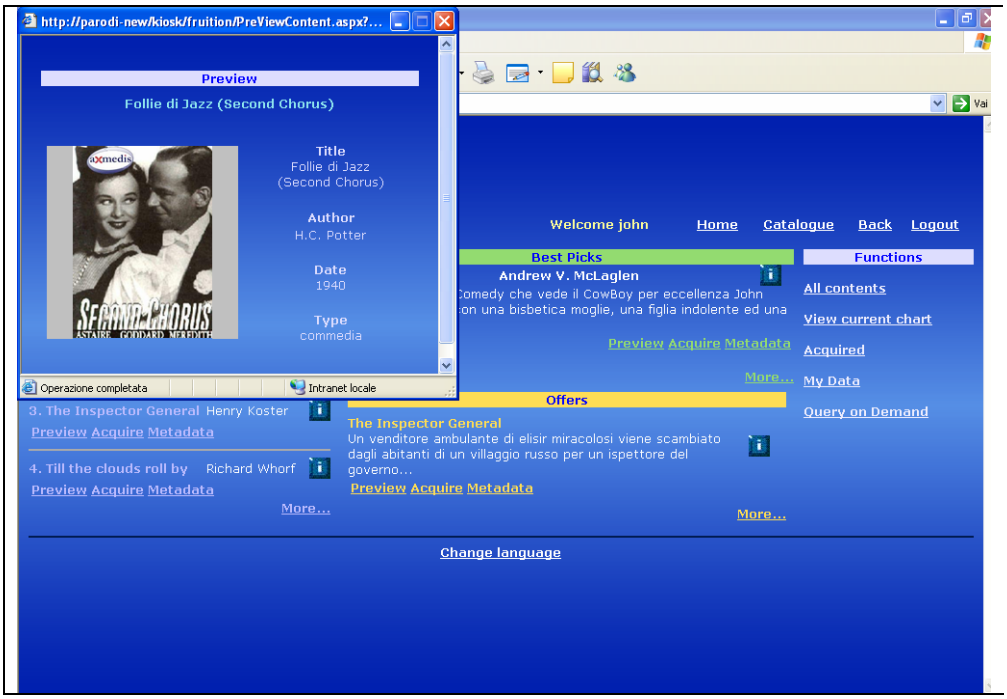
Codes generation

The first time an user accesses to the kiosk, s/he has to register, both on the AXMEDIs system and locally to the kiosk. Only a few data are mandatory.

User registration

Once the user is registered (or, if it is not her/his first time at the kiosk, after s/he has logged in) the kiosk catalogue will appear. As already seen for the catalogue creation, there are three main sections: Top10, Best Picks and Offers. There are some more functionalities accessible from the menu on the right: list of all the contents of the catalogue, current chart, contents for which some rights have already been acquired, personal data, access to Query on Demand.

Catalogue



The user can have a preview of the *contents*…

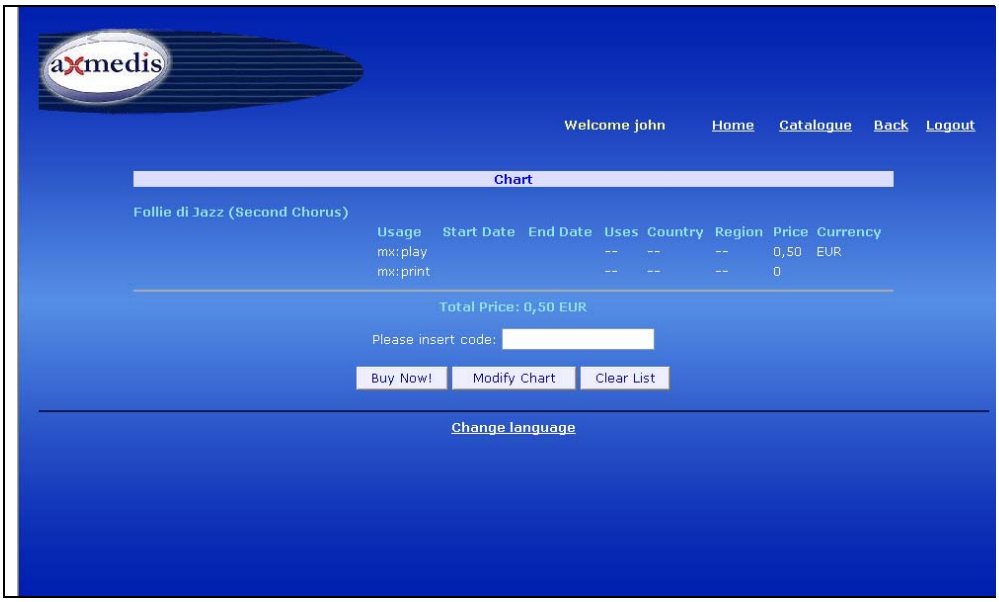
Preview

… take a look at their metadata…

View content metadata

Adding to chart

..and also decide to acquire some of the available grants for the object, adding to his personal chart…



Acquiring

… and confirming by inserting the code previously acquired.

The user can play a content according to the rights s/he as gained.

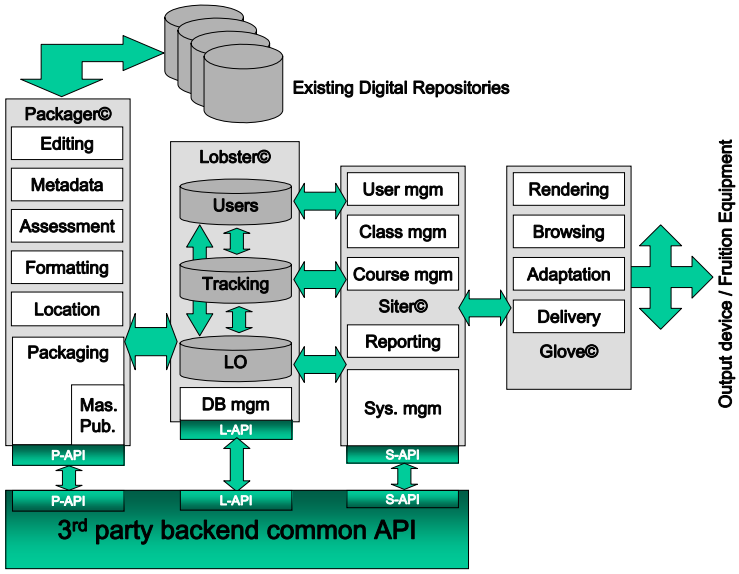
Content fruition

2.6 Kiosk factory overall architecture

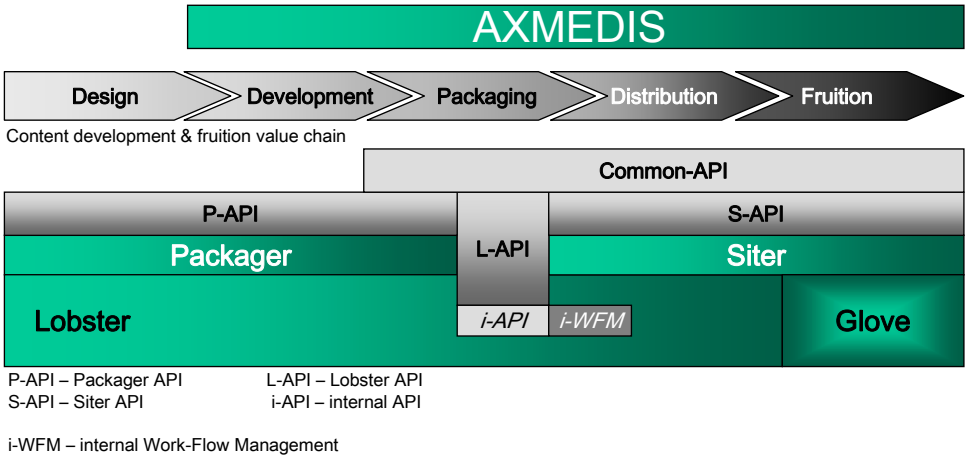
We have already mentioned the bi-partition of the environment in a kiosk factory and kiosks given the nature of the demonstrator and its purpose. The present section focuses on the kiosk factory that will comprise two interacting elements: AXMEDIS system and *learn eXact*® (LEX) production one. The AXMEDIS system has been described in other relevant documents (reported also in the reference section) it is now time to provide a description of LEX system. LEX is a proprietary environment developed in GIUNTI Interactive Labs during the past ten years and devoted to cover the whole value chain of e-learning publishing and fruition in full compliancy with international standards. Basically it comprises the following elements:

- *Packager*© - the authoring and packaging facility of the *learn eXact*® platform;
- *Lobster*© - the data manager that handles all published content of the *learn eXact*® platform;
- *Siter*© - the e-learning management module of the *learn eXact*® platform;
- *Glove*© - the e-learning rendering and fruition module of the *learn eXact*® platform;
- D-Repository - any repository compliant to the *Digital Repository* standard (WebCT, BalckBoard…)
- Back-end - any existing backend compliant with the set of API exposed by *learn eXact*® platform.

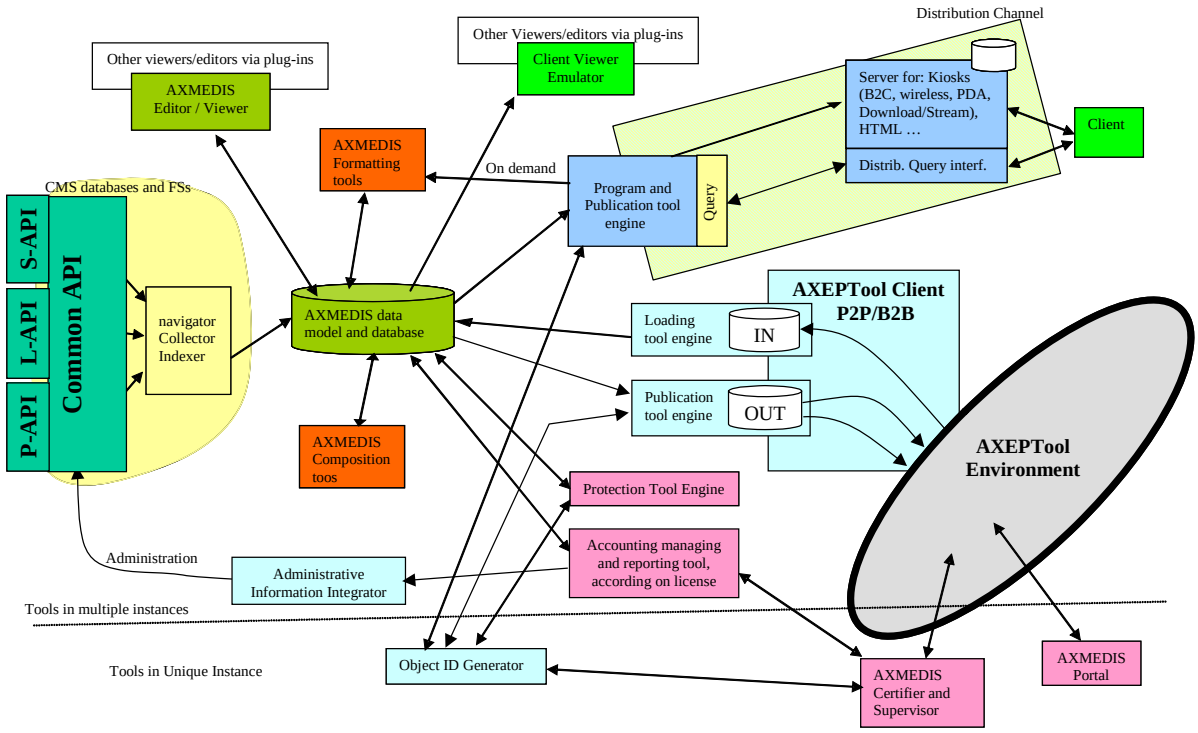
The most relevant aspect in the D-Repository connection is the possibility to use their content either as source or destination of the packaging process as all that is needed is the compatibility with IMS packaging standard. This feature was developed originally to grant users the possibility to continue exploiting the already in place content sources or distribution facilities and infrastructure even in the occasion of a shift in either production or delivery framework. The same principle brought to the development of a set of API for interconnecting with external back-ends; this is actually the way that will be followed to exploit AXMEDIS empowered production and distribution value chain. As stated in the TA it is expected that each distributor may keep on using own distribution media, but should also take advantage of the powerful support provided by AXMEDIS. Given the layered structure of GIUNTI Authoring and Learning Environment, this has been possible operating at API level in the area previously generically marked as “backend” (for example SAP…). More specifically it has been necessary to define a set of new API to allow AXMEDIS co-operation with the new third party environment. In more detail we have the following schema:



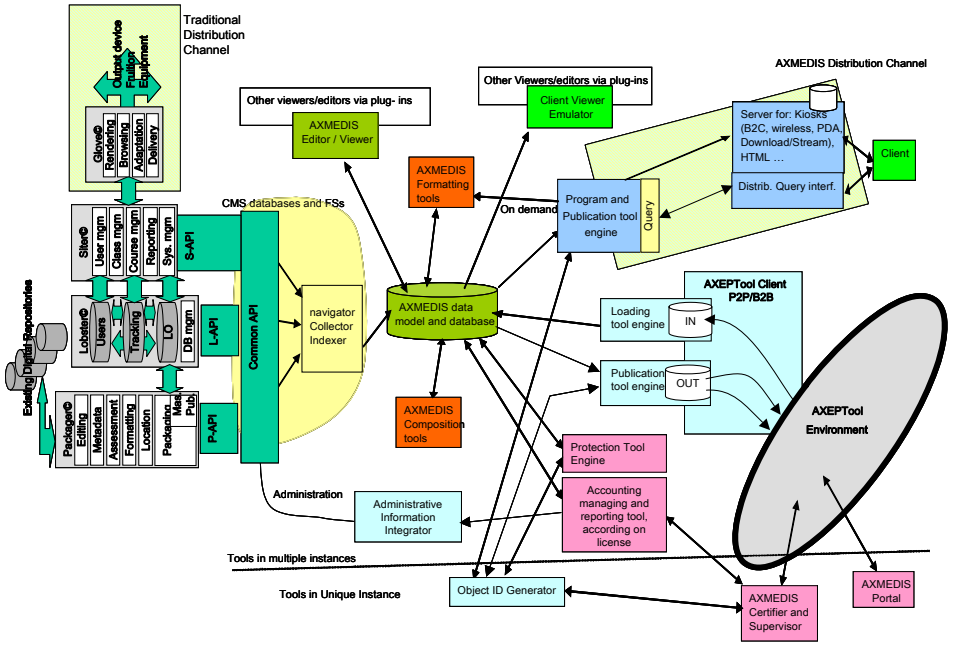
From the previous picture it is evident that the two environments have always a "mediated" interaction, and this is achieved through proper APIs sets. Having said so is necessary to examine in more detail the level of functionalities covered by the various set of API in respect to the process value-chain. In the following diagram is summarised the possible level of interaction between AXMEDIS as a backend and our environment. The set of common API allow interaction mainly at packaging, distribution and fruition level (at least as far as content delivery is concerned) even though the rendering tool (Glove) is designed only for e-learning purposes and therefore is not compatible with the overall aim if the project. On the other side at design and development phase the possibility of interaction is limited (especially at design) to data import/export.



From the above schema is evident that the Lobster embeds a workflow management system used to handle cooperative editing of content. The packager can toggle, via internal API, the status of a package (LO or course) between the status of "locked" and "unlocked" while the Siter can manage, via internal API, all states of the package. The internal API is implemented via web-service and it is not exposed to the backend. Having stated this is now necessary to see how the overall architecture (at the kiosk factory) will look like at the end of the integration phase.

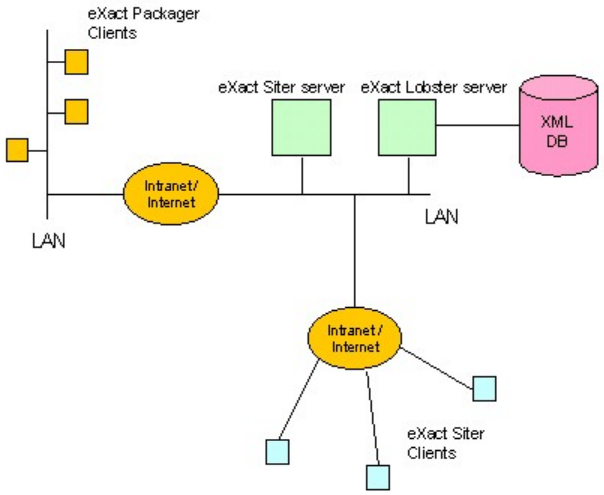


Interaction will basically occur at the level of the CMS and file system thanks to the 3rd party backend common API as highlighted in the picture above. It is worth mentioning that the schema is an adaptation of the one describing AXMEDIS architecture and reported both in the TA and in the framework specifications. The only relevant change is the one depicting the interaction between AXMEDIS (seen here as a backend) and the GIUNTI Authoring and Learning Environment. Having stated this the overall picture becomes the following:



2.7 *learn eXact* Installation and Configuration

The diagram illustrates a typical reference network configuration for the *learn eXact*® platform:



The setup shows a company network with an Intranet on which there are two centralized servers: one web server for the *eXact Siter* and one XML server for the *eXact Lobster*® (Tamino® XML server plus access layer). The platform users can comprise two types: Packager users (from which the authors publish Courses and LO on the *eXact Lobster*) and *eXact Siter* clients (from which the users access the platform services and courses). Please check with your *learn eXact* Reseller the system specification required for best operating performance.

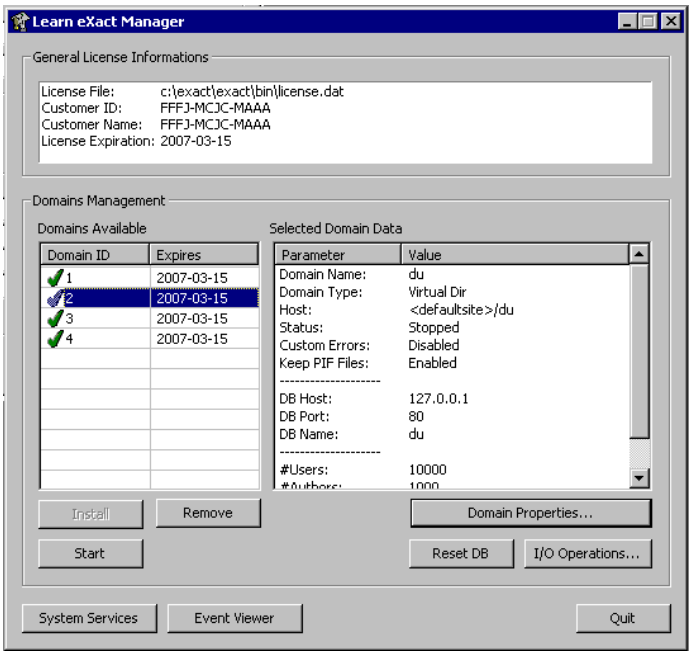
Note: Actually, a massive use of the *eXact Lobster* database and of its "Working Areas" requires a system installed on two servers. The first one is the *learn eXact* front-end, for publishing and management; the second one is specific for Tamino. Please contact the *learn eXact* technical support (www.learneXact.com) if you need any further information.

Note: In system architectures with intensive information traffic, we suggest you raise the data processing timeout limit from 90 (default) to up to 600 seconds. Please follow the procedure below:

1. From Windows *Start* menu go to *Settings* > *Control Panel* > *Administrative Tools* > *Internet Services Manager*
2. On the *Tree* panel in the *Internet Information Services* (IIS) window, spread all *Site* structures and locate the *eXact* Virtual Directory (usually in *Default Web Site*)
3. Select *Properties* from *eXact* right click menu. In the new window, select *Home Directory*. N.B.: If your Windows version is earlier than XP, the *Home Directory* may NOT be available at the level indicated. In this case, please right click on the *Web Site* that includes the *eXact* Virtual Directory and select *Properties* from the pop-up menu. Please select *Home Directory* In the new window
4. On the *Home Directory*, please select *Configuration....*
5. In the new window, select *App Options*, then type in a new, higher value in the *ASP script Timeout* filed (default value: 90 sec., suggested value: 600 seconds or higher). This way , the system can successfully run complex operations, even with large amounts of data.

2.7.1 *learn eXact* Manager

From the *learn eXact Manager* you can fully manage your *learn eXact*® domains. To access *Learn eXact Manager* please access the *Start* menu on Windows and locate Learn eXact Manager within the *Program Files>Interactive Labs>Learn eXact v4.0* folder:



On this dialog window you may

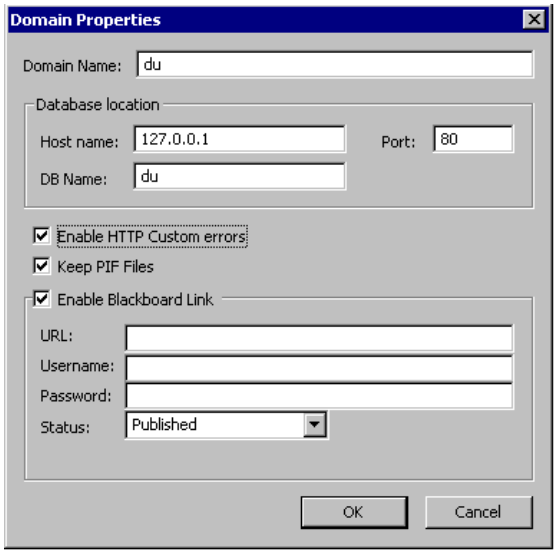
- Modify Domain Properties (enable/disable Http custom errors, enable/disable blackboard link, keep PIF files)
- Perform massive import/export of user data
- Reset your Lobster database (click on the *Reset DB* button). If you perform this last action, all data on current database will be lost.

It is then possible to customize some default settings when installing new domains.

Note: You need to restart IIS in case you have customized default System Settings in your *learn eXact* Web Interface (*Administration Page>Settings>System*: please refer to the section entitled "Learning Management within the eXact Siter" for more information).

2.7.2 To Edit Domain Properties

1. On the *learn eXact* Manager main page, select a domain on the *Available Domains* pane
2. Click the *Domain Properties* button
3. The *learn eXact* Manager *Domain Properties* window displays:



- Select the *Enable HTTP Custom Errors* checkbox to enable all HTTP custom errors report for the selected domain
- Select *Keep PIF files* to preserve pif files within the domain
- Select *Enable Blackboard Link* to activate a domain blackboard link (complete spaces below with blackboard link URL, Username and Password to access it and its status - *Locked* or *Published*)

2.7.3 To install a new domain possibly keeping default settings

Commonly, when you install a new domain from the *learn eXact Manager* window (click *Install*), you may choose to copy Siter and Glove themes after "default" theme. This will allow to correctly complete "theme.xml" and Siter and Glove "domainName" folders for image upload. If you decide not to take this option and, in any case, if you change your mind and decide to "manually" copy and overwrite these files on a second time (as PC administrator), YOU HAVE TO reload both themes as application administrator. If you don't, the system will cache info and will keep "default" theme path. The same applies to strings. As an alternative to application reload, you may restart IIS or your PC as PC administrator.

2.7.4 To perform massive import/export of user data

To import/export massive sets of users into/from a given domain, on *learn eXact Manager* select the *I/O Operations…* button.

2.7.5 To import user data from a CSV file

To import users from a Comma Separated Values (.csv) file, please select the *Import data from csv and xml* option. In order to import a set of users from a CSV file, please make sure that the CSV file conforms to the following specifications:

1. The first line of the file should contain field names, separated by a comma (no commas other than the separating ones are admitted on this line).
2. All fields should be separated by commas.
3. Each field containing any comma or colon should be in quotation marks.
   - In any other case the fields should not be necessarily in quotation marks.
   - If any field already contains quotation marks, you should insert them twice.
4. All records, except for the last one, should be terminated by an end-of-line character:
   - "\r\n" in a Widows environment
   - "\n" in a Unix environment
   - "\r" in a Mac environment

For the purpose of clarification, given a record of the following kind:

```
John Smith
Main Street, #11
"Springfield"
```

it could be transformed in the following way:

```
John Smith,"Main Street, #11","""Springfield"""
```

or else

```
"John Smith","Main Street, #11","""Springfield"""
```

More in general, a csv file valid for the output could be:

```
==========================================
Import.csv
==========================================
"First Name","Second Name","Last Name","Gender","Date of Birth","E-mail","Address","PO-
Box","Locality","Postal Code","Country","Language","Username","Password"
John,,Smith,M,17-02-
1974,j.smith@giuntilabs.it,"Main Street, #11",,"""Springfield""",16030,United Kingdom,"uk","
jsmith","john"
User0001,User0001,User0001,0,17-02-1974,User0001@giuntilabs.it,,,,,it,NewUser0001,
User0002,User0002,User0001,0,17-02-1974,User0002@giuntilabs.it,,,,,it,NewUser0002
"User0003","User0003","User0003","0","17-02-
1974","User0003@giuntilabs.it","","","","","","it","User0003",""
==========================================
```

Of course all changes are made in conformance with the above described rules are valid.

Note: The CSV import operation does not include the assignment of any user role to imported users. It is up to the System Admin to assign them a basic "User" role within the Siter. Refer to the *User and Administrator Profiles* section for more information.

2.7.6 To export user data to a CSV file

To export users to a Comma Separated Values (.csv) file, please select the *Export data to csv* option. Choose your destination directory, enter a valid file name and select *Save*.

2.7.7 To export user data to a Unicode CSV file

To export users to a Comma Separated Values (.csv) Unicode file, please select the *Export data to csv Unicode* option. Choose your destination directory, enter a valid file name and select *Save*.

2.7.8 Client Requirements for Web Access

Hardware Requirements:
CPU: Pentium II class
RAM: 64 MB

Software Requirements:

OS:	Windows® 98 SE, ME, NT Workstation SP4 or higher, 2000 Professional or higher, XP Home or higher.
Web browser:	Microsoft® Internet Explorer 5.0 or higher, Netscape® 4.77 or higher
Other:	Java Virtual Machine (Microsoft JVM 5.0 or Sun JRE 1.4.2 recommended) Additional plug-ins required to view animations and movies

Note:
The first release of Windows XP includes an Internet Explorer version without the virtual machine required for Java encoding (necessary for *eXact Glove*). Subsequently, after a legal dispute conducted by Sun against Microsoft, Redmond House included this component in Service Pack 1 for XP. To view *eXact Glove* on XP Platform, it is necessary to:

- install the JRE provided by Sun, or else
- - install the Service Pack 1 for Win XP provided by Microsoft.

2.7.9 eXact Packager Client Installation

The *eXact Packager* client application can be installed directly from the CD-Rom on any client workstation. On the client machine, run Setup.exe in the *eXact Packager* folder of the Installation CD and follow the instructions appearing on the screen.

Hardware Requirements:

CPU:	Pentium III class or higher
RAM:	128 MB minimum (256 MB suggested)
Free Disk Space:	50 MB (required for the installation)
Network Card	

Software Requirements:

OS:	Windows® 98 Second Edition

Windows NT Workstation Service Pack 6a (SP6a) or higher,
Windows 2000 Professional Service Pack 2 (SP2) or higher,
Windows XP Home/Professional (SP1) or higher

Web Browser: Microsoft® Internet Explorer 6.0 or higher

Other: Microsoft .NET Framework 1.1; some additional plug-ins may be needed to view animations and movies

Note: Both *eXact packager* and *learn eXact* Server (Lobster/Siter) require .Net Framework, version 1.1 (.4322). The Framework must be installed on your PC following these indications: Windows Update[5] direct Download.[6]

2.7.10 Types of eXact Packager License

There are two different types of license:

- Server License: if a server fingerprint has been produced, you can install the *eXact Packager* in accordance with the number of client workstations indicated in the license file within the terms of the license agreement.
- Client License: each *eXact Packager* installation requires the fingerprint of the client workstation which permits a remote use of the Packager, without any connection to the LOBSTER archive server.

2.7.11 To request and install a Client License

1. After installing the *eXact Packager*, select *Programs > Interactive Labs > eXact Packager 2.0 > Request License* in the Windows® Start menu.

The system will automatically calculate the fingerprint code for the unique serial number relating to a set of hardware devices installed on the machine. The fingerprint will appear in a special dialog window.

2. To run an e-mail message on the predefined e-mail application, click on the *eMail Request* button.
3. Insert your data and fingerprint code.
4. Send the message to licenses@learneXact.com.

When the license file is returned to you, save it in the main folder where the application XPackager.exe has been installed.

Note: In order to produce a fingerprint of either the client or server workstations, a network board and NetBIOS must be installed.

2.7.12 To uninstall eXact Packager

To uninstall eXact Packager just perform the standard Windows application removal procedure:

- From Control Panel launch "Add/Remove programs"
- Select "eXact Packager v1.x" and click on "Change/Remove"
- Follow the displayed instructions.

Once the procedure has been performed, all files and folders created by the installation program will be removed from your disk. However, all data added after the installation will be preserved on disk (Client Licence file, Courses, Resources, log files).

[5] http://v4.windowsupdate.microsoft.com/en/default.asp

[6] http://www.microsoft.com/downloads/details.aspx?FamilyId=262D25E3-F589-4842-8157-034D1E7CF3A3&displaylang=en

2.8 Kiosk Installation and configuration

From here and for the following, be $REPOS$ = https://cvs.axmedis.org/newrepos/ (the CVS code repository of the project)

Prerequisites:

- Microsoft.NET Framework v1.1.4322 or higher
- MySQL 4.1.16 or higher
- Internet Information Servcices (IIS) 5.1 or higher

Steps:

- Factory side:
  - create a folder, be it Kioskfactory_folder and copy in it:
    - $REPOS$Applications/kioskcomponents/Factory/bin (create a bin subfolder)
    - $REPOS$ Applications/kioskcomponents/Factory/source (all the files inside, recursively)
  - Open Web.config file inside Kioskfactory_folder and check the application parameters, edit them if you need to
  - Create from IIS a virtual folder and make it point to the Kioskfactory _folder, name it for example “kioskfactory”
  - To test go to:
    http://<your_host_name>/kioskfactory/Default.aspx
    you should be able to see the main page of the application.
- POP side:
  - create and populate a database using the scripts at $REPOS$Applications/ kioskcomponents /Kiosk/doc/other/
  - create a folder, be it Kiosk_folder and copy in it:
    - $REPOS$Applications/kioskcomponents/Kiosk/bin (create a bin subfolder)
    - $REPOS$ Applications/kioskcomponents/kiosk/source (all the files inside, recursively)
  - Open Web.config file inside Kiosk_folder and check the application parameters, edit them if you need to
  - Create from IIS a virtual folder and make it point to the Kiosk _folder, name it for example “kiosk”
  - To test go to:
    http://<your_host_name>/kiosk/Default.aspx
    you should be able to see the main page of the application.

2.9 Interfacing Lex with a backend

eXact Lobster is a full XML and eLearning standards native content storage solution that enables to store and retrieve contents (raw assets, learning objects and courses) and to store, query and retrieve their related metadata, in a common and interoperable digital repository. Lobster implements a WebService that enables third party applications to use Lobster for publishing, querying and retrieving contents and their metadata by means of WebServices, SOAP, HTTP and FTP technologies. The present document describes in detail this WebService. Lobster relies on the XML binding of the IMS Content Packaging Specification.(IMS CP). Any content to be published has to be made available to Lobster as a Package Interchange Format (PIF). Thereofore from now on, with the term *package*, we refer to all types of contents: asset, learning objects, and courses. A PIF is a ZIP archive which contains an xml file, named imsmanifest.xml, and some (or all) binary files referenced into the imsmanifest.xml. The imsmanifest.xml, schema available at *www.imsglobal.org*[7], contains several information used by the Lobster to store and retrieve packages, in particular:

[7] http://www.imsglobal.org/content/packaging/

1. The value of the attribute **identifier** of the element **manifest**, it is used by Lobster to identify the package once published. Therefore this information must be explicitly expressed when referring to the package in some of the methods of the WebServices.
   NOTE When publishing or updating, is the third party application responsibility to check consistency between the value of the identifier of the manifest file and the values of the identifier passed as parameter into the WebServices methods.
2. The value of the element **metadata** and of its sub-element is used by the Lobster to retrieve packages on the base of the values of their metadata.

2.9.1 Web Service Interface Description

This paragraph contains a detailed description of the Lobster WebService. Each method of the interface is described by a table; the related information are structured as shown in the table below:

Name of the method	
Description of the method	
SOAP XML description of the message request Parameters passed to the method	*Parameters passed by the method*
	Parameters returned by the method
SOAP XML description of the message response Values returned by the method	

Login()	
Performs mandatory login procedure on the Service. By providing authentication information this function opens a working session and returns a session identifier. Every operation on Lobster must start with a call to Login() and must end with a call to Shutdown().	

`<message name='Login'>`
`<part name='Domain' type='xsd:string'/>`
`<part name='User' type='xsd:string'/>`
`<part name='Pass' type='xsd:string'/>`
`</message>`
	`Domain name`
	`Username`
	`Password`

`<message name='LoginResponse'>`
`<part name='SID' type='xsd:string'/>`
`</message>`
	`Session ID`

Shutdown()	
Performs logout operation closing the specified session.	

`<message name='Shutdown'>`
`<part name='SID' type='xsd:string'/>`
`</message>`
	`Session ID`

`<message name='ShutdownResponse'>` `</message>`	

GetVersion()	
Retrieves Lobster Web Service version number ("4.0" for the version of this document).	
`<message name='GetVersion'>` `</message>`	

`<message name='GetVersionResponse'>`
`<part name='Version' type='xsd:string'/>`
`</message>`
	`Version number`

KeepSessionAlive()	
Keeps a session alive. Clients must call this function periodically during a session, with the frequency provided by the value of the tag <pollinginterval> returned by GetDomainInfo().	

`<message name='KeepSessionAlive'>`
`<part name='SID' type='xsd:string'/>`
`</message>`
	`Session ID`

`<message name='KeepSessionAliveResponse'>`	

```
</message>
```

GetDomainInfo()

Retrieves information about the domain you have logged in. The function provides general information such as material location, FTP location, authentication data.
Information are provided in XML format:

```
<message name='GetDomainInfo'>
       <part name='SID' type='xsd:string'/>                    Session ID
</message>
```

```
<message name='GetDomainInfoResponse'>
       <part name='XMLResult' type='xsd:string'/>              Information XML*
</message>
```

* this is an example of the XML format returned:

```
<?xml version="1.0"?>
<domaininfo>
   <domainurl>:80/domainname</domainurl>
   <coursedomainpath>nameofdomain</coursedomainpath>
   <ftpuser>eXactFTPUser</ftpuser>
   <ftppassword>ftppass</ftppassword>
   <ftpport>21</ftpport>
   <pollinginterval>60</pollinginterval>
</domaininfo>
```

FindPackage()

OBSOLETE. Use FindPackageInfo2() instead.

GetPackageFolder()

Retrieves the folder where the Package specified by PackageID is deployed on the server

```
<message name='GetPackageFolder'>
  <part name='SID' type='xsd:string'/>                    Session ID
  <part name='PackID' type='xsd:string'/>                 Package ID
</message>
```

```
<message name='GetPackageFolderResponse'>
       <part name='Folder' type='xsd:string'/>             Folder name
</message>
```

InsertPackageRequest()

Makes a request to store a new package on Lobster. A package unique identifier must be provided with the request; the unique identifier must be equal to the manifest identifier to be uploaded. The Object type provides information about the type of the package: Course or Learning Object.
The return value provides the filename you have to use for PIF uploading.

```
<message name='InsertPackageRequest'>
       <part name='SID' type='xsd:string'/>                  Session ID
       <part name='PackID' type='xsd:string'/>               Package ID
       <part name='ObjType' type='xsd:unsignedInt'/>         Object Type*
</message>
```

```
<message name='InsertPackageRequestResponse'>               Name of the file to be uploaded
       <part name='FileName' type='xsd:string'/>            via FTP
</message>
```

* the "Object Type" parameter shoud have one of the following values:
0: The package contains a Course
1: The package contains Resources/Learning Objects

UpdatePackageRequest()

Makes a request to update a package on Lobster.
To perform an Update operation, the user connected must have *write* permissions on the package.

```
<message name='UpdatePackageRequest'>
       <part name='SID' type='xsd:string'/>                  Session ID
       <part name='PackID' type='xsd:string'/>               Package ID
</message>
```

```
<message name='UpdatePackageRequestResponse'>                    Name of the file you must upload
       <part name='FileName' type='xsd:string'/>                 via FTP
</message>
```

DeletePackage()

Deletes a package from Lobster. To perform a Delete operation, the user connected must have write permissions on the package.

```
<message name='DeletePackage'>
       <part name='SID' type='xsd:string'/>
       <part name='PackID' type='xsd:string'/>
</message>
```

```
<message name='DeletePackageResponse'>
</message>
```

GetWAList()

Returns the list of the Working Areas of the user.

```
<message name='GetWAList'>
       <part name='SID' type='xsd:string'/>                      Session ID
</message>
```

```
<message name='GetWAListResponse'>
       <part name='Result' type='xsd:string'/>
</message>
```

* this is an example of the XML format returned:

```
<XML>
   <WA>
      <ID>100</ID>
      <TITLE>My WA</TITLE>
      <DESCR>My WA Description</DESCR>
   </WA>
</XML>
```

SetInsertOptions()

Specifies if the package to be published must be tagged as “public accessible” and/or must be inserted into a particular Working Area.

```
<message name='SetInsertOptions'>
       <part name='SID' type='xsd:string'/>                      Session ID
       <part name='TargetWAID' type='xsd:string'/>               WA identifier for automatic enroll-
                                                                 ment into a WA.
       <part name='AllowPublicAccess' type='xsd:int'/>           1 for public access material, 0 oth-
</message>                                                       erwise.
```

```
<message name='GetWAListResponse'>
       <part name='Result' type='xsd:string'/>
</message>
```

* this is an example of the XML format returned:

```
<XML>
   <WA>
      <ID>100</ID>
      <TITLE>My WA</TITLE>
      <DESCR>My WA Description</DESCR>
   </WA>
</XML>
```

GetPackageInfo() Available in version 2.3

Retrieve status information about the specified Package.

```
<message name='LexSOAPConnector.GetPackageInfo'>
   <part name='SID' type='xsd:string'/>                          Session ID
   <part name='PackID'type='xsd:string'/>                        Package ID
</message>
```

```
<message
name='LexSOAPConnector.GetPackageInfoResponse'>                  XML Result
```

```
   <part name='Result' type='xsd:string'
</message>
```

This is an example of the XML format returned:

```
<package packid="Course1" status="locked" islastmodifier="yes">
   <userrights canupdate="yes"/>
   <allowedstatus>unlocked</allowedstatus>
</package>
```

The `status` attribute shows the current Package status.
The `islastmodifier` is true when the current user is the last modifier of the status. If the status is locked and the last modifier is the current user, it means that the current user is the user that has lock the material.
The `canupdate` attribute is true if the user has write privilege on the Package.
The list of the `allowedstatus` tags shows a list of the allowed status that the current user can set.

SetStatus()	Available in version 2.3

Change the current status of a Package.

```
<message name="LexSOAPConnector.SetStatus">
   <part name="SID" type="xsd:string"/>                Session ID
   <part name="PackID" type="xsd:string"/>             Package ID
   <part name="Status" type="xsd:string"/>             The new Package status. Can be "locked" or "unlocked".
</message>
<message name="LexSOAPConnector.SetStatusResponse">
</message>
```

The SetStatus() can easily throw and E_PERMISSION exception if the current user has not the privilege to change the status. Use GetPackageInfo() before calling SetStatus() in order to check if the user has the correct rights.

FindPackageInfo()	Available in version 3.1

OBSOLETE. Use FindPackageInfo2() instead.

GetPackageHistory()	Available in version 4.0

Provides information about the version history of the specified package.

```
<message name="LexSOAPConnector.GetPackageHistory">
   <part name="SID" type="xsd:string"/>                Session ID
   <part name="PackID" type="xsd:string"/>             Package ID
</message>
<message
name="LexSOAPConnector.GetPackageHistoryResponse">
   <part name="Result" type="xsd:string"/>             XML Result
</message>
```

This is an example of the XML format returned:

```
<materialhistory gid="100001" packid="TEST">
   <version datetime="2006-05-03T13:59:05.698+00:00" label="1.0" number="1" old-
```

```
status="draft"   status="draft"   user="00"   useremail="admin@host.example.net"
userfn="System Admin">
     <comment>Fist Version</comment>
  </version>
  <version datetime="2006-05-03T13:59:44.449+00:00" label="1.1" number="2" old-
status="draft"   status="draft"   user="00"   useremail="admin@host.example.net"
userfn="System Admin">
     <comment>Some fixing...</comment>
  </version>
  <version datetime="2006-05-03T14:00:30.621+00:00" label="2.0" number="3" old-
status="draft"  status="published"  user="00"  useremail="admin@host.example.net"
userfn="System Admin">
     <comment>This is the final release.</comment>
  </version>
</materialhistory>
```

NotifyUploadDone2()	Available in version 4.0
Like NotifyUploadDone(), but provides the possibility to specify a comment and a label for the new version that you are publishing.	

`<message name="LexSOAPConnector.NotifyUploadDone2">`
`  <part name="SID" type="xsd:string"/>`
`  <part name="VersionLabel" type="xsd:string"/>`
`  <part name="VersionComment" type="xsd:string"/>`
`</message>`
	`Session ID`
	`Label`
	`Comment`

`<message name="LexSOAPConnector.NotifyUploadDone2Response">`
`  <part name="Result" type="xsd:unsignedInt"/>`
`  </message>`
	`New version number`

This function also returns the incremental number of the new version that has been created.	

SetVersionStatus()	Available in version 4.0
Set the status of a particular version of a package.	

`<message name="LexSOAPConnector.SetVersionStatus">`
`  <part name="SID" type="xsd:string"/>`
`  <part name="PackID" type="xsd:string"/>`
`  <part name="VersionNumber" type="xsd:unsignedInt"/>`
`  <part name="VersionStatus" type="xsd:string"/>`
`</message>`
	`Session ID`
	`Package ID`
	`Version Number`
	`Version Status`

`<message name="LexSOAPConnector.SetVersionStatusResponse">` `</message>`	
Version status can be one of: • *draft* • *review* • *final* • *published* • *obsolete*	

GetPackageTypes()	Available in version 4.0
Provides the list of the custom package types supported by the Lobster server.	

`<message name="LexSOAPConnector.GetPackageTypes">`
`  <part name="SID" type="xsd:string"/>`
`</message>`
	`Session ID`

`<message name="LexSOAPConnector.GetPackageTypesResponse">`	

```
   <part name="Result" type="xsd:string"/>
</message>                                                    XML Result
```

This is an example of the XML format returned:

```
<packtypes>
   <packtype  code="course"  preview="glove"  aggregationlevel="course"  deliv-
ery="glove" icon="" name=""/>
   <packtype code="resource" preview="asset" aggregationlevel="resource" deliv-
ery="asset" icon="" name=""/>
   <packtype  code="lo"  preview="asset"  aggregationlevel="resource"  deliv-
ery="asset" icon="" name=""/>
   <packtype  code="qti"  preview="asset"  aggregationlevel="resource"  deliv-
ery="asset" icon="" name=""/>
   <packtype  code="wctcourse"  preview="no"  aggregationlevel="course"  deliv-
ery="no" icon="" name=""/>
   <packtype code="mycustomtype" preview="no" aggregationlevel="resource" deliv-
ery="glove" icon="customtype.gif" name="My Custom Type"/>
</packtypes>
```

Please note that the first five packtypes are the standard Learn eXact package types and they are always present.

FindPackageInfo2() Available in version 4.0

From Lobster 4.0 substitutes FindPackage() and FindPackageInfo() to performs XPath find operation.

```
<message name="LexSOAPConnector.FindPackageInfo2">
   <part name="SID" type="xsd:string"/>                  Session ID
   <part name="Query" type="xsd:string"/>                XPath Query
   <part name="PackTypes" type="xsd:string"/>            Package type list
   <part name="Start" type="xsd:unsignedInt"/>           Starting element
   <part name="Size" type="xsd:unsignedInt"/>            Result page size
</message>
```

```
<message
   name="LexSOAPConnector.FindPackageInfo2Response">
   <part name="Result" type="xsd:string"/>                  XML Result
</message>
```

*Details on the usage of this method are available in the "**Details on the query procedure**" section.*

2.9.2 Error Codes

The table below describes the errors codes returned by method calls:

ERROR CODE	HEX VALUE	DESCRIPTION
E_FAIL	0x80004005	Generic exception
E_INVALIDSESSION	0x80050001	Invalid session ID
E_ALREADYEXISTS	0x80050002	The element already exists
E_DONOTEXISTS	0x80050003	The element does not exists
E_PASSWORD	0x80050004	Invalid password
E_USERNOTFOUND	0x80050005	User not found
E_PERMISSION	0x80050006	User does not have permissions
E_INVALIDXML	0x80050007	Invalid XML file
E_NOOPERATION	0x80050008	No operation requested
E_FILENOTEXISTS	0x80050009	File does not exists
E_MANIFNOTEXISTS	0x8005000A	Manifest file does not exists
E_INVALIDCOURSEID	0x8005000B	Package/Course ID is not valid
E_LOCKED	0x8005000C	Package is locked
E_TOOMANYUSERS	0x8005000D	Too many users connected
E_USERALREADYLOGGED	0x8005000E	User already logged
E_UNZIPPING	0x8005000F	Error during PIF unzipping

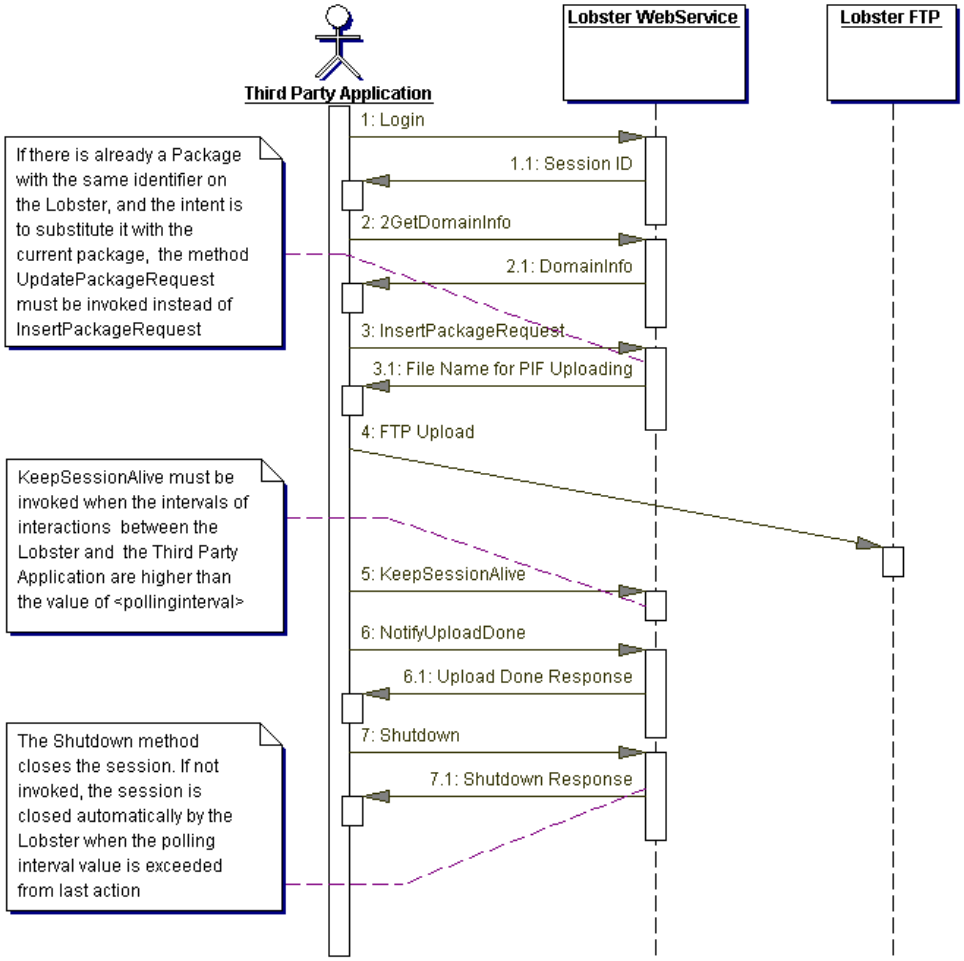
E_DOMAINNOTFOUND	0x80050A01	Domain not found
E_SESSIONNOTOPENED	0x80050A10	Session not opened
E_INVALIDPACKID	0x80050A16	Invalid Package ID format

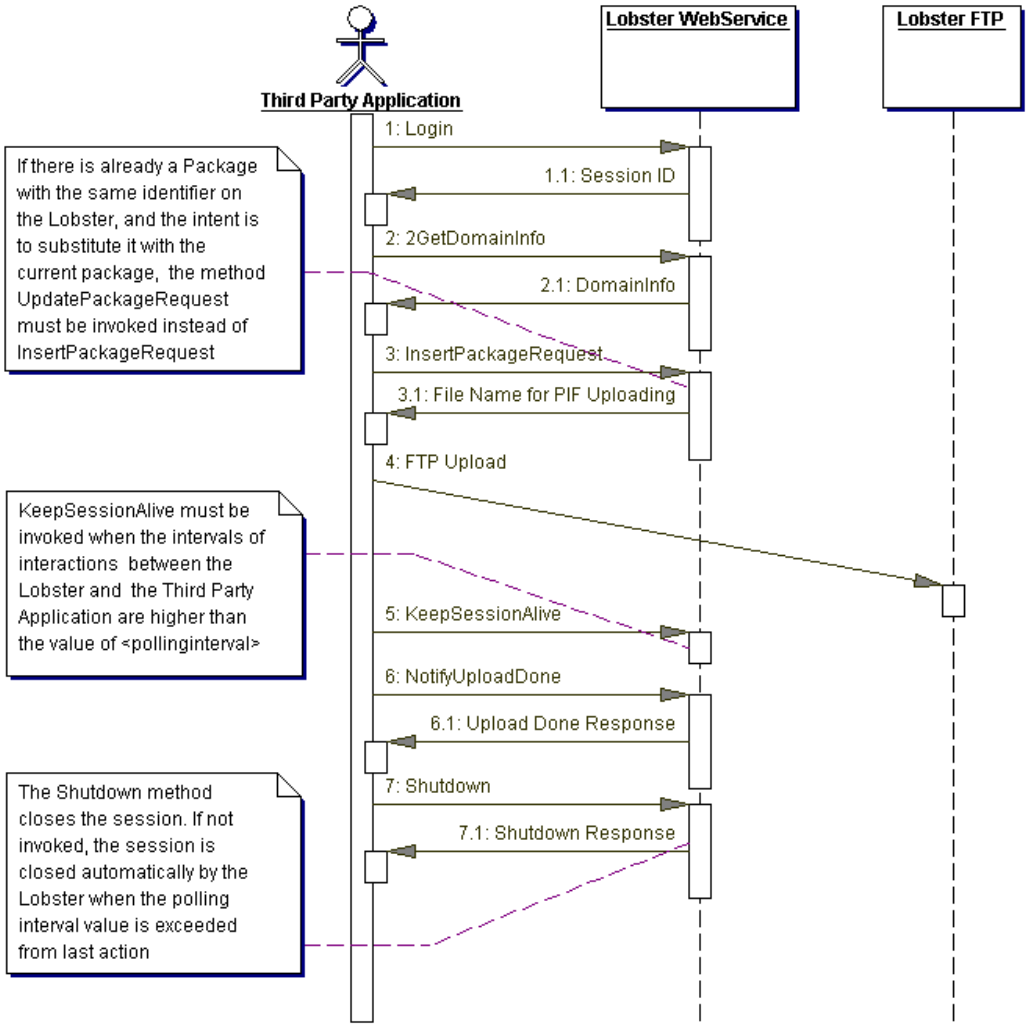
2.10 Publishing Procedure

The present paragraphs describes, step by step, the procedure that a third party application has to follow to publish (insert or udpate) a package into Lobster. Packages are published using FTP protocol.

1. Invoke Login() in a Lobster Domain with valid authentication data and receive a LoginResponse Message
2. Invoke GetDomainInfo() in order to obtain FTP authentication info and receive a GetDomainInfoResponse Message
3. Invoke either InsertPackageRequest() or UpdatePackageRequest() for an Insert or an Update operation. In both cases the identifier of the package to be uploaded is requested.
   NOTE: Each package (Course, LearningObject, Asset) published on Lobster, has a unique identifier inside the repository: this identifier is represented by the IMS manifest identifier attribute
4. In case of new package insertion, call SetInsertOptions() in order to specify if the material shout be treated as a public access material, or to insert the material into a particular Working Area. By default, new material are private and are not enrolled into a Working Area. Use GetWAList() in order to obtain a list of the Working Areas where the user is enrolled.
5. Perform an FTP transfer or an HTTP(S) transfer of the PIF file.If you choose FTP, use FTP authentication info provided by GetDomainInfo() and upload a file with the name provided by InsertPackageRequest() or UpdatePackageRequest().
   NOTE: The PIF file must be in Zip archive format. Call KeepSessionAlive() periodically (the period is provided by GetDomainInfo) if the file transfer is longer than the session timeout (usually about 120 seconds)
6. Invoke NotifyUploadDone() to notify the Lobster when the transfer is finished and receive a NotifyUploadDoneResponse() Message. This response is an implicit confermation of a successful registration of the package on the repository
7. Invoke Shutdown() to close the connection and receive the ShutdownResponse message.

The following sequence diagram describes the call sequence for a publish operation.


Third Party Application
Lobster WebService
Lobster FTP
1: Login
1.1: Session ID
2: 2GetDomainInfo
2.1: DomainInfo
3: InsertPackageRequest
3.1: File Name for PIF Uploading
4: FTP Upload
5: KeepSessionAlive
6: NotifyUploadDone
6.1: Upload Done Response
7: Shutdown
7.1: Shutdown Response
If there is already a Package with the same identifier on the Lobster, and the intent is to substitute it with the current package, the method UpdatePackageRequest must be invoked instead of InsertPackageRequest
KeepSessionAlive must be invoked when the intervals of interactions between the Lobster and the Third Party Application are higher than the value of <pollinginterval>
The Shutdown method closes the session. If not invoked, the session is closed automatically by the Lobster when the polling interval value is exceeded from last action



2.10.1 Querying and retrieving procedure

To perform a query on Lobster materials, a third party application has to express its query with the Xpath language syntax. The result set is represented in XML format and could be paged. Packages are retrieved using HTTP protocol.

1. Invoke Login() in a Lobster Domain with valid authentication data and receive a LoginResponse Message
2. Invoke GetDomainInfo() in order to obtain some information about content location.
3. Invoke FindPackage() to find the identifier of the packages that satisfy the constraints imposed into the Xpath expression passed as parameter into the FindPackage() method. To locate the package the third party application there are two possibilities to create the URL:

   a) ***domain-url***`+"/"+`***coursebasepath***`+"/cppid/"+`***packageid***`+"/"+`***version***`+"/imsmanifest.xml"`

   - The **`domainurl`** returned by DomainInfo() is an absolute path, a string like http://127.0.0.1:80/domainsample
   - **`coursebasepath`** is returned by GetDomainInfo().
   - "/cppid/" is a fixed string.
   - packageid is the package identifier, returned by the FindPackageInfo().
   - version is the wanted version of the package. It can be a positive integer number or one of:

- "last": request the last version of the package.
- "lastpub": request the last version of the package in the "published" status.
- "lastfinal": request the last version of the package in the "final" status.

- Also, when a specific version number is used, this parameter can be followed by the character "f" in order to force the download of the exact version even if it is marked "obsolete". If "f" is not specified and the requested version is obsolete, a more non obsolete version of the package will be provided by the system.
- "/imsmanifest.xml" can be used to retrieve the imsmanifest file, but can be any other complete file path inside the package.

b) protocol+"//"+host+":"+port+"/"+domainurl+"/"+coursebasepath+"/cppid/"+packageid+"/"+version+"/imsmanifest.xml"

- protocol is either http or https (usually http) and host is the IP of the machine hosting the WebService, port is the port where the transfer protocol service is available.

NOTE These information can be retrieved by the third party application from the url of the wsdl file, e.g. http://127.0.0.1:80/LobsterWebService/Lobster.wsdl

- The domainurl returned by DomainInfo() is an relative path, a string like :80/domainsample
- The rest of the URL is the same as the case a).

4. Perform http(s) transfer of the imsmanifest.xml and possibly (all) the binary files referenced by it.
5. Invoke Shutdown() to close the connection and receive the ShutdownResponse message.

2.10.2 Sample of Query and Result Set in XML Format

XPath Query: `[tf:containsText(general/title/langstring, "Package")]`

FindPackage() response:

```
<?xml version="1.0"?>
<ino:response                              xmlns:ino="http://namespaces.softwareag.com/tamino/response2"
xmlns:xql="http://metalab.unc.edu/xql/" ino:sessionid="162" ino:sessionkey="546882039">
   <ino:message ino:returnvalue="0">
      <ino:messageline>XQuery Request processing</ino:messageline>
   </ino:message>
   <xq:result xmlns:xq="http://namespaces.softwareag.com/tamino/XQuery/result">
      <lexresult         identifier="Package_1"         publicaccess="true"         status="unlocked"
                type="course" format="application/zip" >
         <title>Package 1</title>
         <descr>Package 1 Description</descr>
      </lexresult>
      <lexresult         identifier="Package_2"         publicaccess="true"         status="unlocked"
                type="resource" format="image/jpeg">
         <title>Package 2</title>
         <descr>Package 2 Description</descr>
      </lexresult>
   </xq:result>
   <ino:message ino:returnvalue="0">
      <ino:messageline>XQuery Request processed</ino:messageline>
   </ino:message>
</ino:response
```

NOTE: In relation to the version of Tamino XML Server installed, the response XML format could change.

See APPENDIX A for the whole Lobster wsdl file.

2.11 Design & Development

Aim of the present section is to give a basic idea on how to exploit authoring resources available to derive effective content completed by a good set of metadata. More detailed indications for using all authoring features of *learn eXact* Packager© are provided in the manual. By browsing this section the user should be able to properly understand what to do in respect of course design, content, LO and related metadata. The introductory part is aimed to give some indication about the design, development and usage of courses and LO including also notions on how this is related to the learning activity. The remaining part of the section de-

scribes metadata, related mandatory and recommended fields and suggests what may be considered optional. All is thought in light of computer-supported edutainment and strictly reflects the e-learning model approach. A set of relevant info related / preliminary to the production of effective e-learning objects is provided. In all our description the starting point is the general principle completed by some practical suggestions and hints for preparing efficient and effective content, where metadata play an essential role in the process. Some of these aspects have already been addressed in other documents, namely DE3.1.3 and DE8.2.1, in any case in those documents were primarily addressed general issues while here we focus on the specific instance of the process related to production for kiosks.

2.11.1 Authoring and Content development for e-applications

Producing multimedia applications, especially in the educational field, is a complex, time consuming and costly task. In more detail it is necessary to take into account that even if available content may be extremely rich and valuable for producing learning content, there are plenty of issues related to copyrights and content usability to be taken into account, nominally:

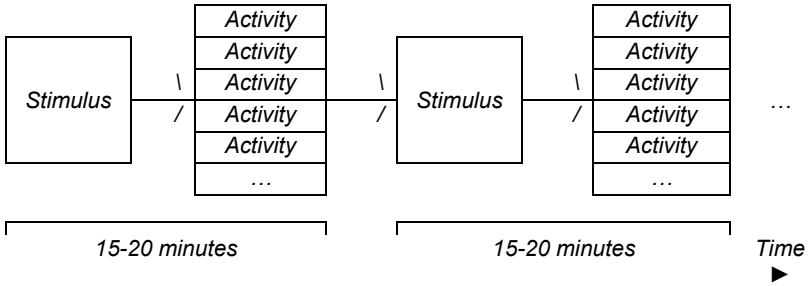
- Images are available but covered by copyrights and the clearance procedure is quite complex;
- Multimedia content (video, audio…) presents even more copyright problems than images;
- Text is available from many sources (from literature to newspapers…) but also in this case there is need for copyrights clearance;
- Content should be suitable for web-based applications (images, audio and video formats…) therefore may need adequate post-processing.

The editorial board following directions of the pedagogues/instructional designers involved in the course design usually performs content selection. The legal department performs copyright clearance once the list of chosen contents has been finalised. These are preliminary steps that apply to the courseware overall design. Then takes place the typical e-learning authoring process, which is structured as follows (please note that in this phase people involved and called authors are usually skilled both in education and programming or at least in the usage of the adopted authoring tool – in other words they are often instructional designers):

- An author begins the development process by choosing from a library of page templates or content wizards or by opening a blank page and selecting the navigation buttons, text placeholders, and other objects from a catalogue. A template contains placeholders for the text and media that an author wants to display and usually contains a common navigation mechanism and background image.
- Next, the author adds text, graphics and other media to the object placeholders in the page templates. All of the object placeholders can be modified by setting properties that define the appearance for each object.
- To create interactive content, an author can build a quiz or test by selecting question objects from a catalogue. The author can enter feedback for each response to a question. The feedback can be text, media, animations or other types of instructional material. The behaviour of a course can be based on the user interactions with test questions.
- Authors can use the preview feature at any time to see the content in action from the perspective of a learner.
- When the content is ready for deployment, authors can publish it.

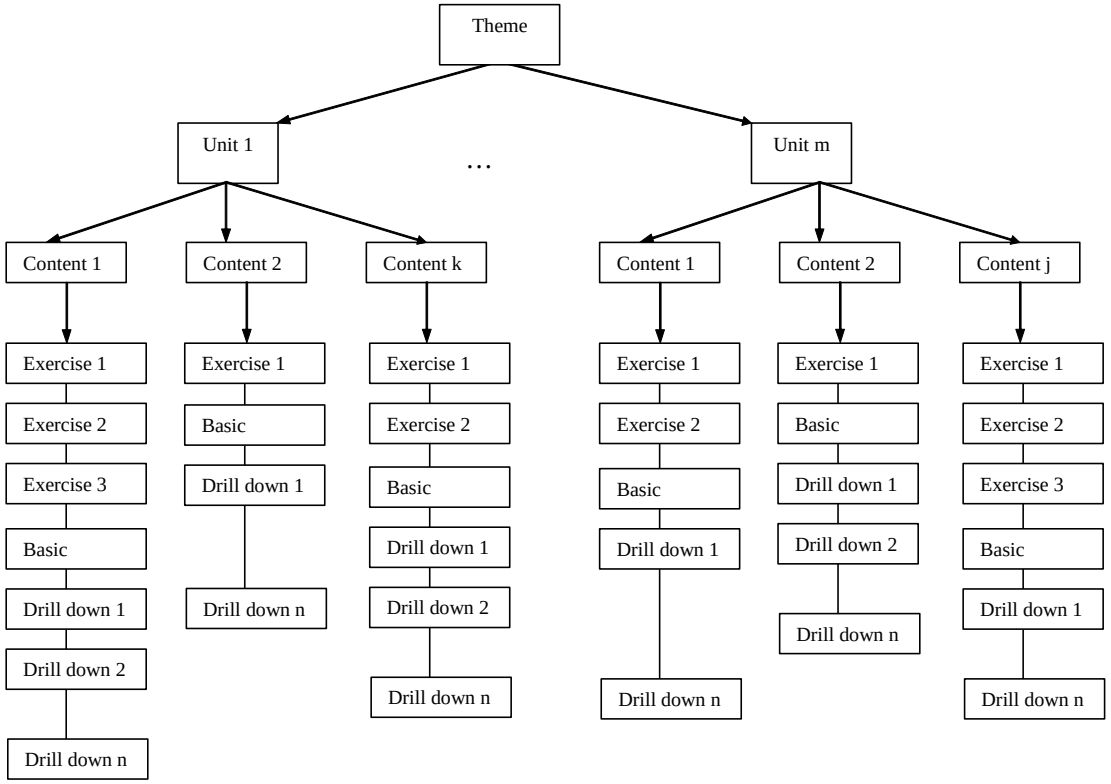
This implies that suitable basic assets are already available and catalogued. Often this is not the case and is necessary to deal with the process of making content available, nominally: *how to transform in learning objects structured and unstructured content*. Also this process is usually performed by personnel usually skilled both in education and programming or at least in the usage of the adopted authoring tool with the specific support of graphics experts… The first step would be to properly select and classify content. Then it will be necessary to take textual content and turn it into smaller self-consistent chunks. During this step glossaries will have to be defined and populated to be available then as complementary / support content. Textual content may present a chapter / paragraph-oriented structure and such sub elements can be either logically related but self confined or logically related and interdependent. In the first case it will be simply necessary to segment text to the smallest self-confined / functional level and transform it into a learning object. In the other case it will be necessary to segment it, structure it and produce a set of interdependent objects (via prerequisites). Finally it will be necessary also to provide relevant metadata and associate it to produced objects in order to grant easiness in the search & retrieve process.

It is extremely important to realise that e-content materials live and die by the quality of their design. Therefore in the e-world, content and design have remained the most critical dimensions of success. Not surprisingly, the shortage of well-designed, engaging and relevant edutainment / e-learning products is still high on the list of reasons for limited acceptance of e-learning.
Even if originally devised for language learning supported by ICT, may studies have proved that the best approach for retaining user attention and increase interest is to alternate stimuli and activities in sessions that last around 15-20 minutes each. This brings to the following structure:



Typical fruition scheme of a content unit

This offers an optimal compromise among feasibility (especially in an hybrid CD/internet based content delivery environment) and sustainability in terms of attractiveness, engagement, effectiveness and efficiency. Moreover this approach allows also a development diluted in time as new content can be produced while available one is just being delivered or used. Each set of stimuli and activities can be organised logically in a sort of treed structure led by a theme and comprising a set of "units". Each unit can have a set of content (usually the one including the stimulus) from which are dependent different kinds of activities: exercise-like to put user at challenge and further stimulate engagement, basic and more in-depth (drill-downs) activities aimed to address different levels of curiosity and engagement. Possibly each content, or component, should be defined as a self consistent and self confined object in the case of a dynamic construction, while it should be inherently compound and structured in case of institutional learning provision (to meet curricula, assessment and standardisation requirements).



Content: a possible logical structuring

2.11.2 LO Design and construction

The process to develop consistent LO is basically quite simple, yet comprises a set of well-defined steps that should be followed to increase overall process efficiency as well as result efficacy. In the present section we will outline this set of steps. The first issue usually is the definition of the learning method to be applied in the process. In our case was chosen JADE (Justification, based on Argument, arising out of Discussion, based on Evidence – developed by Warwick University). The reason for adopting JADE is that it enables an edutainment based learning process that spans over a period of time and encompasses several cycles of knowledge acquisition, assessment and evaluation… bringing also to a refinement in user's ability in managing a learning process. Once defined the logics it is also necessary to define activities structure. This is needed in order to properly describe the concept according to which available content will have to be organised. It is essential to take into account that, following the adopted approach, relevant content may be profitably exploited only if properly organised and split into elementary bricks. Each brick will be self-consistent and stored into the overall system as a learning object (LO) along with its own metadata (LOM). It is expected that such bricks can also be aggregated to provide more refined content (already structured into units). Such units will typically comprise a certain set of contents (basic and drill down information) samples and exercises. Therefore to summarise the previous process we can say that the description of an e-activity should necessarily include the following parts:

- Concept description (how units are linked, which is the rational behind the experiment…)
- User role (simulation performance, practical activity, logical activity…)
- Aims and objectives (what kind of knowledge should be acquired and in relation to what…)
- Activity steps (content that has to be acquired and level of drill down available…)
- Skills exercised (which skills the user should use and eventually reinforce with the activity…)

What follows is an example of how to describe an activity exploiting a "concept map" of the activity - skills practiced through each exploration, possible paths to go through the activity etc. The grid below may be used as an aid to depict one whole activity. What follows is just an example based on art:

		Questions					
		Have styles evolved?	***Who was an artist?***	***Which were the disciplines?***	…	…	…
Skills	***Logic skills***	Understand theme	Understand theme	Understand theme	…	…	…
	Information handling	Discard irrelevant sources	Find old and new definition	Find old and new disciplines	…	…	…
	Graphical skills	Draw an evolution timeline	…	…	…	…	…
	Practical skills	…	…	…	…	…	…
	ICT skills	…	…	…	…	…	…
	Collaboration	…	…	…	…	…	…
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…
	…	…	…	…	…	…	…

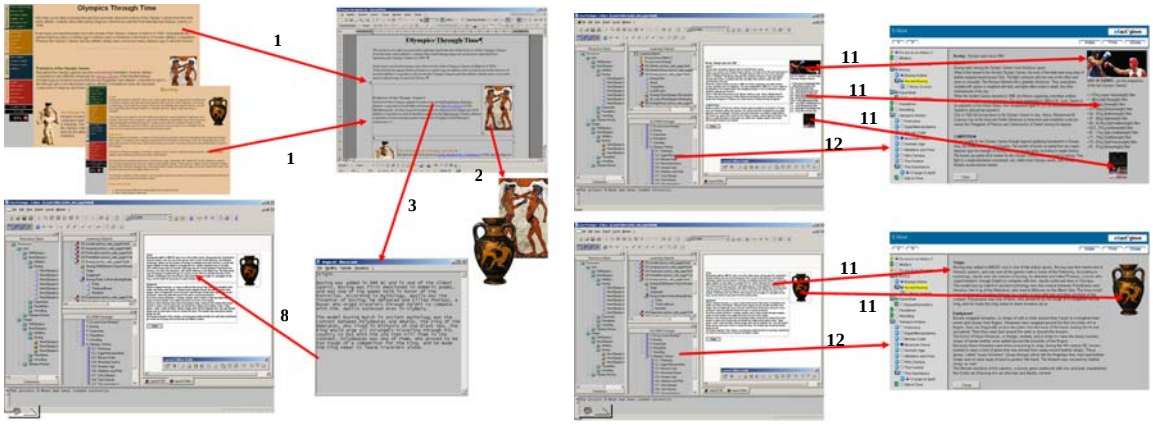
2.11.3 Structuring content in the authoring environment

Content fruition along with knowledge acquisition has implications on how raw contents have to be organised and transformed in order to be profitably usable in the provided authoring system. Once again we will take up examples to support explanation. In the picture is presented a possible content organisation to manage content prepared for course / LOs development. The basic idea behind is to select data from available sources and then organise and structure all according to a simple structure where each item would correspond to a leaf of the classification tree. Then, according to target users and educational aims, output pages overall structure and possible templates should be selected or generated. This is useful also to identify in more detail what will be needed and further refine the content classification and organisation structure. The next step should be the identification of a set of sources to complete the work (taking into account even the eventual need for copyright clearance). The work can be actually organised as follows:

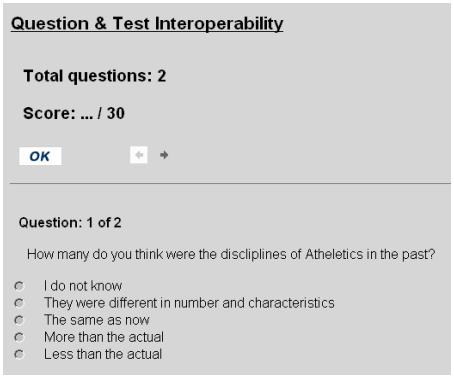
1. Save sources text into chunks (.RTF or :DOC files).
2. Save images in ".JPG" format.
3. Saved text has to be further split into titles, headers, bodies, captions, notes… (.TXT files).
4. File system can be structured in a way similar to the one presented.
5. Files should be stored in the file system following the structure devised.
6. Sketch exercise based on available content.
7. Chosen content organisation structure for the file system should be replicated in terms of logics using the packager (the virtual folder of the Resource store presented in the previous picture)
8. content should be imported in the corresponding area of the logical structure
9. an overall organisation for the content should be sketched.
10. for each devised LO an active web page should be created
11. page layout should be chosen, or designed, in the packager and content associated (eventually producing traceable learning objects)
12. developed LO should be arranged into a package following the devised structure and the corresponding data structure, links, connections.

According to SCORM standards non-accessed content is presented with a white dot, accessed but not completed content is presented with a blue half filled dot and accessed and completed content is presented with a blue filled dot. Usually a SCORM tracked course foresees that the user completes the full unit before being granted the chance to move any further. Moreover if there are exercises for knowledge assessment usually they can be also conditioning the overall navigation procedure. It could be possible to prevent the user from accessing any other section of the content until the present one has been completed and related tests passed. Nevertheless this suits a training course where knowledge has to be gained according to a well-organised path and not a more general edutainment approach and therefore in our case this approach is not feasible. We have to grant the user a higher level of flexibility. Content should be interesting and appealing. Tests should stimulate user curiosity and willingness to access the available content, performing a drill down on the issues that present the higher interest. We suggest that no compulsory path should be devised for browsing the content. Tests should be in form of challenges and should provide feedback but not force any specific constraint

on content access. They should be designed in order to stimulate the user to learn more in case of negative feedback or to gratify in case of positive one. Content should be arranged in terms of drill down to accommodate different level of curiosity that may characterise the user. What just sketched is the suggested process to be performed for setting up a course. In the pictures numbers associated to arrows represent the specific passage in the previously described set of steps.



Importing from sources in the authoring environment and mapping content in the target LO



An example of Test

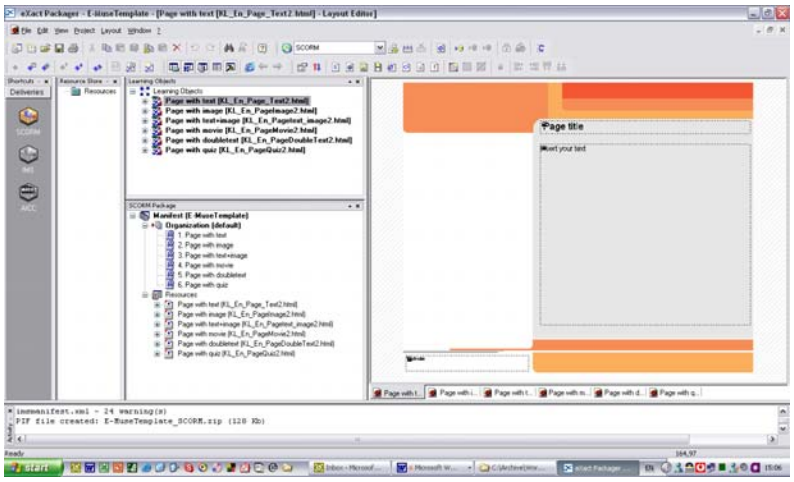
2.11.4 LO Templates

Learning Objects templates have been designed in order to be as general as possible and to comprise basically the following items:

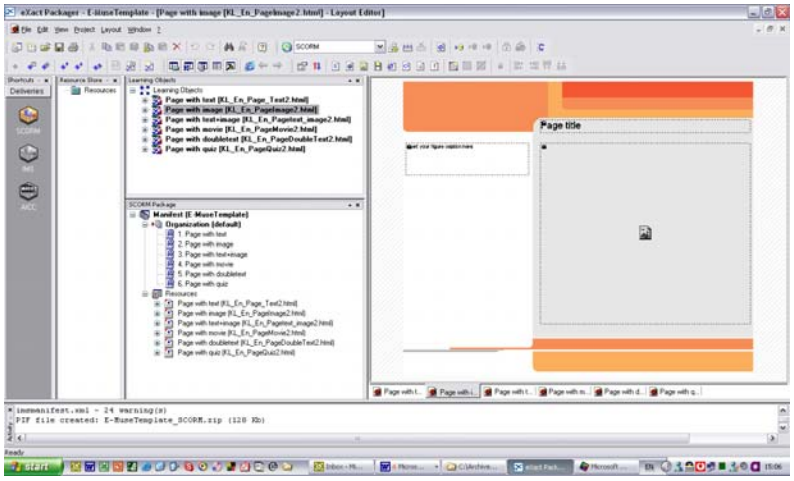
- Text (including title / notes / captions);
- Multimedia in general (image / video / audio);
- Exercises (if necessary).

In more detail the basic LO structure corresponds to a simple CSS2.0 based HTML page. Basically this corresponds to apply a style description (held into the .css file accompanying the .htm page) where absolute positioning and dimensioning of objects represented onto the page are provided. In the following pages are presented the basic expected templates as they appear in the packager while editing the style. In the presented case templates are always empty. They can be enriched and modified according to needs by the content authoring using the packager. The same tool allows also pre-viewing of achieved results and a set of other functionalities like metadata management for selected objects or whole pages. Some example of modified pages is also presented to give a clear idea of tool and templates flexibility. For the sake of completeness and in order to accommodate expected kinds of contents we have been providing the following basic templates:

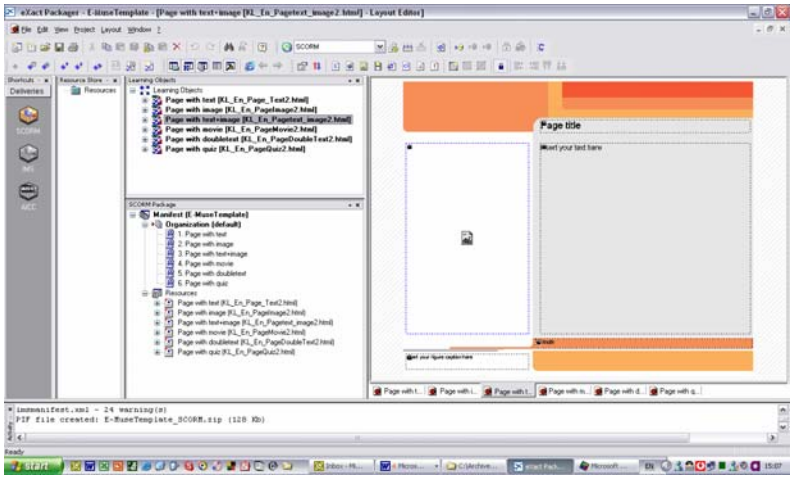
- **Page with text**
  it represents the basic page to present text; it could be used to introduce subsequent content or simply to present textual information;



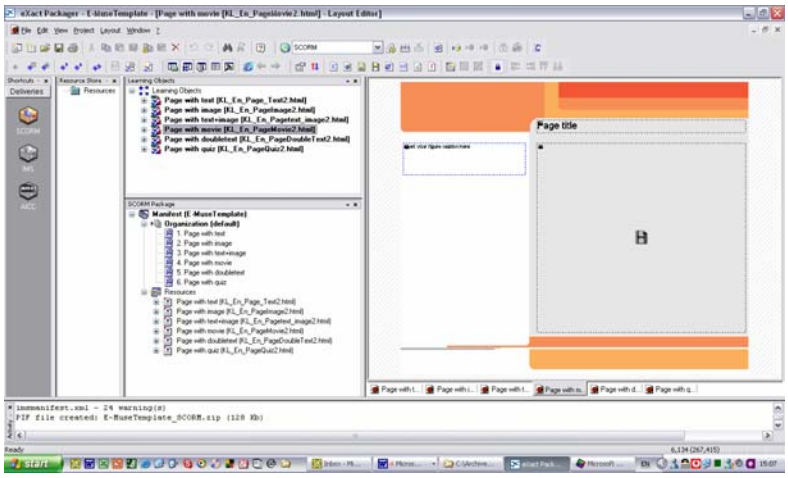
- **Page with image**
  basically it is designed to present a visual content with related caption as a part of something more complex;



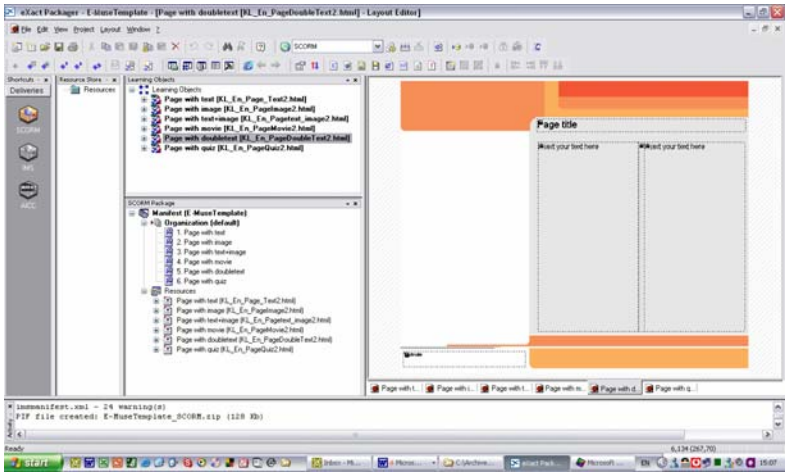
- **Page with text and image**
  similar to previous page holds also space for textual information and therefore can be used to convey a more complete info;



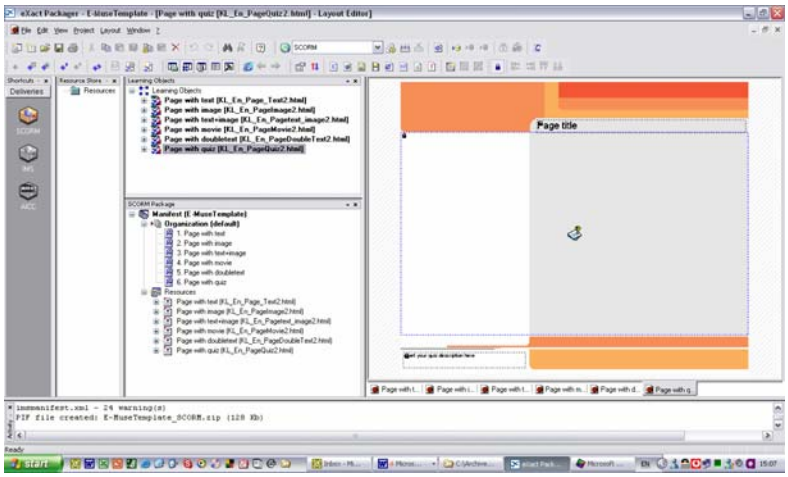
- **Page with movie**
  similar to the page with an image is designed to hold a video file with a basic description (equivalent to a image caption);



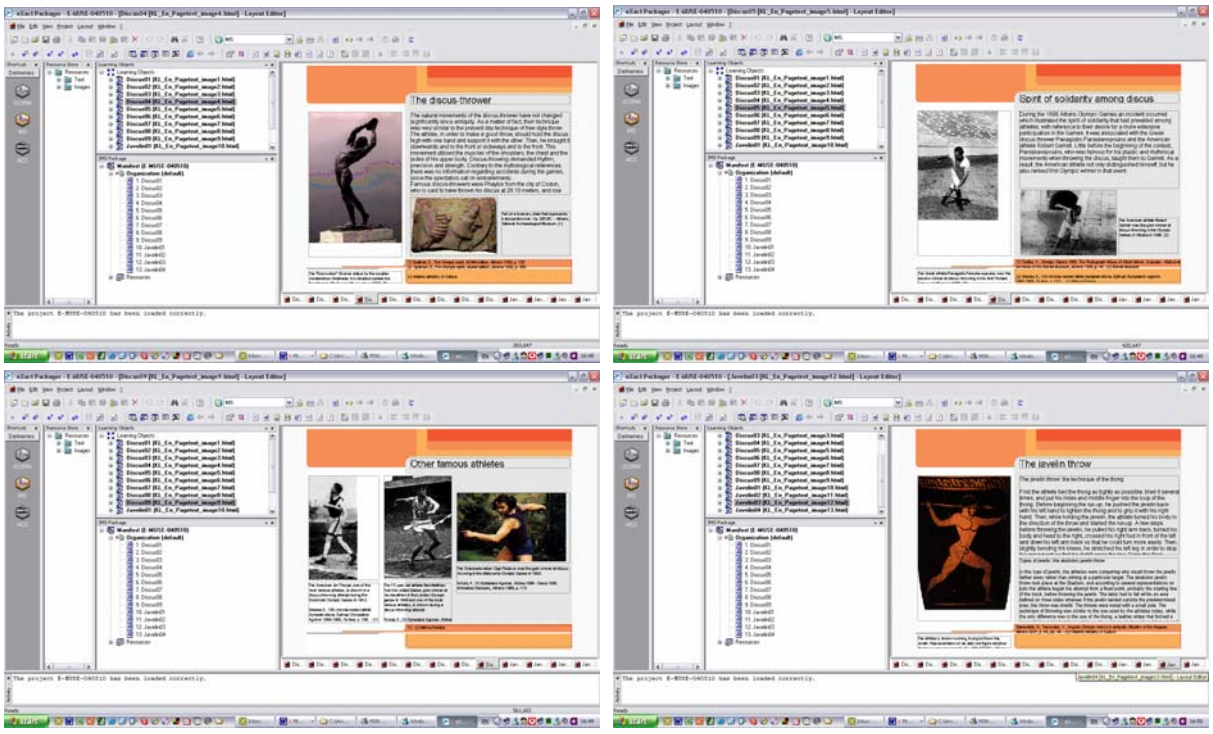
- **Page with double text**
  this page has been designed to hold text in a synoptic way (in our case could be used to present a comparison between actual and ancient rules of a given sport;



- **Page with quiz**
  this page is specifically devoted to the practice activity and would provide the best results when used in a SCORM delivery where tracking is enabled.

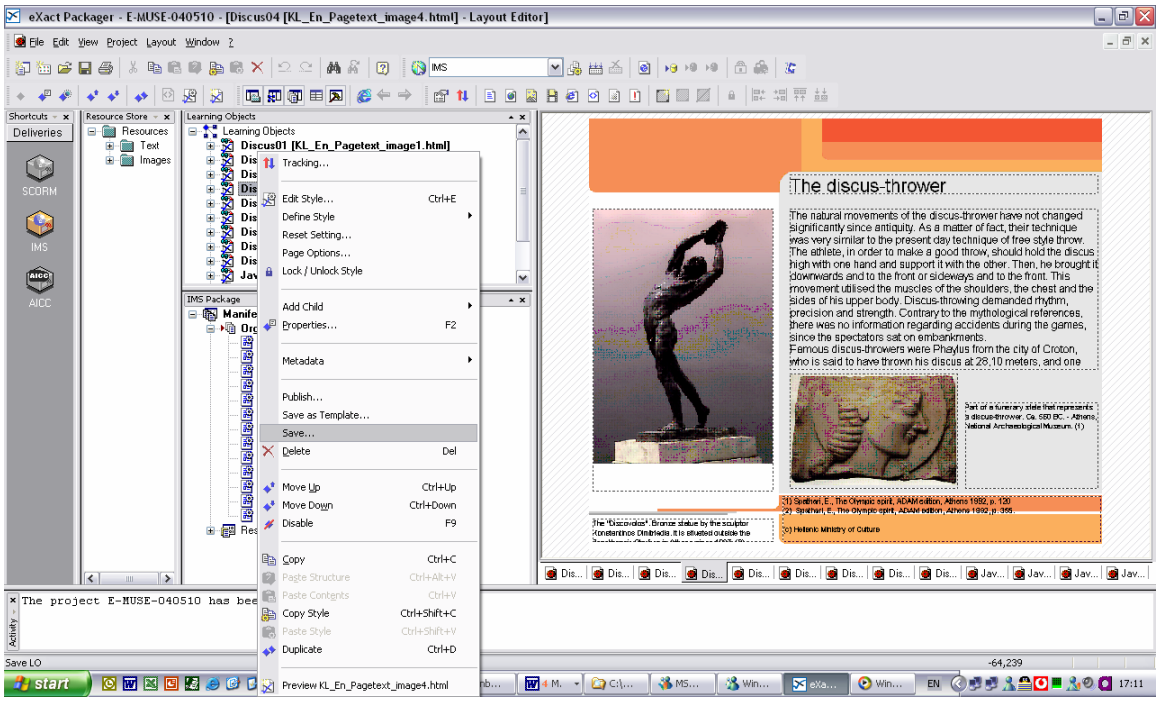


As already mentioned it is possible to derive more complex pages staring from the previously provided templates and then modifying them to fit the author expectations. Some example of such modified pages is reported hereafter to give a clear idea of tool and templates flexibility.



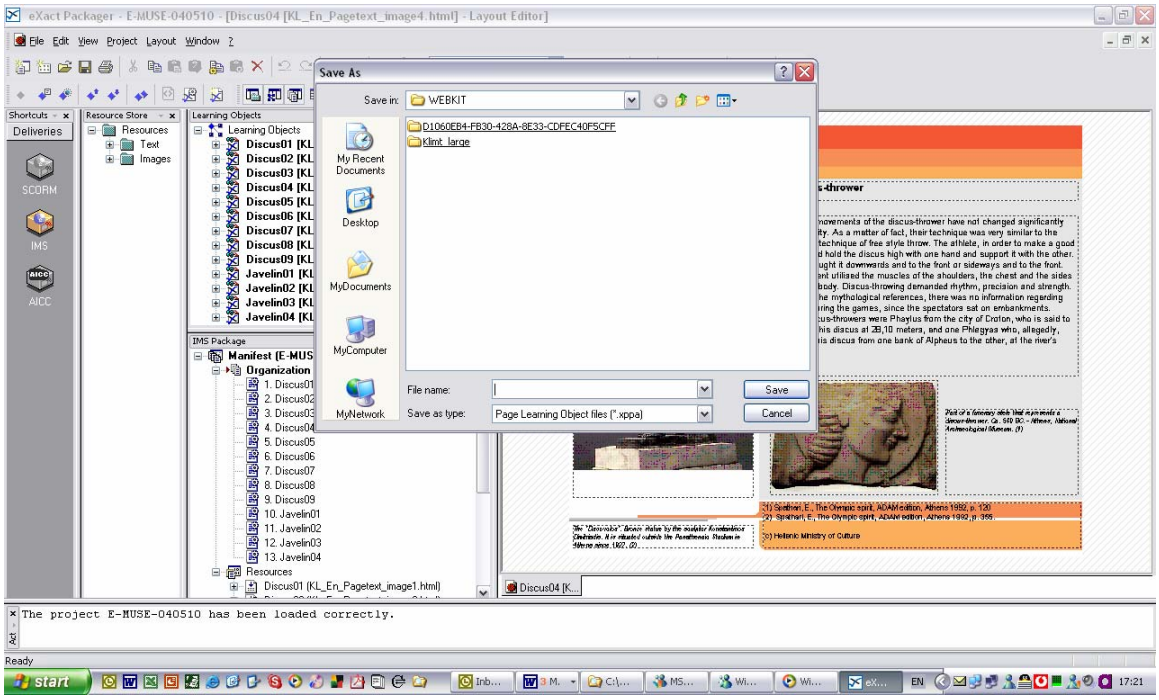
2.12 Export procedure

Is basically aimed at providing authors with the possibility to develop a generic LO using all utilities and tools provided by ***learn eXact Packager*** **©** and then simply export them in a format suitable for free usage either as LO in an other e-learning environment or as simple CSS2.0 based HTML pages. Once a LO has been created either form a template or from scratch, the author should select the LO from the LO list and right click on it. A pull down menu will appear (like in the following image):



Export procedure step 1: selecting the object to export and recalling the menu.

The author shall now select the “Save” option from the pull-down menu appearing on the screen. Once this step is performed the following dialogue will appear:



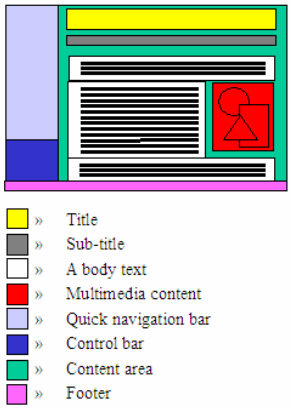
Export procedure step 2: selecting the target for exporting.

The author should choose the desired location, insert the destination file name and then press save. The procedure will then generate a “.xppa” file holding all the required data to allow proper management of the saved object. The object obtained can now be exchanged for processing in another IMS packaging compliant environment by being previously loaded in the LCMS and then in the authoring environment.

2.12.1 Template definition and Format adoption

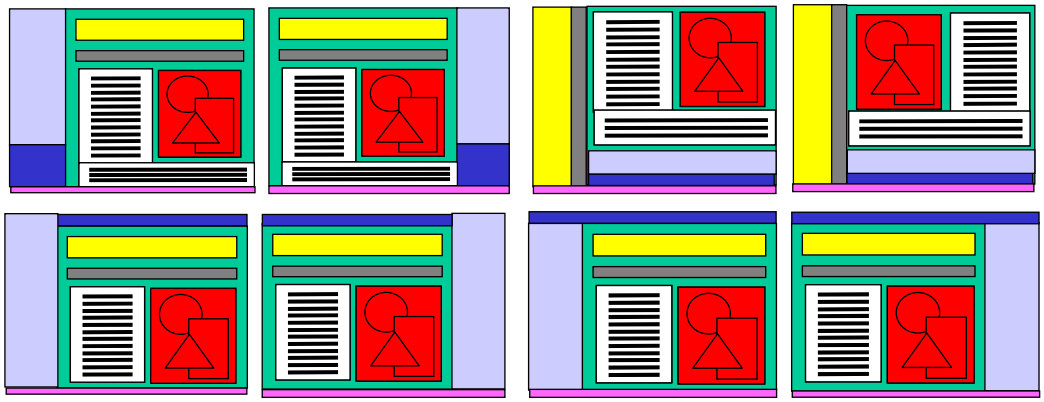
In our environment is possible to easily transform a content layout into a template. All that is needed is to select the desired LO in the Learning Object window of the Packager, right click on it, select the "Save as Template" option of the pop-up menu and all is done. Next time the newly generated template will be available into the "Custom" tab of the "Add New Learning Object" dialogue. This offers a high degree of flexibility to the author as it is possible to build on the basis of what already done, modify and save back generating templates that reflect the need of the moment/customer.

Basically we have to take into account that each publisher has developed a well-defined company image based on a highly studied combination of page layout, colours, type-fonts, styles… this is usually reflected in the defined templates (just as mentioned before). In some cases the same publisher may have different editorial lines each with a well-defined image reflecting the expectations of the consumers to which it is addressed. In traditional book/magazine publishing it is usual to have series devoted to children (colourful, with big fonts and plenty of drawings, thick paper…) and others for scholars (very dim colours, tiny and packed fonts, thin paper…). The same applies to the e-world where paper is replaced by the screen of a device (playstation for youngsters, smartphones for professionals, PCs for both…). In both cases there is a "grey" area of superposition that sees content and formats that are generically acceptable to the widest possible public. This is actually the segment we are referring to, as this is the one best suitable at the moment for automatic composition and adaptation of content. A typical content page holds a set of content comprising: title, sub-title (if needed), body text (resizable and scrollable in respect to display area), some multimedia content (with related controls when needed), a quick navigation bar (if needed to allow content browsing) and potentially comprising or associated with a control bar (to allow overall content management: print, save…) and a footer usually holding copyright information. The usual structure is the one in the picture aside. What just mentioned could be synthetically expressed by the following formula:



$$\exists!\ \text{Title} + \{\text{sub-title}\} + \{\text{body text} + \{\text{scroll bar}\}\} + \{\text{multimedia content} + \{[\text{control bar}]\}\} + \{\text{quick navigation menu} + \{\text{control bar}\}\} + \{\text{control bar}\} + \text{Footer}$$

Given the previously reported page structure, the typical content page layout could be one among the following:

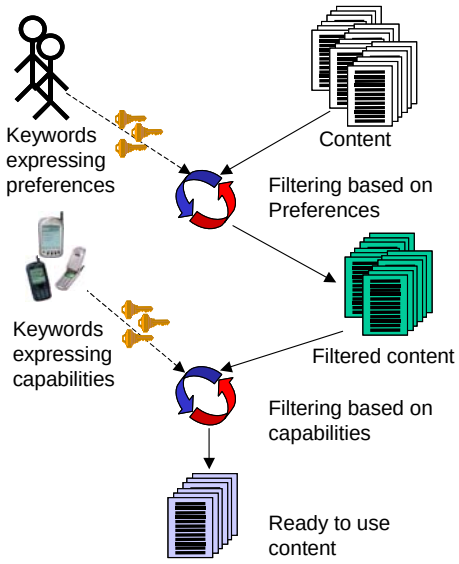


Some of the most typical multimedia page layout used in the e-world

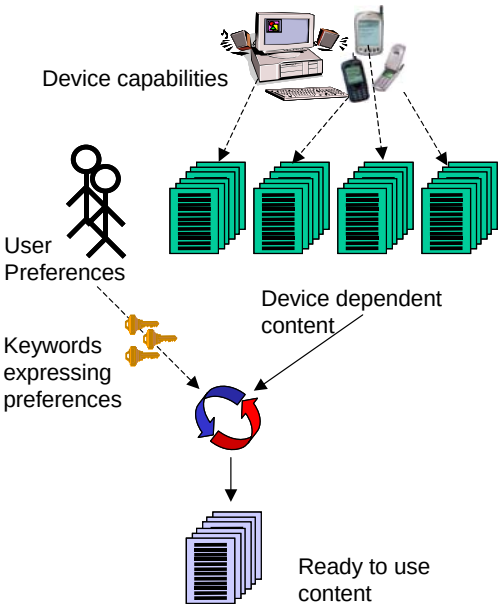
2.12.2 Content combination and adaptation

In the e-publishing world this issue has been widely addressed and may different solutions adopted, yet there is still a lot of discussion on which is the best way to follow. There are publishing tools (mainly the one used for newspapers and magazines) that follow a distributed approach combined with a workflow management system to replicate in virtual the usual production process (therefore there are "baskets" for "in", "out", "transit", "wired-news", "wired-pictures"… content).

In other environments like book-oriented publishing houses there is a "library" approach where classification of content is the primary issue; any new idea that should be turn out into a product will have to be adapted on a case by case basis following the most suitable editorial line… In the multimedia environment the approach is basically a combination of the two. It is still partially lacking a standardized classification approach but some solutions are presently emerging mainly in those environment where the huge amount of content already produced and wide target market generates enough volumes to justify the costs related to a proper formalization of a classification and a related archival structuring to enable efficient and effective content reuse. In a very similar manner the issue of content delivery to end-user respecting user-preferences (including delivery format and device) has been tackled several time and under many points of view. If the paper-based solutions are well established, not so much is possible to say about multimedia content. Being more specific we should distinguish some categories of multimedia content as each of them has been addressed in a different manner.



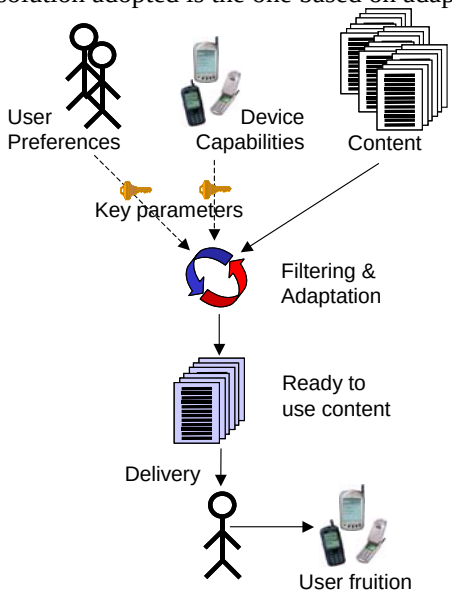
Two step filtering



Device dependent content production

For example music has foreseen a set of different format of mass distributions prior to the advent of MP3, nominally vinyl, audiocassette and CD-ROM; in a similar manner video was initially distributed in filmed format only, then thanks to BETAMAX in video-tape that then moved to VHS, HVHS and DVD, now live streaming is possible and already used. These are just examples needed to explain why certain publishing houses have adopted certain solutions. For example, in the filed of mass distribution of combined (text + multimedia) content, most of the publishers provide the user with the possibility to chose whether to have audiocassette, VHS, CD or DVD support, this at least as far as some market segment are still relevant enough, then progressively they will drop out the older support media to move towards the new one and this is basically done on the basis of Nielsen data. In the area of content for mobiles a very similar approach is emerging. Some telecom operators started to offer WAP services and device dependent games; then service companies offered a wider set of customized data, now the offer is wider even if limited to some segments of content. It is yet interesting to note that still there are several approaches for managing the combination of possibilities coming form the combination of user-preferences and delivery-device-constraints. Usually the process is based on either differentiated content production or on selection process based on some way to express user preferences and device capabilities (usually via set of keywords). Publishers often prefer to have device dependent version of content to simply perform preference related adaptation to content as all is demanded at production level and the user simply has to select the desired format for the chosen content. This approach is certainly more time consuming and costly, yet the most diffused (it is reported in principle in the picture aside). In this case device dependent content is prepared and stored in a structure reflecting device dependencies, this latter stage is made for making it easier to the user to proper access to the desired content (wap …) as usually such services are on a "pay per use" basis and the customers have signed a specific contract comprising an agreement on the quality of service (QoS). Then device dependent content is selected matching user preferences. It is worth mentioning that usually the content avail-

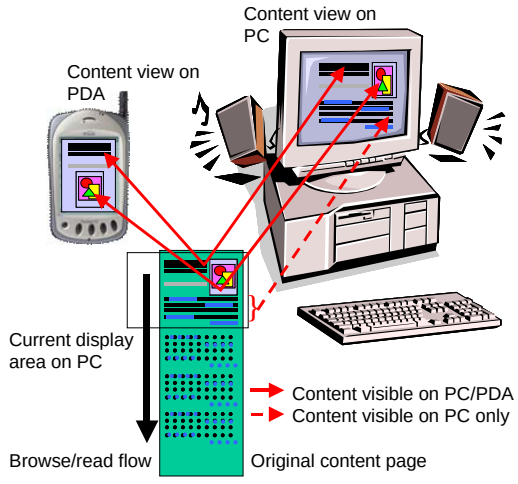
able to the user following this approach is less than the one possibly available in the overall. An alternative solution adopted is the one based on adaptation via filtering.



The ideal process

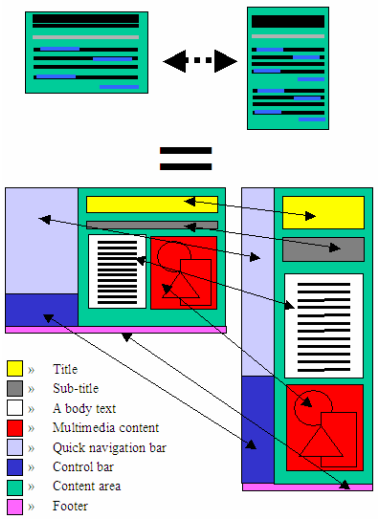
In essence the process changes slightly from the just mentioned one as in this case both user preferences and device dependencies are used as keywords for performing a two consecutive filtering steps as described in the picture aside. Actually, if possible, would be far better if the ready to use content could be derived directly from the original content simply via filtering (on the basis of user preferences) and adapting (on the basis of device dependencies). This is actually one of the scopes of the project and of the present demonstrator. The aside diagram, where the same terms of the previous ones are used for consistency, could represent the desired process. In such an approach is foreseen a single step where are combined filtering and adaptation. To make this possible is essential to examine the issues that are at the back of the device dependencies and that have direct impacts on the content production process. What will be presented here is also based on what exposed in other deliverables listed in the reference section of the document and dealing with content production and selection.

We have already explained that typically content developers have preferred to develop "ad hoc" content for each device. This was, and often still is, basically due to a fact: *it is far easier to manually optimize the rendering aspect of content rather than making it generically adaptable via a programming approach*. This assumption may seem quite in counter position with the usual logics of production optimization where *"ad hoc"* solutions are usually avoided as more demanding in terms of resources. Actually the possibility to fine tune a content that has a high degree of probability to remain stable on a reasonably long time span (basically comparable with the object expected life time), makes it more convenient to pursue the just described approach that leads to an higher quality results (as purposely designed and optimized) that could be also more easily distributed using already existing distribution channels. This would still fit if technology evolution was still proceeding at the speed it had at the beginning of the digital world retaining the high related costs of those times. Present evolution and constant miniaturization trend has somehow reshuffled all this. The average low-end device of a new generation is now far more capable of the high-end device of the previous generation and a new generation of product will be available in around 6 months. This means that in 5 years time is to have had more than 10 generation of devices that would possibly make possible and convenient a "device-independent" content production. Having said so let's focus on present issues related to constraints and changes imposed by format adaptation. It is worth mentioning as the first thing that when taking into exams the two extremes of the device panorama we have wide-screen computers on a side (with a typical landscape 16:9 display ratio) and PDA or smart-phones on the other (with portrait 4:3 display ration).

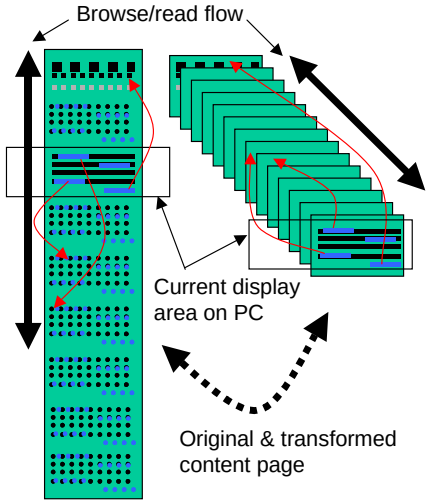


Therefore the most frequent and essential change in content rendering is the orientation. Unless the PDA is able to automatically flip the screen (that is presenting data in landscape rather than portrait) such a transformation has to be done prior to delivery. Now this is not the only point as passing from PCs to PDA, the change in display size implies:

- Lower display area
- Lower computational power
- Lower image quality
- Smaller fonts
- Worse anti-aliasing
- Scroll-bars / page tabs



Page adaptation in page conversion



Navigation structure change due to adaptation

- Different content distribution

What just stated implies that a page that has been designed for fruition on a PC when presented onto a PDA may undergo a set of changes. The first, and probably the more apparent is that content is highly probable to be shrunk to a smaller dimension, as even the simplest of all apparent changes occurring (that is converting from landscape to portrait or vice versa) implies a change in:

- areas sizes (to retain the same content) and
- location of some specific content.

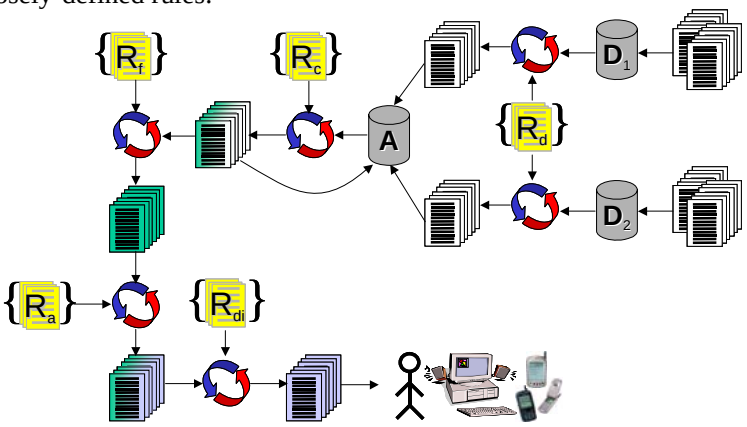
This latter point may sound strange, but given the fact that the most diffused browser for Windows CE / Windows Mobile equipped PDA does not support CSS files a formatted HTML page will lose most of what foreseen at content design and production time.
This, in some contexts, where content quality or publisher / distributor branding is highly related to content quality may be a major limiting factor that may lead to the voluntary production of device customized content production.
Moreover at rendering time (either at server or client side) of a content has to be taken into account that passing from a single page to a set of pages, links between portions of the same page (anchors) are turned into links between pages (URLs), including the return to "top of page" as the way content is browsed changes from:

- Scrolling to
- Paging

The same applies vice-versa. This process if performed manually is long, time consuming and potentially error prone as it could lead to programming problems like "missing links"… while if performed automatically requires a quite complex environment.

In the kiosk factory all this will be made possible thanks to the combination of the solution initially introduced in our editing environment with the procedures and tools of AXMEDIS. In more detail this will be achieved as follows thanks to sets of purposely-defined rules:

- Content component identification and description,
- Content composition,
- Content formatting,
- Content adaptation and eventually
- Content distribution.



The kiosk content processing operation schema

Such a schematic approach could be better detailed as follows taking into account that each set of defined rules has a specific purpose and should be used consistently. The approach is based on what presently developed for

handling IMS/SCORM/AICC content delivery in LEX. Original content belonging to different locations (D_x) is processed with the support of a set of descriptive rules $\{R_d\}$ that enable deriving an "*enriched*" content holding info about single content components typology and characteristics as simplified in the example reported here where the extended description format can therefore look somehow similar to what follows depending on the set of extended information provided[8].

```
<COMPONENT type= title LANGUAGE=en …>
        <ID=1> (insert the XML coding for additional info on object ID) </ID>
        <CONTENT=…> (insert the XML coding for the content) </CONTENT>
        <POSITION=…> (insert the XML coding for the position) </POSITION>
        <ORIENTATION=90dl …> (insert the XML coding for additional orientation info) </ORIENTATION>
        ...
        (insert the XML coding for additional descriptive and format related info)
        ...
</COMPONENT>
```

Having done so the new content is suitable for storage into the AXMEDIS database (A) from where it can be extracted for composition and aggregation. Such operation will be supported by a set of composition rules $\{R_c\}$ that can benefit from the "enriched" description format reported before. Is given for granted that newly produced, aggregated or composed content will be stored back in the AXMEDIS databases either as a new version of the same object or as a new object depending on the kind of operation performed onto it. A similar approach applies for subsequent formatting and adaptation steps where the $\{R_f\}$ and $\{R_a\}$ sets of rules will be respectively used to achieve the desired result. The last step is the distribution one where a last set of rules $\{R_{di}\}$ will be used to apply constraints, limitation or other peculiarities needed to properly handle the process. As previously said this is based on what presently achieved in LEX to handle a similar procedure related, so far only, to learning objects. The following in an example of XML code representing what presently possible and implemented in our environment following IMS standard and that will therefore represent for us the starting point for future developments in the field.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- XML file generated by eXact Packager -->
<!-- eXact Packager licensed to: monbile2.giuntilabs.com -->
<LOMobile xp:deliverytype="SCORM" xlink:label="L5414" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xp="http://www.giuntilabs.com/exact/xp_v1d0" href="LOMobile.html" xp:description="L06_Madonna_pomegranate" Tracking="AICC-
HACP" id="L06_Madonna_pomegranate">
    <style xmlns="http://www.giuntilabs.com/exact/xp_v1d0">
        <colors c1="#000000" c2="#FFFFFF" c3="#D4D4D6" c4="#005FAE" c5="#003060" c6="#01D4FA"/>
        <linkcolors normal="#0000FF" visited="#800080" active="#FF0000"/>
        <tracking>AICC-HACP</tracking>
        <pagesize width="760" height="640" type="1"/>
        <textstyles>
            <textstyle type="Title">
                <normal fontface="Arial" fontsize="28" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="28" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="28" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="Subtitle">
                <normal fontface="Arial" fontsize="24" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="24" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="24" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="Subsubtitle">
                <normal fontface="Arial" fontsize="20" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="20" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="20" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="TextIntro">
                <normal fontface="Arial" fontsize="10" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="10" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="10" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="TextBody">
                <normal fontface="Arial" fontsize="12" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="12" color="#000000" bold="false" italic="true" underline="false"/>
```

[8] The final grammar represents the work in hand

```
                <moreemph fontface="Arial" fontsize="12" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="TextEnd">
                <normal fontface="Arial" fontsize="10" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="10" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="10" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="References">
                <normal fontface="Arial" fontsize="8" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="8" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="8" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="Fig-caption">
                <normal fontface="Arial" fontsize="8" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="8" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="8" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="Footnote">
                <normal fontface="Arial" fontsize="8" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="8" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="8" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
            <textstyle type="Button Text">
                <normal fontface="Arial" fontsize="8" color="#000000" bold="false" italic="false" underline="false"/>
                <emph fontface="Arial" fontsize="8" color="#000000" bold="false" italic="true" underline="false"/>
                <moreemph fontface="Arial" fontsize="8" color="#000000" bold="true" italic="false" underline="false"/>
            </textstyle>
        </textstyles>
    </style>
    <Page xp:deliverytype="SCORM" xlink:label="L5415" xp:description="S1" id="Page01">
        <SubPages xp:deliverytype="SCORM" xlink:label="L5416" xp:description="Introduction" kind="R">
            <SubPage xp:deliverytype="SCORM" xlink:label="L5417" xp:description="Introduction_text">
                <ElementGroup xp:deliverytype="SCORM" xlink:label="L5418" xp:description="introduction_text_en" language="en">
                    <ElementGroup xp:deliverytype="SCORM" xlink:label="L5419" xp:description="Introduction_text_a" type="Tourist">
                        <Text xlink:label="L5420">Welcome. In this section you will discover one of Botticelli most beautiful masterpieces
shown in the Galleria degli Uffizi, the painting Madonna of the Pomegranate.</Text>
                        <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_en_tourist.wav"/>
                    </ElementGroup>
                    <ElementGroup xp:deliverytype="SCORM" xlink:label="L5421" xp:description="Introduction_text_b" type="Student">
                        <Text xlink:label="L5422">Welcome.  In this section you will discover one of Botticelli most beautiful masterpieces
shown in the Galleria degli Uffizi, the painting Madonna of the Pomegranate. Though it you will learn more about the painting.</Text>
                        <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_en_student.wav"/>
                    </ElementGroup>
                    <ElementGroup xp:deliverytype="SCORM" xlink:label="L5423" xp:description="Introduction_text_c" type="Teacher">
                        <Text xlink:label="L5424">Welcome. In this section you will discover one of Botticelli most beautiful masterpieces
shown in the Galleria degli Uffizi, the painting Madonna of the Pomegranate. Though it you will learn more about the painting and its mythological,
philosophical and literary background.</Text>
                        <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_en_teacher.wav"/>
                    </ElementGroup>
                </ElementGroup>
                <ElementGroup xp:deliverytype="SCORM" xlink:label="L5425" xp:description="introduction_text_it" language="it">
                    <ElementGroup xp:deliverytype="SCORM" xlink:label="L5426" xp:description="Introduction_text_a" type="Tourist">
                        <Text xlink:label="L5427">Benvenuti. In questa sezione scoprirete uno dei capolavori del Botticelli presente nella Galleria
degli Uffizi la Madonna della melagrana.<br/>
                        </Text>
                        <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_it_tourist.wav"/>
                    </ElementGroup>
                    <ElementGroup xp:deliverytype="SCORM" xlink:label="L5428" xp:description="Introduction_text_b" type="Student">
                        <Text xlink:label="L5429">Benvenuti. In questa sezione scoprirete uno dei capolavori del Botticelli presente nella Galleria
degli Uffizi la Madonna della melagrana.<b> </b>Potrete scoprire i segreti del dipinto.<br/>
                        </Text>
                        <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_it_student.wav"/>
                    </ElementGroup>
                    <ElementGroup xp:deliverytype="SCORM" xlink:label="L5430" xp:description="Introduction_text_c" type="Teacher">
                        <Text xlink:label="L5431">Benvenuti. In questa sezione scoprirete uno dei capolavori del Botticelli presente nella Galleria
degli Uffizi la Madonna della melagrana.<b> </b>Potrete scoprire i segreti del dipinto e i suoi legami con la storia, la filosofia e la letteratura del pe-
riodo.<br/>
                        </Text>
                        <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_it_teacher.wav"/>
                    </ElementGroup>
                </ElementGroup>
            </SubPage>
        </SubPages>
    </Page>
```

```
</LOMobile>
<!-- End of XML file generated by eXact Packager  -->
```

2.12.3 Authoring & Language related design

So far we have focussed on templates and way to handle content in an easily adaptable format, this is already very useful yet still limiting as nothing is mentioned about creation and usage of the same content from different groups of people that may also have different sets of languages. This problem has been taken into account since long in the publishing environment yet is still at its dawn in the e-world. Traditional publishing tended to solve this issue through co-editions in which publishers belonging to the different linguistic environment would edit the same content in their own language adapting images and any other multimedia content to the needs. This would result in a set of titles bearing in essence the same original copyright (for the original version) and a set of new ones for each language. If the passage from language "A" to language "J" would require a passage from language "C" this would be addressed with specific contracts and in the edition "J" will be mentioned both "A" and "C" ©. In the context of e-learning we have addressed the problem exploiting the possibilities offered by the IMS standard format for content package description (the manifest) and in AXMEDIS we will be further exploiting this approach possibly extending it to a new level of completeness. What follows is an example of how to handle Multilanguage content in the same object using a simple structure.

```
<Page xp:deliverytype="SCORM" xlink:label="L5415" xp:description="S1" id="Page01">
    <SubPages xp:deliverytype="SCORM" xlink:label="L5416" xp:description="Introduction" kind="R">
        <SubPage xp:deliverytype="SCORM" xlink:label="L5417" xp:description="Introduction_text">
            <ElementGroup xp:deliverytype="SCORM" xlink:label="L5418" xp:description="introduction_text_en" language="en">
                <ElementGroup xp:deliverytype="SCORM" xlink:label="L5419" xp:description="Introduction_text_a" type="Tourist">
                    <Text xlink:label="L5420">Welcome. In this section you will discover one of Botticelli most beautiful masterpieces
shown in the Galleria degli Uffizi, the painting Madonna of the Pomegranate.</Text>
                    <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_en_tourist.wav"/>
                </ElementGroup>
            </ElementGroup>
        </SubPage>
    </SubPages>
</Page>
```

In fact, to add a new language version is enough to add an element with the same structure but a different language identification that

```
<Page xp:deliverytype="SCORM" xlink:label="L5415" xp:description="S1" id="Page01">
    <SubPages xp:deliverytype="SCORM" xlink:label="L5416" xp:description="Introduction" kind="R">
        <SubPage xp:deliverytype="SCORM" xlink:label="L5417" xp:description="Introduction_text">
            <ElementGroup xp:deliverytype="SCORM" xlink:label="L5418" xp:description="introduction_text_en" language="en">
                <ElementGroup xp:deliverytype="SCORM" xlink:label="L5419" xp:description="Introduction_text_a" type="Tourist">
                    <Text xlink:label="L5420">Welcome. In this section you will discover one of Botticelli most beautiful masterpieces
shown in the Galleria degli Uffizi, the painting Madonna of the Pomegranate.</Text>
                    <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_en_tourist.wav"/>
                </ElementGroup>
            </ElementGroup>
            <ElementGroup xp:deliverytype="SCORM" xlink:label="L5425" xp:description="introduction_text_it" language="it">
                <ElementGroup xp:deliverytype="SCORM" xlink:label="L5426" xp:description="Introduction_text_a" type="Tourist">
                    <Text xlink:label="L5427">Benvenuti. In questa sezione scoprirete uno dei capolavori del Botticelli presente nella Galleria
degli Uffizi la Madonna della melagrana.<br/>
                    </Text>
                    <audio src="L06_Madonna_Pomegranate_high/L06_Madonna_Pomegranate_high_Page01_it_tourist.wav"/>
                </ElementGroup>
            </ElementGroup>
        </SubPage>
    </SubPages>
</Page>
```

The language aspect is to be handled also in the metadata part of the content, luckily in that respect standards provide some support as they state that wherever it is necessary to specify a language such as in data element 'General.Language' or in any language string the following coding can be used:

1. use a 2 letter code from ISO 639-1

2. use a 3 letter code from ISO 639-2 (see: http://www.loc.gov/standards/iso639-2/normtext.html, it does not matter between bibliographic & terminology since they only differ for languages that have 2-letter codes)
3. add the ISO Country code [ISO3166] when necessary, separated by a dash.
4. use IANA registered language tags, prefixed with **i-**
5. use SIL Ethnologue 3-letter codes, prefixed with **x-E-**
6. make up a name for token languages prefixed with **x-T-**
7. make up a name, prefixed with **x-** for user defined languages

All the above are acceptable but partners should prefer 1, 2, or 3; in the following table some examples are provided:

Coding	Description	Type
nl	Dutch	ISO 639-1
aus	Australian Languages	ISO 639-2
IT	Italian	ISO 3166
i-xxxxx	IANA registered xxxxx	IANA
x-E-pcd	Picard	SIL Ethnologue
x-T-ELR	The ELR Token Language	SIL Ethnologue
x-none	Not possible to identify a language	SIL Ethnologue

2.12.4 Post processing & Localising

This step is usually performed to finalize produced content either following a specific request coming from the customer/user or in line with the publisher/distributor need for a "personalized" look and feel of the content. It can comprise several operations but none altering the content substance. Usually what is done is to change layout template to accommodate a different logo, change in boldface type/color, change in background colour or position of the content. More details in relation to templates and formats have already been provided both in this and other reference documents, namely:

- DE3.1.3 AXMEDIS Content Aspects Specification
- DE8.1.1 Content for Test Cases and Validation
- DE8.2.1 Content Selection Guidelines
- DE8.4.1 AXMEDIS Editorial Format Guidelines and basic examples

2.12.5 Metadata and tagging

This step of the process is extremely important as the whole management infrastructure for the kiosk is strictly dependent on the availability and quality of the information that accompany the object, regardless of the kind. Basically it would be necessary to fill in the Mandatory and part of the Optional metadata described in the LOM standard if all was limited to our environment. Given the fact that AXMEDIS has its own metadata it is necessary to see where superposition allows automatic replication/transferring of data and where it will be necessary to specifically insert relevant metadata. In practice we have:

- **Data common between AXInfo and LOM** – this data will be possibly automatically filled in at creation time.
- **Data common between Dubblincore and LOM** – this data will be possibly automatically filled in at creation time.
- **Data specific to AXInfo** – this data will be filled in at authoring time.
- **Data specific to LOM** – this data should have been possibly filled in at authoring time.
- **Data specific to Dubblincore** – this data will be possibly filled in at authoring time for classification and management purposes.

There are two types of element subsets defined here: the elements that should be filled in every metadata instance (mandatory elements) and the elements that would be very useful to be filled (recommended elements). All other elements of our full element set are considered as optional and there is also information about some optional elements. In addition to element explanations this section contains full listings of the vocabularies defined by the project and the data types to be used as value spaces of metadata elements. The purpose of the Metadata Application Profile is to support the exchange of information about online digital resources (Learning Objects) between partners. The metadata described in this application profile supports a variety of LO uses including:

- Management
- Searching and finding
- Technical interoperability
- and description of properties of individual LOs including:
- Educational attributes
- Digital rights
- Technical features

IEEE Learning Object Metadata standard (LOM) has been selected as basis for the set of adopted metadata and to support interoperability with other metadata schemes. The information model for the metadata is similar to that of LOM where metadata for a described LO is stored in a metadata element and actual content of an element is called a value. Values can be entered as free text, inserted in predefined format or they are selected from set lists, which are called vocabularies. There are five data types in the LOM information model briefly, LOM types are:

- **CharacterString**: text can be entered in the element directly.
- **LangString**: the text must identify its language and there can be one or more character strings in the element.
- **DateTime**: the element contains date and time information and there can also be textual information about this point in time.
- **Duration**: the element contains information about an interval in time and there can also be textual information about the duration.
- **Vocabulary**: the element contains source and value where source is a reference to publicly sourced and maintained value set and value is a value from that set.

In the remainder of the section are reported the principal metadata associated to a LO and their partitioning in are of major relevance. There are several areas and for each there is a set of values. It is nevertheless possible to modify this structure adding or removing data. Both operations have to be performed following IMS standards in order to avoid ending up with a metadata structure no more compliant. In the packager interface most relevant metadata have been grouped for easier input. In the advanced view is possible to insert, modify or delete data following the regular structure foreseen in IMS. Such structure and related meaning is reported hereafter:

2.12.5.1 Mandatory Elements

In the following paragraph are reported the metadata field that should be considered mandatory when compiling metadata. These are basically the fields that will enable efficient search and retrieval of LO.

- ***General.Identifier*.**
- ***General.Title*.**
- ***General.Language*.**
- ***General.Description*.**
- ***Technical.Location*.**
- ***Educational.Intended End User Role*.**
- ***Educational.Typical Age Range*.**
- ***Rights.Copyright and Other Restrictions*.**
- ***Rights.Description*.**
- ***Classification.Keyword*.**

2.12.5.2 Recommended Elements

Recommended elements are those that would be very useful to have filled in for every metadata instance that is exposed, but they could be left unfilled.

- ***General.Keyword***.
- ***General.Structure***.
- ***Life Cycle.Contribute***.
- ***Life Cycle.Contribute.Role***.
- ***Life Cycle.Contribute.Entity***.
- ***Life Cycle.Contribute.Date***.
- ***Meta-Metadata***.
- ***Meta-Metadata.Contribute.Role***.
- ***Meta-Metadata.Contribute.Entity***.
- ***Meta-Metadata.Date***.
- ***Meta-Metadata.Language***.
- ***Technical.Format***.
- ***Technical.Size***.
- ***Technical.Facet***.
- ***Technical.Facet.Value***.
- ***Educational.Learning Resource Type***.
- ***Educational.Learning Context***.
- ***Educational.Typical Learning Time***.
- ***Educational.Description***.

Further information and background knowledge can be found in the related sections of DE3.1.3 and DE8.2.1, what reported here is just for quick reference.

2.12.6 Sealing / Securing

This phase of the process is extremely important, but nowadays this process is quite difficult and requires plenty of efforts and some additional trick to be fully operational. The reason for such statement is that the current state of the art in the sectors foresees the need for several different steps of protection. In fact, usually raw content is fingerprinted and watermarked (whenever possible) prior to any usage. The achieved result in this way is the possibility to track content in the market and (after verification of watermark existence) possible suing action is taken to oppose any illegal usage of content. To this effort usually publishers devote a full staff. The adoption of AXMEDIS will make this all easier as apparent from the following comparison:

Action	**Traditional publishing**	**AXMEDIS supported publishing**
Asset protection	Done on images, audio and text (the latter only in some cases like music) usually mainly via watermarking and fingerprinting. Verification process is demanded to legal experts and specific staff.	Automatic at content saving if requested. Verification is performed via framework.
Object protection	Done like on assets	Done like on assets
DRM management	Done via specific tools that are available only for digital based distribution	Fully handled by the platform.

In detail for the kiosk environment it is expected that the personnel in charge of the content finalisation will follow the procedure reported hereafter:

Issue	**Performed action**
Assets & objects protection	The action is necessarily differentiated on the basis of the object status, meaning with this that the object could be newly created or already exhisting, in detail we have that: *New objects* Once produced the object it is saved inside the platform requesting the desired level of protection and related DRM *Already existing objects* According to granted rights the object is extracted, modified and then saved back inside the platform. At this stage it will be possible either do decide to leave protection and DRM aspects unchanged or to request the system the definition of a new set of desired level of protection and related DRM
DRM management	Personnel in charge of DRM management will be able to revise and modify at any time the info inserted on objects based on the available rights. This last sentence is refereed to the fact that in the kiosk there will be both own and acquired objects that will be therefore ruled by different sets of constraints.

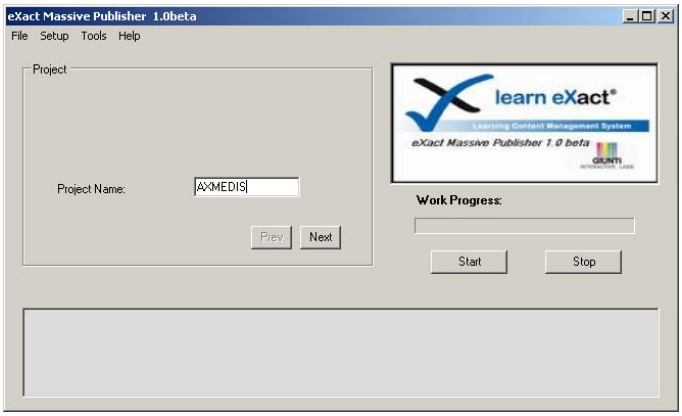
2.12.7 Packaging

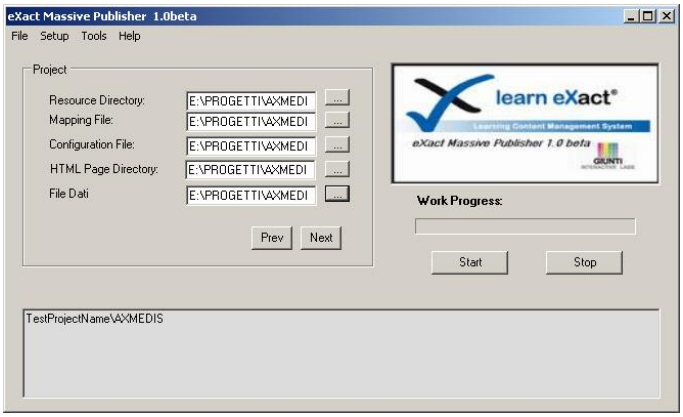
This is the almost final step of the publishing process and represents the last checkpoint prior of publication. Basically this implies that in a publishing environment, no matter if in the digital or in the physical world, the last step prior to distribution is this. During this production phase people will ensure that the product will be made suitable for distribution applying all due procedures to turn a raw product into a “ready to buy” one. In more detail we should distinguish two kind of products: those that are inherently digital and do not need to be turned into physical (placed on a physical support other but for backup purpose) and those that are very much likely to be exploited in both nature (digital and tangible). In the first case the “packaging” activity means basically securing and protecting content into some kind of packaging or streaming format that en-

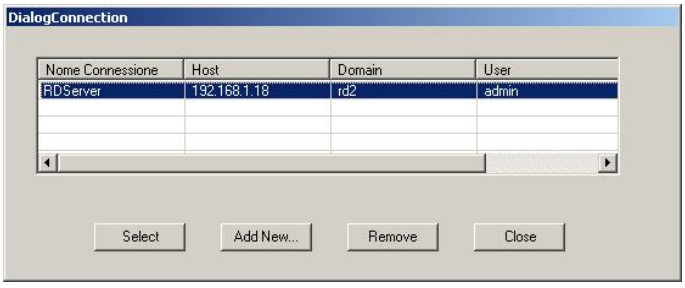
ables fruition while retaining the possibility to monitor, via DRM management, the fruition process. In the second case the issue is tackled in a rather more complex way:

	Business nature	
	B2B	**B2C**
Object nature	• Digital (DOC, TIF, MP3, PDF, EXE…), {*} • Physical (CD, DVD…)	• Digital (DOC, TIF, MP3, PDF, EXE…), [*] • Physical (CD, DVD…)
Object fruition	• Digital (use as by products, composition, distribution…) {*} • Physical (replication…) • Hybrid (replication & mastering, composition abd print…)	• Digital (fruition) [*] • Physical (CD, DVD playing) • Hybrid (using and printing a PDF…)

Items marked [*] in the previous table are the one applying to kiosk while those marked {*} apply to kiosk factory, the other are just to provide the general picture. In our environment the need to interoperate with digital repositories of several actors holding huge quantity of content that is compatible in terms of standardised format description created the need for a platform extension called Massive Publisher aimed at handling the import/export into/from the platform of substantial content quantity in a single step. Such utility may be used in combination with AXMEDIS framework.



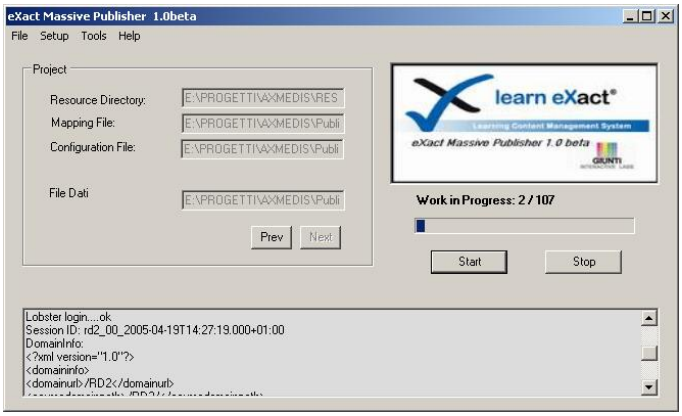




What follows is just a brief description along with some screenshots related to the above-mentioned process in the case of export, as this is the most relevant case for us as producers of content.
The starting point is the generation of mapping and configuration files. Once this step has been achieved is possible to start up the actual massive publishing process which cab be described as follows:

1) define the project name,
2) select the resource directory,
3) select the mapping file,
4) select the configuration file,
5) select the output directory,
6) select the data files,
7) select the server hosting the source digital repository
8) launch the process.

In what just described (and especially from the picture aside) is apparent that there are quite a number of files involved in the process. This is mainly due to a need for flexibility to be achieved during the process.
The mapping fiel is used to establish which field of the input source structure map onto which field of the destination (in our case usually is from an external digital repository to LEX platform).
The configuration file specifies the boundaries condition applying to the process.
The HTML page directory allows to specify templates to be used in the formatting of the



expoting content that is going to be published.

Once the process has been accomplished the final results can then be accessed and used in the destination platform. In the presented case the result is a set of packages holding IMS content packages and all related resources that can be subsequently imported into other authoring environments or ingested into the AXMEDIS system thanks to the crawling process.

What follows are excerpts samples of support files used for the process just depicted.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata">
<Artista>
<IDArtista>1</IDArtista>
<Cognome>Gauguin</Cognome>
<Nome>Paul</Nome>
<LuogoNascita_ENG><![CDATA[Parigi ]]></LuogoNascita_ENG>
<DataNascita>1848</DataNascita>
<LuogoMorte_ENG><![CDATA[Hiva Oa  (Isole Marchesi) ]]></LuogoMorte_ENG>
<DataMorte>1903</DataMorte>
<Secolo>800</Secolo>
<Ritratto>/img/ritratti/schecd03.jpg</Ritratto>
<Biografia_ENG><![CDATA[Paul Gaugin was born in Paris in 1848 and the following year his family moved to Peru. He returned to his native country when he was seven and studied in boarding schools in Orleans and Paris. In 1865 he sailed for South America as a midshipman on a merchant ship. Over the next two years he spent his time at sea and fought in the Franco-Prussian War (1870). At the end of the war in 1871 he obtained a position as a stockbroker and began painting. In the coming years he met Pissarro and Cezanne and joined the Impressionists, participating in some of their exhibits. In 1883 he left his job and moved to Rouen, where he stayed with Pissarro. Following an artistic maturation that led him to consider “primitive” artistic experiences as fundamental he began a series of journeys that took him from Europe to South America to the French colonies in the Marquesas Islands. In 1886 he went to Brittany, and specifically to Pont-Aven, for the first time and he returned in 1888 after a trip to Martinique. His experience in Brittany was fundamental the development of his “synthetisme” a style that Albert Aurier, a contemporary critic was to define as “idealistic, symbolistic, synthetic, subjective and decorative” art. At the base of the synthetisme was his familiarity with Japanese prints, the primitivism expressed by Breton sculpture, the flat colors and cloisonnisme of Gothic stained glass windows. A fundamental example of the synthetisme is the painting entitled <I>The Vision After the Sermon</I>, 1888. After a brief stay at Arles, as a guest of Van Gogh and at Le Pouldou, after 1891 he made several trips to Tahiti where he colored his already marked eclectic primitivism, that was also developed on the basis of his “photographic” knowledge of Egyptian painting and the sculptures of Partendone and Borobodur, with exoticism. His life in the refound paradise of Oceania was not, however, idyllic: it was marked by illness, a suicide attempt and in the Marquesas Islands where he moved in 1901 – a prison sentence for having instigated the natives to revolt. He died at Hiva Oa in 1902. His artistic experience that was fundamental for the contemporary Nabis painters, also influenced the studies of the Fauvists and German Expressionists of the Brücke group.]]></Biografia_ENG>
<Citazione_ENG><![CDATA[I am a great artist and I know it. And it is precisely because I am that I have suffered greatly: to follow my life, otherwise I would consider myself an outlaw. Which is what I am for many. In the end, what does it matter? What torments me most is not so much the poverty as the continuous obstacles to my art that I cannot do as I feel it, and could do without the poverty that ties my hands. You say that I am wrong in wanting to stay away from the art centers. No, I am right: it is some time now that I know what I am doing and why I am doing it. My artistic center is in my brain and nowhere else, and I am great because I do not let myself be disturbed by others and because I do what is inside me.
<BR> <BR> Paul Gauguin, <I> Lettera alla moglie </I> , Tahiti, marzo 1892
]]></Citazione_ENG>
</Artista>
…
</dataroot>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata">
<QuadroCronologico>
<Anno>1864</Anno>
<Cognome>Toulouse-Lautrec</Cognome>
<Nome>Henri de</Nome>
<VitaArtista_ENG></VitaArtista_ENG>
<EventoStorico_ENG></EventoStorico_ENG>
</QuadroCronologico>
<QuadroCronologico>
<Anno>1865</Anno>
<Cognome>Toulouse-Lautrec</Cognome>
```

```
<Nome>Henri de</Nome>
<VitaArtista_ENG></VitaArtista_ENG>
<EventoStorico_ENG></EventoStorico_ENG>
</QuadroCronologico>
<QuadroCronologico>
<Anno>1868</Anno>
<Cognome>Toulouse-Lautrec</Cognome>
<Nome>Henri de</Nome>
<VitaArtista_ENG></VitaArtista_ENG>
<EventoStorico_ENG></EventoStorico_ENG>
</QuadroCronologico>
<QuadroCronologico>
<Anno>1737</Anno>
<Cognome><![CDATA[Canaletto ]]></Cognome>
<Nome></Nome>
<VitaArtista_ENG></VitaArtista_ENG>
<EventoStorico_ENG></EventoStorico_ENG>
</QuadroCronologico>
<QuadroCronologico>
…
</dataroot>


<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata">
<Iconografia>
<Iconografia_IDIconografia>1</Iconografia_IDIconografia>
<TermineIconografico_ENG>Gabriel, archangel</TermineIconografico_ENG>
<Descrizione_ENG></Descrizione_ENG>
<IDIcona>90</IDIcona>
<IconografiaOpere_IDOpera>5</IconografiaOpere_IDOpera>
<IconografiaOpere_IDIconografia>1</IconografiaOpere_IDIconografia>
<Opere_IDOpera>5</Opere_IDOpera>
<ImmagineXL>/img/large/206gcd05.jpg</ImmagineXL>
</Iconografia>
<Iconografia>
<Iconografia_IDIconografia>1</Iconografia_IDIconografia>
<TermineIconografico_ENG>Gabriel, archangel</TermineIconografico_ENG>
<Descrizione_ENG></Descrizione_ENG>
<IDIcona>194</IDIcona>
<IconografiaOpere_IDOpera>47</IconografiaOpere_IDOpera>
<IconografiaOpere_IDIconografia>1</IconografiaOpere_IDIconografia>
<Opere_IDOpera>47</Opere_IDOpera>
<ImmagineXL>/img/large/232gcd02.jpg</ImmagineXL>
</Iconografia>
<Iconografia>
<Iconografia_IDIconografia>1</Iconografia_IDIconografia>
<TermineIconografico_ENG>Gabriel, archangel</TermineIconografico_ENG>
<Descrizione_ENG></Descrizione_ENG>
<IDIcona>379</IDIcona>
<IconografiaOpere_IDOpera>633</IconografiaOpere_IDOpera>
<IconografiaOpere_IDIconografia>1</IconografiaOpere_IDIconografia>
<Opere_IDOpera>633</Opere_IDOpera>
<ImmagineXL>/img/large/i08g-091.jpg</ImmagineXL>
</Iconografia>
…
</dataroot>


<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata">
<Opere>
<IDOpera>1</IDOpera>
<IDArtista>10</IDArtista>
<Titolo_ENG>St. Jerome</Titolo_ENG>
<Sottotitolo_ENG>St. Jerome</Sottotitolo_ENG>
<Data><![CDATA[1480-1482 ]]></Data>
<Tecnica_ENG>oil on wood</Tecnica_ENG>
<Altezza>103</Altezza>
<Larghezza>75</Larghezza>
<LuogoConservazione>Roma, Pinacoteca vaticana</LuogoConservazione>
<IDTipologia>1</IDTipologia>
<IDGenere>1</IDGenere>
```

<Commento_ENG>We know very little if anything at all about this painting that was probably done during Leonardo s Florentine sojourn. This period is linked to his interests in the human form and representing its movement in space. The painting, a contemporary of the Adoration of the Magi is unfinished, but it is precisely the incomplete state that confers intense drama on the figure. The anatomical model is carefully planned, as it is hit by light; the compact and sculptural. St. Jerome is set in a rocky cavern against a barely sketched landscape that is typical of the profound and misty backgrounds of Leonardo s paintings.</Commento_ENG>
<ImmagineSM>/img/small/212pcd05.jpg</ImmagineSM>
<ImmagineL>/img/medium/212mcd05.jpg</ImmagineL>
<ImmagineXL>/img/large/212gcd05.jpg</ImmagineXL>
<DirittiFoto></DirittiFoto>
<Museum>1</Museum>
</Opere>
<Opere>
<IDOpera>3</IDOpera>
<IDArtista>10</IDArtista>
<Titolo_ENG><![CDATA[Baptism of Christ]]></Titolo_ENG>
<Sottotitolo_ENG><![CDATA[Baptism of Christ]]></Sottotitolo_ENG>
<Data>1470-1474 circa</Data>
<Tecnica_ENG>Oil and tempera emulsion on wood</Tecnica_ENG>
<Altezza><![CDATA[117]]></Altezza>
<Larghezza>151</Larghezza>
<LuogoConservazione>Firenze, Galleria degli Uffizi</LuogoConservazione>
<IDTipologia>1</IDTipologia>
<IDGenere>1</IDGenere>
<Commento_ENG>Oil and tempera emulsion on woodpainting of St. John baptizing Jesus before the eyes of two angels was done for the Church of San Salvi in Florence, and it has been in the Uffizi Gallery since 1914. It is probable that Verrocchio, Leonardo s teacher, had been commissioned to do the painting, but as was his custom, much of it was done by his pupils. Sources dating from as early as the beginning of the sixteenth century say that it was the young Leonardo who painted the angel on the left, and recently even the background landscape has been attributed to him. It is believed that other pupils of Verrocchio such as Domenico di Michelino, Sandro Botticelli and Francesco Botticini also worked on this painting.</Commento_ENG>
<ImmagineSM>/img/small/201pcd05.jpg</ImmagineSM>
<ImmagineL>/img/medium/201mcd05.jpg</ImmagineL>
<ImmagineXL>/img/large/201gcd05.jpg</ImmagineXL>
<DirittiFoto>Archivio Giunti</DirittiFoto>
<Museum>1</Museum>
</Opere>
<Opere>
<IDOpera>21</IDOpera>
<IDArtista>10</IDArtista>
<Titolo_ENG>Self-Portrait</Titolo_ENG>
<Sottotitolo_ENG>Self-Portrait</Sottotitolo_ENG>
<Data>dopo il 1515</Data>
<Tecnica_ENG>red crayon on paper</Tecnica_ENG>
<Altezza>33,3</Altezza>
<Larghezza>21,3</Larghezza>
<LuogoConservazione>Torino, Biblioteca reale</LuogoConservazione>
<IDTipologia>14</IDTipologia>
<IDGenere>2</IDGenere>
<Commento_ENG>This intense and powerful self-portrait shows Leonardo in his sixties with a long thick beard, when he had been living in France for some time. The paper was probably trimmed on the sides to make his curved shoulders less conspicuous. Leonardo wanted to leave a drawing of himself for posterity, according to an idea that he expressed in his notes. Scholars believe that he periodically painted or drew pictures of himself that can be recognized in various works. Many scholars believe it unlikely that Leonardo, by then an old man, could still draw as sharply as this self-portrait would suggest and not all concur that it was done by his hand. In fact, it has been stated that the drawing is an early nineteenth century forgery. Another, perhaps erroneous yet curious hypothesis maintains that it is Leonardo s father and was done by the master some time around 1505.</Commento_ENG>
<ImmagineSM>/img/small/245pcd05.jpg</ImmagineSM>
<ImmagineL>/img/medium/245mcd05.jpg</ImmagineL>
<ImmagineXL>/img/large/245gcd05.jpg</ImmagineXL>
<DirittiFoto>Archivio Giunti</DirittiFoto>
<Museum>1</Museum>
</Opere>
…
</dataroot>

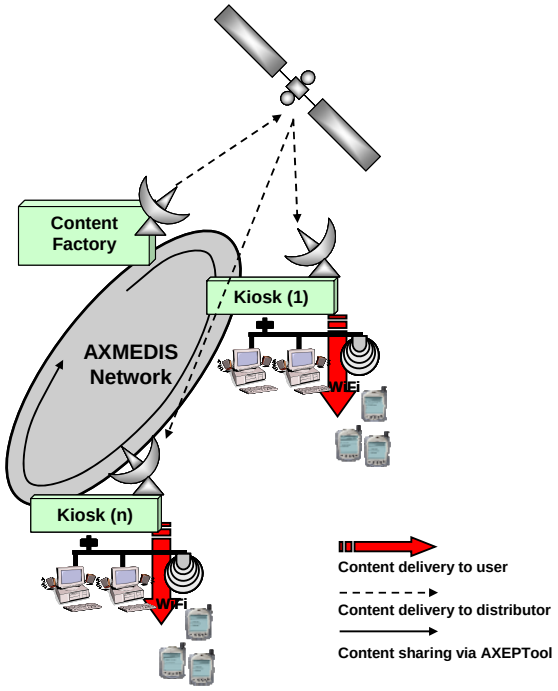
As already mentioned, packaging is the last step of the production chain. We will drop now any consideration on deliveries that are not digital as they are out of the present demonstrator and therefore we can say that once content has been published onto a digital repository it will be available for the management either with an e-learning platform (which is usually the case for us) or with the facilities of the backend environment (in our case the kiosk factory management) including AXMEDIS.

3 Kiosk distributor demonstrator fact sheet

Within AXMEDIS the kiosk demonstrator aims to demonstrate the benefits coming from the combination of several technologies in a totally new environment. Usually when referring to kiosks (in tourism or museums) people is focused on point of services with the typical aspect of what is called a “totem”.

In AXMEDIS a kiosk is much more; it is a basic subset of the overall framework, designed and tailored to provide a fully-fledged set of services to a set of users in parallel exploiting all available tools as a real and new distribution channel.

The Kiosk within AXMEDIS Architecture is divided in two components: the “factory” where content are produced and the “kiosk” where users have access to services and contents. Distribution from the factory to kiosks is achieved via satellite so to optimise bandwidth and data transfer rate when updating (in broadcast) units that may be geographically dispersed on the territory, while content access, selection, acquisition and fruition at the kiosk is achieved exploiting either local terminals (true points of service) or a WiFi based access via PDA or mobiles.



- Kiosk architecture integration with AXMEDIS
    - Content Acquisition / provision:
        - From partners’ databases, from partners granting access, through Query Support and LoaderSaver Web Services
        - Internally produced
    - Content Production / Processing (SMIL, HTML…):
        - From Kiosk factory, through either the AXMEDIS Editor or the AXCP Editor through rules and templates.

        - Number of content items produced/processed per month will largely depend on the expected number of events that the kiosk location will hold in the same period, but in line of principle they will range from 15 to 30 including the related catalogue.
        - Number of content items produced/processed at the same time will derive directly form the previous procedure as once defined the target for the incoming month production, related templates will be selected or produced, rules will be selected or adapted from available ones and then content will be aggregated consequently via batch process.
    - Content protection will be achieved from the kiosk factory, through the:
        - AXMEDIS Editor
        - AXCP Editor through rules
    - Mother licenses will be produced in the kiosk factory for our contents and on the basis of a master license granted by ANSC, through:
        - AXMEDIS DRM Editor
        - AXCP Editor through rules
    - End-user final licenses will be produced at the kiosk factory side through:
        - AXMEDIS DRM Editor
        - AXCP Editor through rules
    - Registration of user and devices, will be divided in two:
        - KIOSK users are registered when they ask to; our application uses the AXCS Registration Web Service (an AXCS instance is installed at the factory side)
        - Local PDA will belong to the kiosk installation and therefore will be certified once only (at the time of entering into operation) unless the operation will be required a new consequently to a change in tools version.
    - Content distribution at the kiosk will occur in two forms:
        - Kiosk: at the demonstration place there will be a number of POP = Point of Presence with PCs equipped to show a “catalogue” of contents, granting to registered users the ability to interact with contents
        - Mobile distribution: at the demonstration place there will be a number of PDAs and phones equipped with our application to list a set of contents, granting to registered users the ability to interact with contents
    - Accounting collection and action monitoring will rely on the fact that the AXOM inside the AXMEDIS Tools will communicate to the AXCS the usage data. Both Accounting services and AXCS will be at the Factory side
- Description of the effective installation
    - Distribution infrastructure needs comprise a server (where AXMEDIS Tools will be installed along with the firewall and the other back office components needed), a few terminals acting as POPs (basically PCs) and a few PDAs/Smartphones acting as mobile POPs.

  - Servers: at least one
  - Players: one AXMEDIS player for each POPs, a proper AXMEDIS Mobile/PDA Player for each Mobile/PDA device.
  - Streaming/downloads: TBD

- AXMEDIS tools
  - Major AXMEDIS tools used:
    - The whole set of AX-Tools
    - P2P will not be used in the Kiosk, while it may be used at the kiosk-factory mainly for content search and retrieval or for content sharing between partners involved in the demonstrator (ANSC and ILABS)
    - AXCP will be used at the factory side in order to create contents and licenses
    - Workflow tools will be used at the factory side only in order to manage the content production process
    - Programme and publication will be used at the factory side only in order to communicate with the satellite system to deliver catalogue and contents from factory to kiosks
  - PMS/AXCS usage:
    - During content creation in order to protect and create mother licenses
    - During content distribution preparation, in order to create end user licenses
    - During content fruition, through AXMEDIS players and tools, since they have to verify the users' rights
  - AXMEDIS database usage:
    - Kiosk-factory/Kiosk database instance, for locally storing and retrieving
    - Remote databases, to retrieve partners' contents

- Target Market:
  - Cultural heritage institutions and/or societies providing services for museums and archaeological sites.

- Description of the business model
  - The expected business model is a combination of two typical ones: the subscription and the pay per access/use. Payments will be done in cash and the paying user will be assigned a code that will be used in payment procedures (the code corresponds to one or several tokens that will grant access/fruition of selected content)

- Description of content:
  - On the average each month will be available a set of about 100 AXMEDIS objects ready to be distributed and focused on the defined event schedule. To this has to be added the availability of all the collections which are provided by ANSC and ILABS and that will be accessible in a standardized manner that may differ form the "monthly" offer mainly in terms of aggregation.

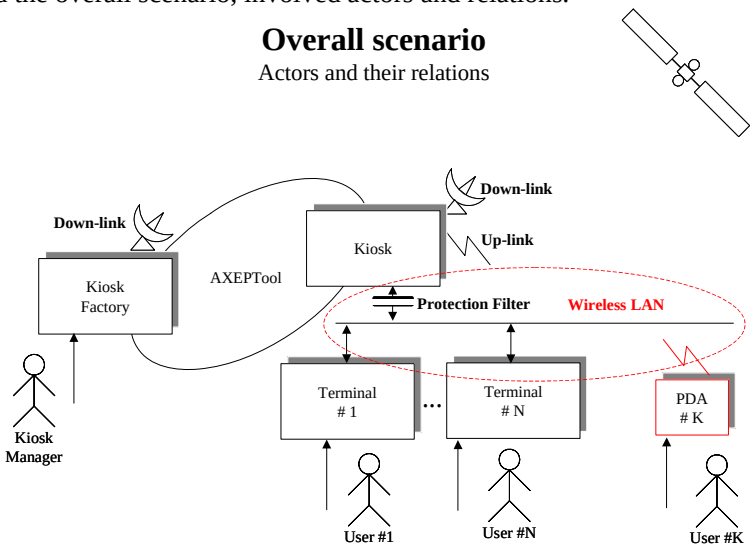
- Digital resources with the needed clearance of rights will come from ANSC and ILABS
- Content will comprise data related to the collections of the ANSC museums and events organized at the ANSC auditorium plus textual and graphical content related to art and history form ILABS.
- Available resources will certainly comprise text, images and audio, while videos may be available but only in limited number and dimensions.
- Typical Content size for each content type may greatly vary especially as far as audio/video is concerned while the typical size textual/graphical content is in the order of 2-10Mb.

- Final Users/Clients:
  - In terms of final users it is expected that a 10% of regular visitors of the ANSC museum and auditorium could be highly probably interested in experimenting the Kiosk installation and use its content, in this manner approximately 500 final users per year will be reached.
  - It is expected that for those requesting to use Kiosk's local PDAs terminals a registration is needed, at least to ensure and prevent the risk of loosing the provided equipment.

- Partners involved and roles:
  - ANCS content owner and manager of the local site
  - ILABS content owner, manager of the factory

4 Technical Specification and details

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 GUIs 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 configured to properly interact with kiosks and therefore the user will be able to use them as if he was using the kiosk

System components

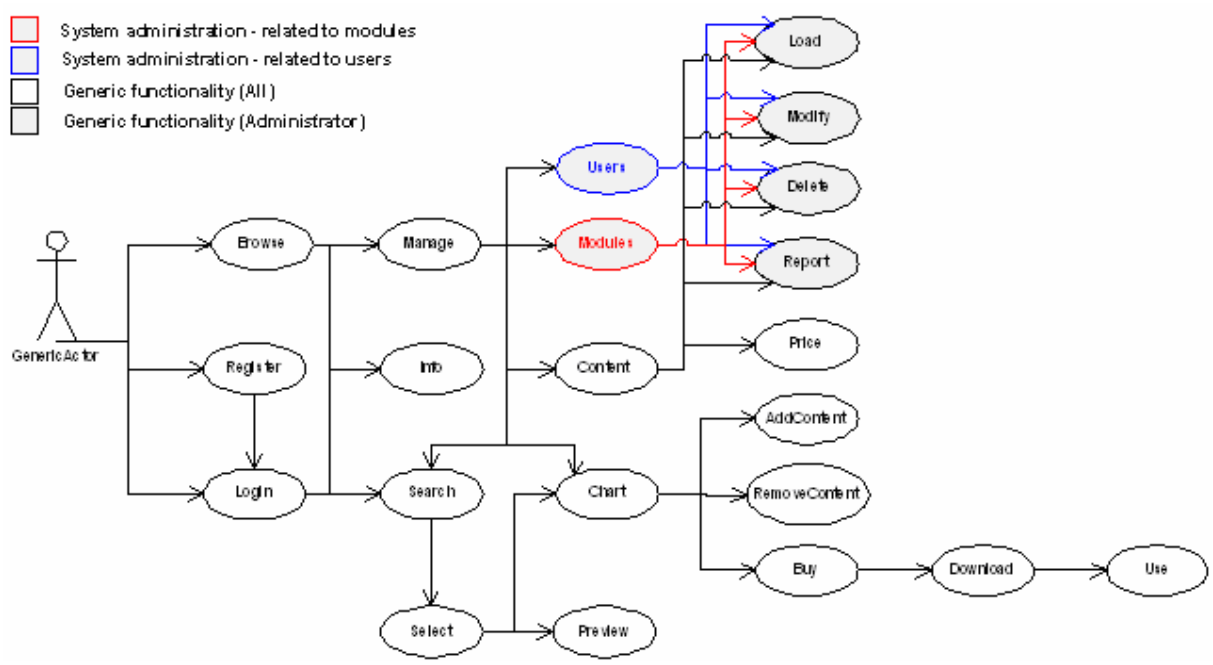
At a very high level the system components are summed up as follows. Later one we will enter in much more details.

- **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). It provides security features to the local LAN. The kiosk architecture will enable wireless communication with local mobile devices (PDA…).
- **Terminal**: this 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): this 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.

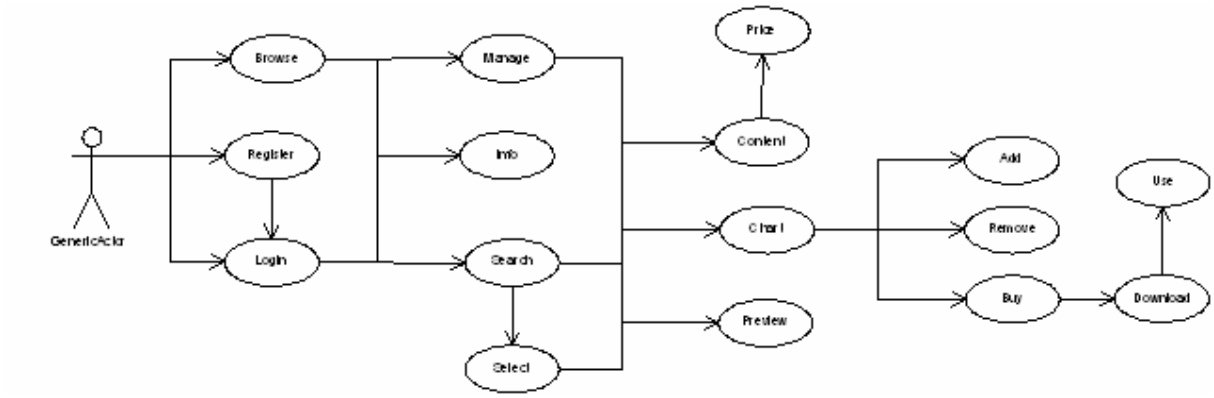
It is worth noting that Kiosk's terminals and local PDA should be certified separately and autonomously from the user PDA. They do belong to the kiosk and not to the user, therefore even if the user logs-in is not meant to certify the device as this step should occur when the kiosk starts-up.

4.1 Overall use case

Starting from what just stated before and taking into account all the possible actions that will be performed at the kiosk it 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.

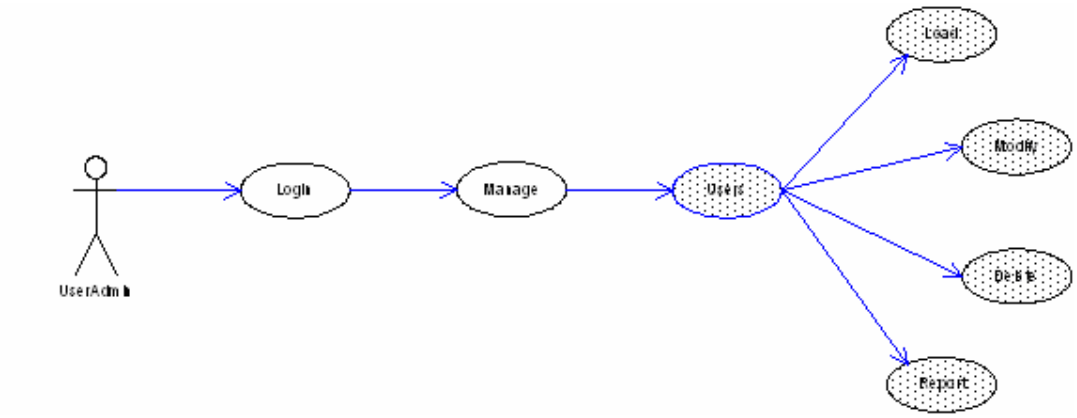


For enhancing readability, all use cases have been split into three groups, based on the main actor. Following use cases are the ones that involve a Generic Kiosk User: he can register and log in, browse and search content, add/ remove content to chart, ask price, buy, preview, download.



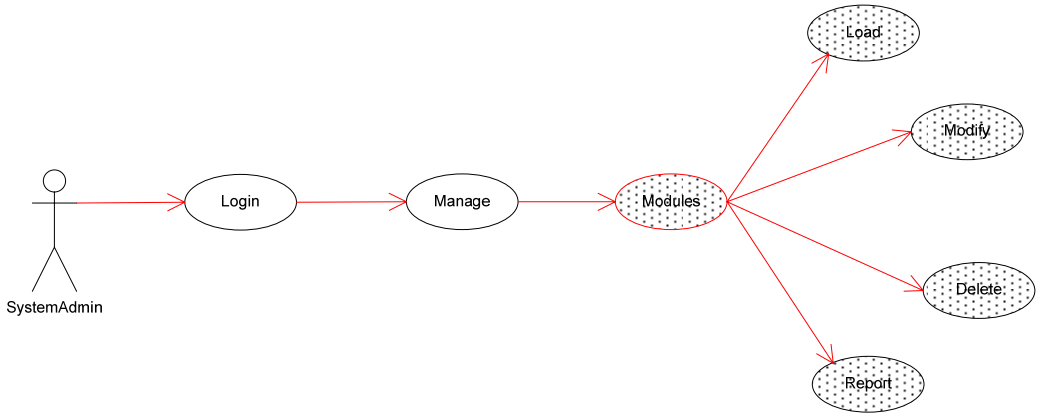
Generic User Use Cases

Following use cases are the ones related to Users Administrator reserved users management functionalities, that is, inserting new users, modifying users' data, deleting users and extracting report information. Of course, Users Administrator is also allowed to perform all Generic User's Use Cases: these are not reported in below diagram only for enhancing readability.



Users Administrator Use Cases

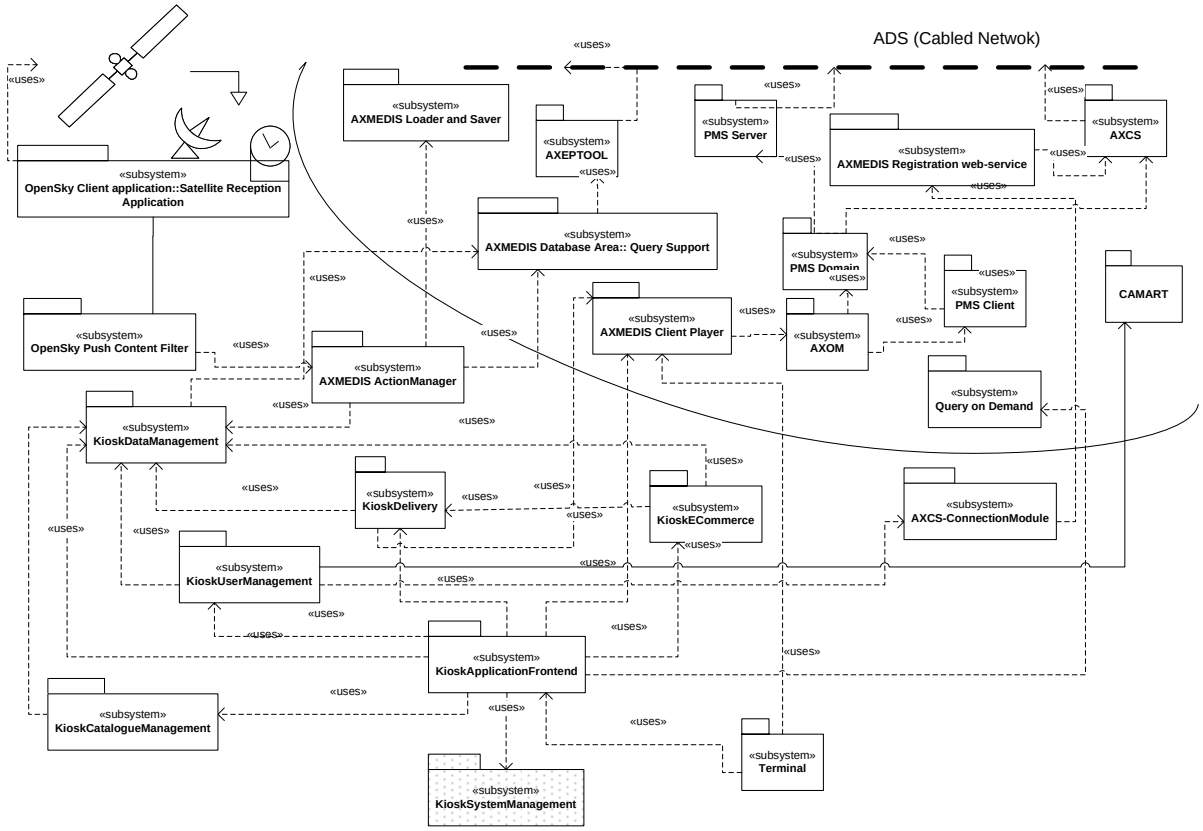
Last ones are the use cases related to System Administrator capabilities: loading, modifying, deleting and retrieve report information about system modules. A simple sketch of these use cases is presented below, omitting Generic User's Uses Cases again.



System Administrator Use Cases

4.2 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 (for example, PMS server and AXCS). There is also a 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 smartphone.



4.3 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 operations 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.

4.3.1 SetUp Use Cases

Following Use Cases are the ones related to system setup: creation of a new domain (only for administrators users), download and installation of AXMEDIS client application, user and device certification.

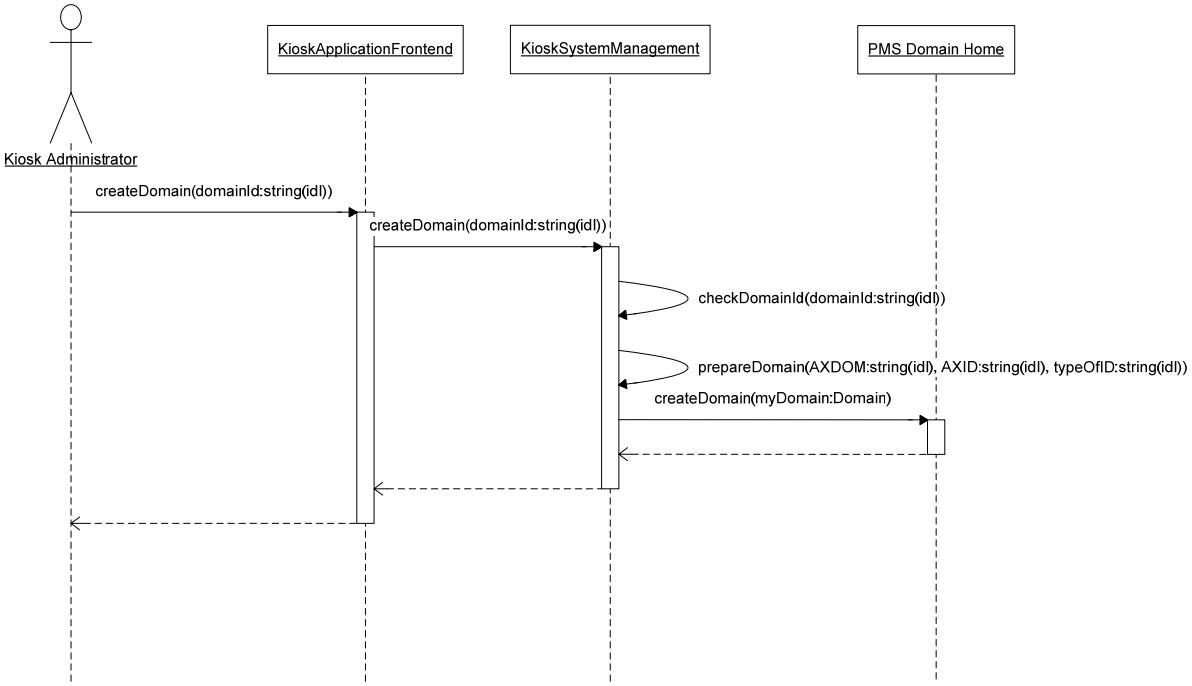
4.3.1.1 Domain registration

From a distributor's point of view, a very basic operation is domain registration. For further details about Domain definition, please see Part H, Section 2.9.2, and Section 4.6.4. A domain is defined in this way:

```
<xs:complexType name="Domain">
    <xs:sequence>
```

```
        < !- - ID of the AXMEDIS Domain - - >
         <xs:element name="AXDOM" type="xs:string" nillable="true"/>
        < !- - Domain Manager ID: it can be an ID of a B2B User or an End User according to the value of typeOfID - - >
         <xs:element name="AXID" type="xs:string" nillable="true"/>
        < !- - Reference to AXID and can have one of the two values>B2BUser/AXUID - - >
         <xs:element name="typeOfID" type="xs:string" nillable="true"/>
      </xs:sequence>
    </xs:complexType>
```

To start Domain registration, Kiosk Administrator will setup a specific configuration file giving the new domain desired identifier. Naming convention for domain names allows whatsoever sequence of characters and numbers whose length is between 3 and 30, starting by a char. The application front-end will pass information to Kiosk system management. This one will check received domain identifier and take the charge of preparing a new domain object, this will be send to the DomainPMS module which will register new domain within the whole AXMEDIS System. After that, DomainPMS will reply the result of the operation (success/failure) that will be forwarded to Kiosk Administrator, as depicted in below diagram .



Domain registration

4.3.1.2 Certification of users

When a user wishes to be certified in order to access to AXMEDIS objects, he has to send information to front-end application that forwards user and tool information to DomainPMS, this one performs all necessary checks and operations. Then DomainPMS returns acknowledge and enabling code to the application. At last, a feedback is given to user. Certification diagrams are not reported since they have been already presented in Part H, Section 2.3.2. Here certification process is also detailed.

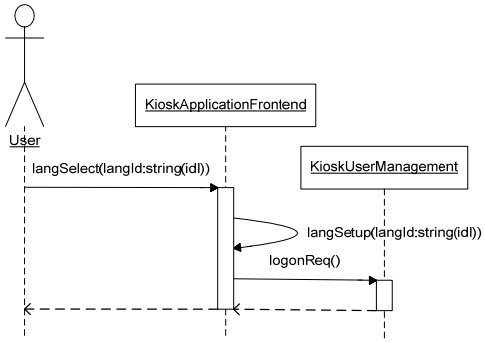
4.3.1.3 Certification of a new device

Before using a device to connect to AXMEDIS system and play contents, users have to certificate their devices. The Protection Processor inside AXMEDIS Object Manager is capable of estimating the device fingerprint and activating the Tool. Diagrams for new device certification and a more exhaustive explanation of the process have already been presented in AXMEDIS-Scenarios-Total-v4-2.ppt, slide 224, and Part H, Section 2.3.2. For more details about tool certificate, please see Part A, Section 7.2.

4.3.2 User System Access Use Cases

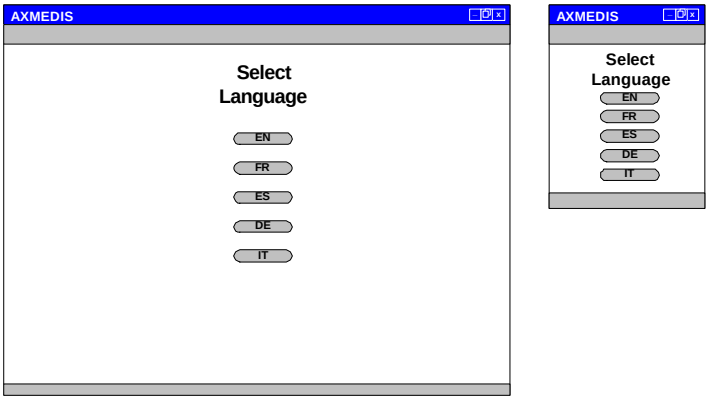
Following Use Cases describe basic operations new system users have to perform in order to acquire system fruition capabilities and rights: interface language selection, registration, login and optionally AXMEDIS client application download on personal device.

4.3.2.1 User interface language selection



User Interface language selection

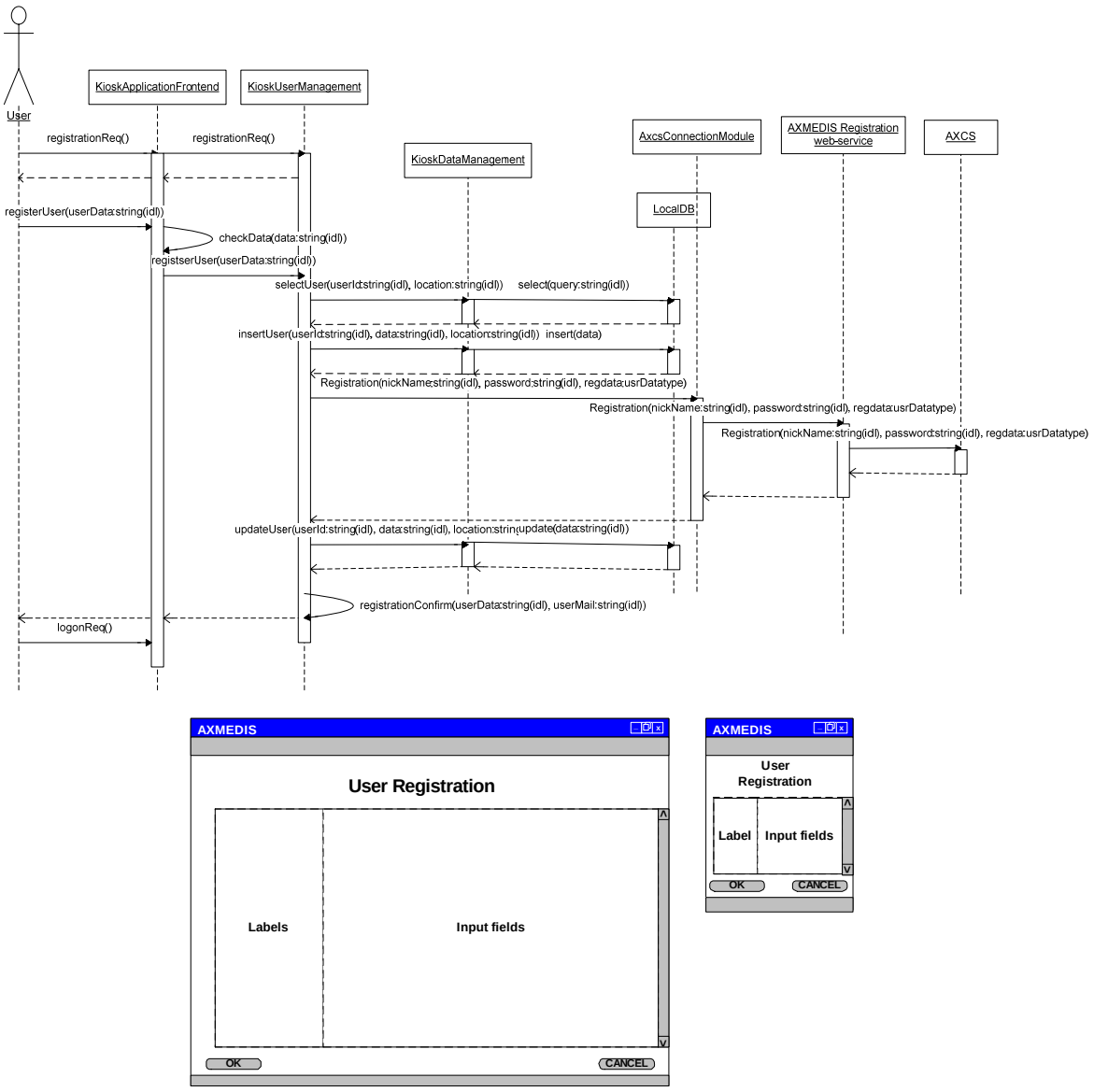
From this diagram is apparent that the user selects as the first thing the GUI language. The Application front-end intercepts such request and sets-up the appropriate environment variables to achieve expected results. Once this is achieved the Application front-end requests to the user management to 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).



4.3.2.2 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 register user. 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

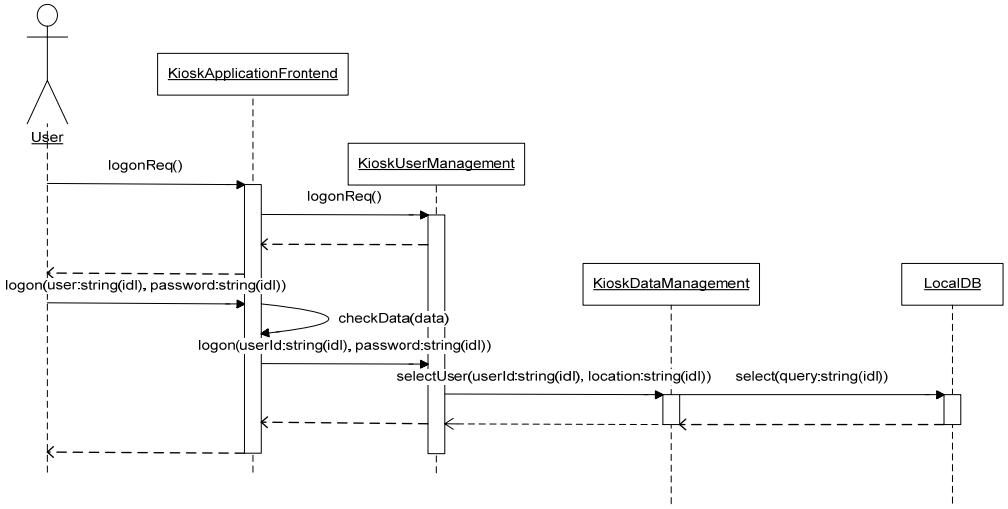
perform is to return the application front-end all AXMEDIS user related data for proper display to the user. The user will have to take note of part of the data (the AXUID, the password and her/his personal certificate) prior to proceed accessing the main kiosk GUI.



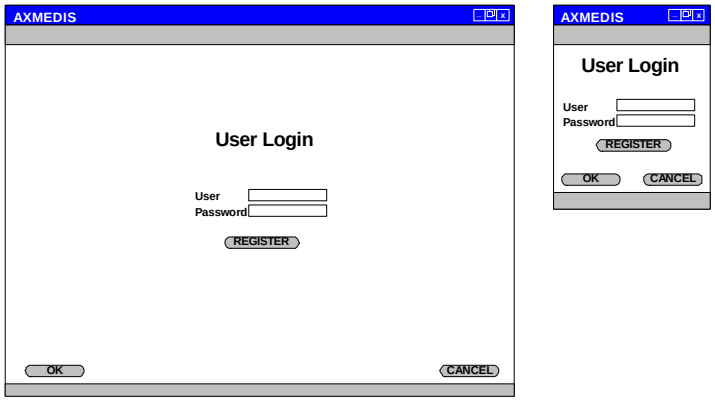
User registration

4.3.2.3 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 check 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. 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.

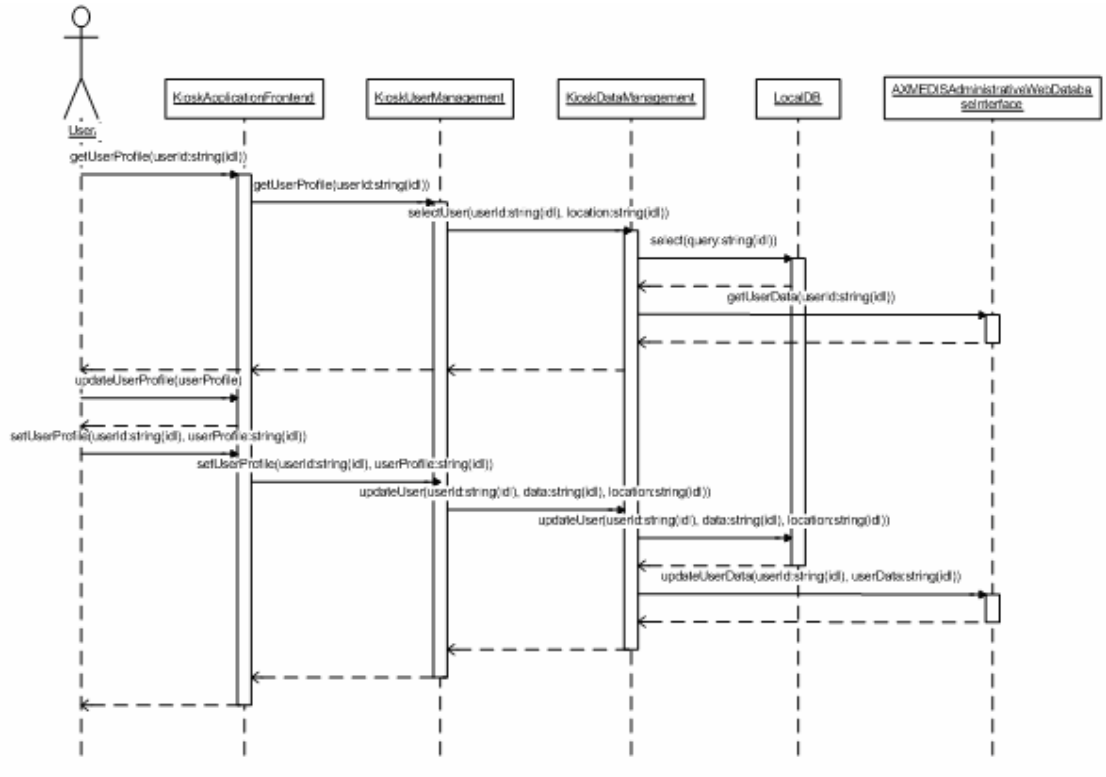


User login



4.3.2.4 Update profile

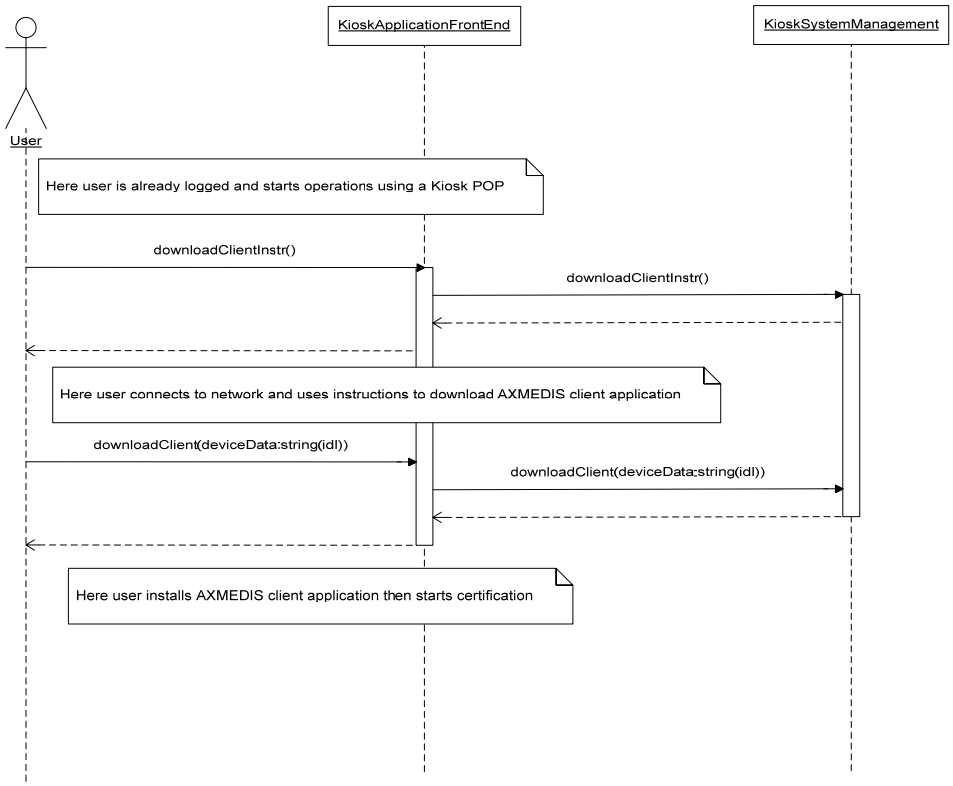
When s/he wishes to update her/his profile, the user asks back for her/his registered data both from local and AXMEDIS databases, performs updates and then stores the updated profile. Of course, s/he has no direct access to database but interacts with front-end application; this, in turn, will interact with user management module, that will communicate to data management module in order to retrieve and store data into repositories. The temporal diagram for this use case is reported below. A simple sketch of a possible user interface is also reported.



User profile update

4.3.2.5 Client application download

An user at the Kiosk can choose to connect to AXMEDIS using a Kiosk POP or her/his own personal device. In the second case, s/he has to download an AXMEDIS client on device before. In order to get AXMEDIS client, s/he has to use a Kiosk POP on a starting phase. In following use case, we assume the user is already registered and logged in the system. So s/he requires download instructions to the application front end, which in turn requires instructions to the system management module. Once the instructions are returned to the user, s/he can connect to network using her/his personal device and download the AXMEDIS client player.



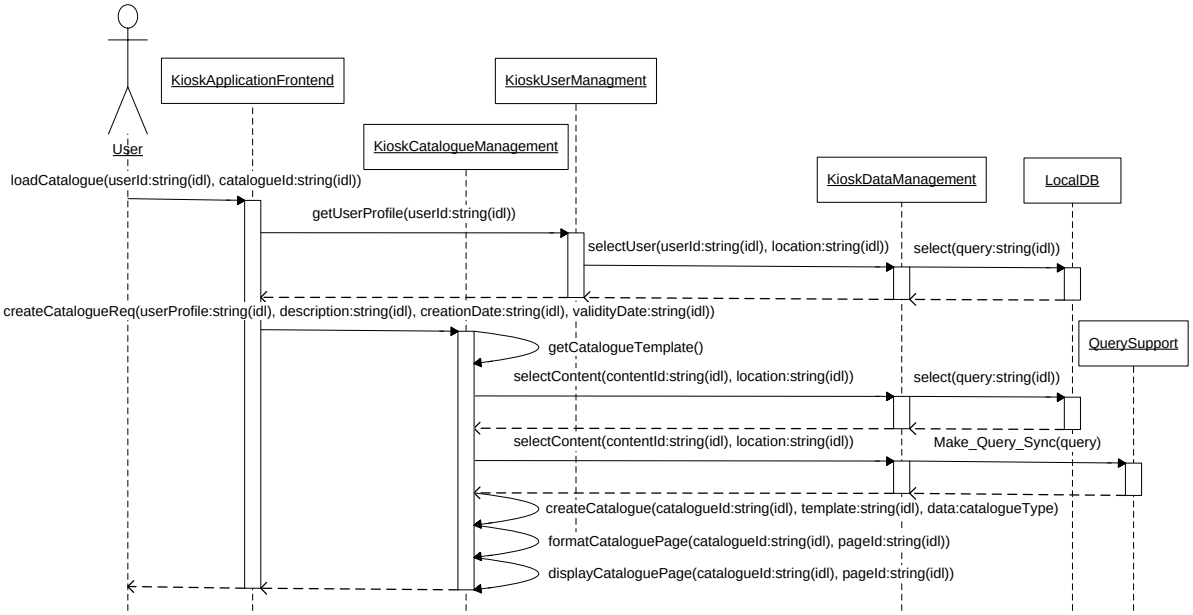
Client application download

4.3.3 Content Fruition Use Cases

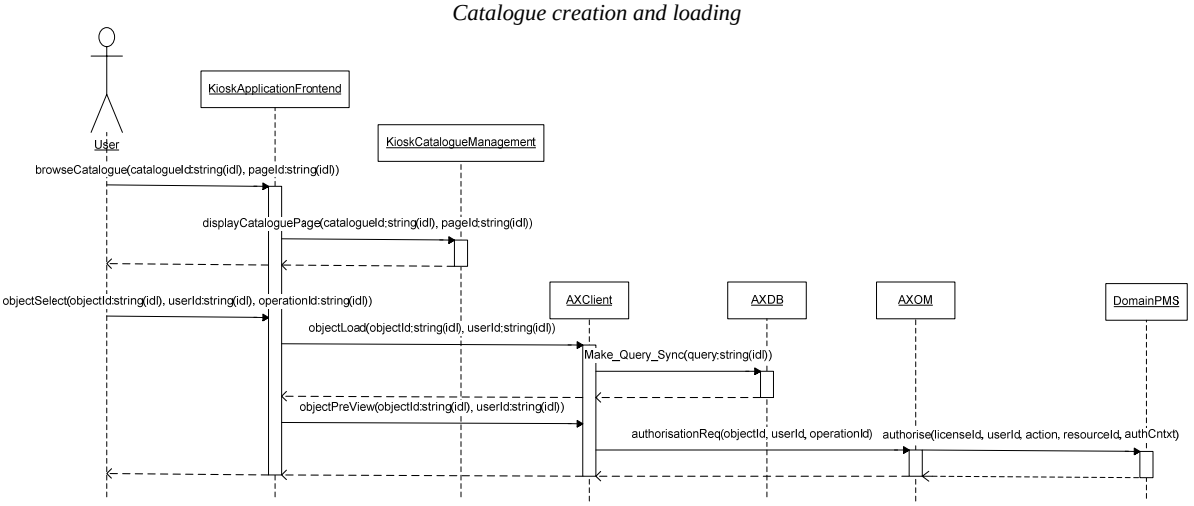
Following Use Cases are the ones conceiving contents fruition: content load, selection and preview; content adding to chart; content purchase; content delivery. Each one will be explained and detailed.

4.3.3.1 Content load, selection & preview

What follows 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.

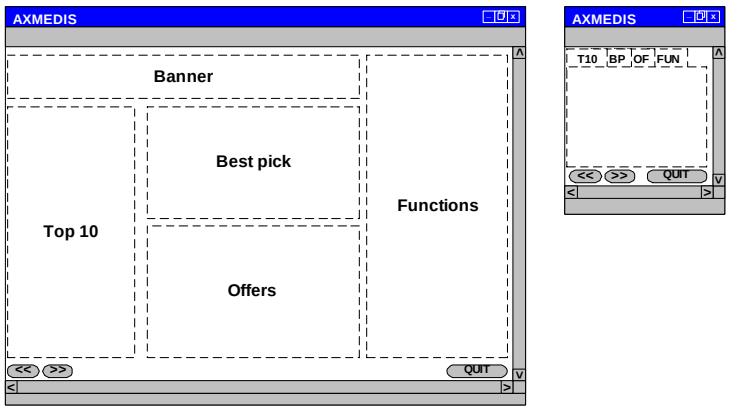


Catalogue creation and loading



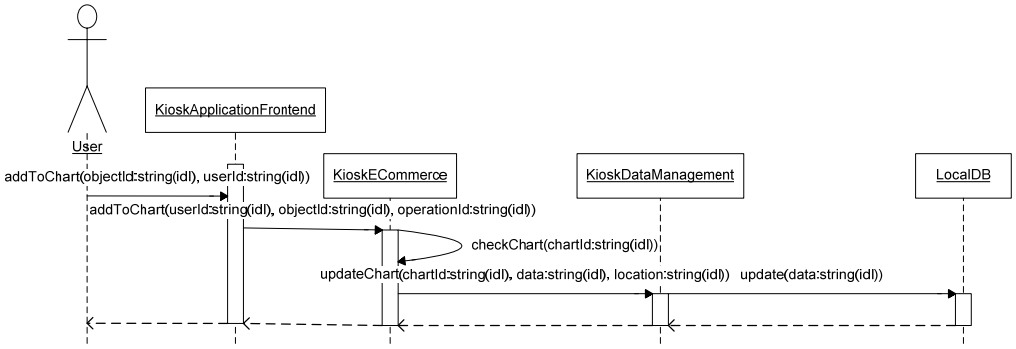
Catalogue browsing and content previewing

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

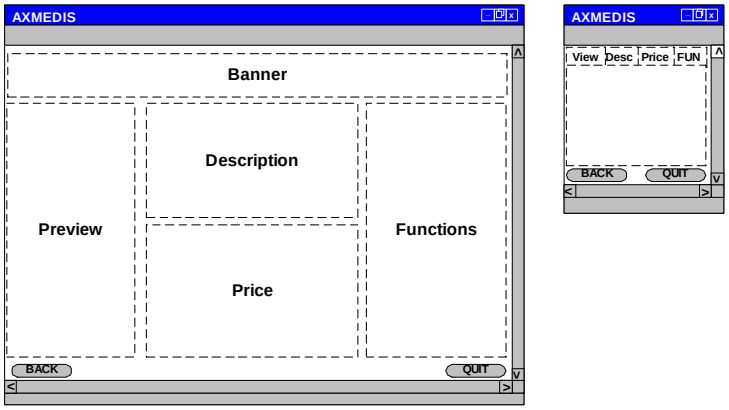


4.3.3.2 Content adding to chart

What follows 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 and the required grants on the 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.

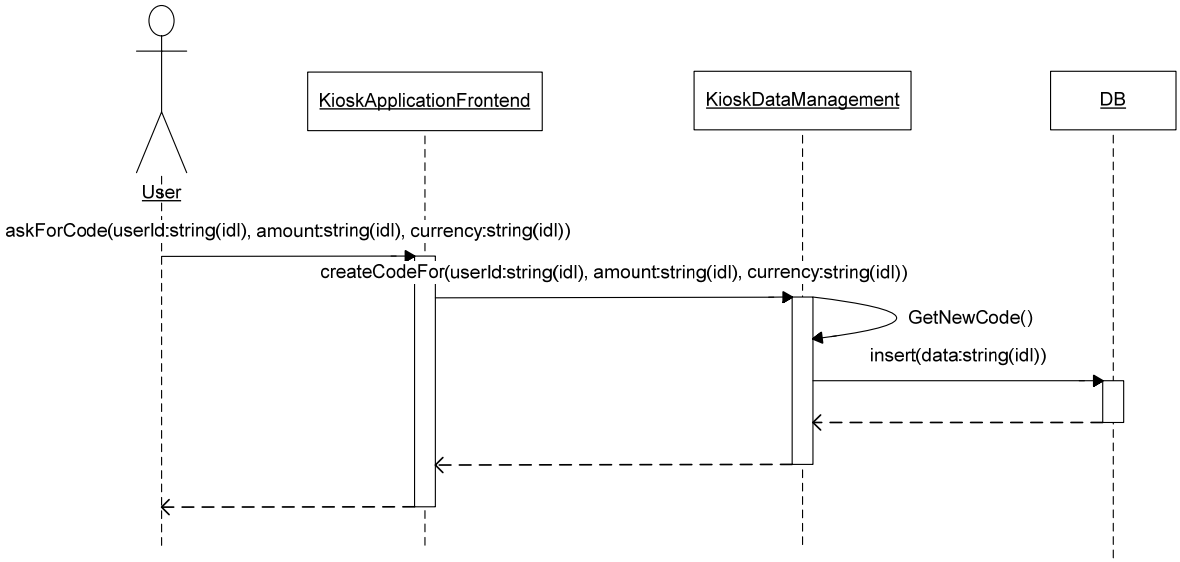


Content adding to chart

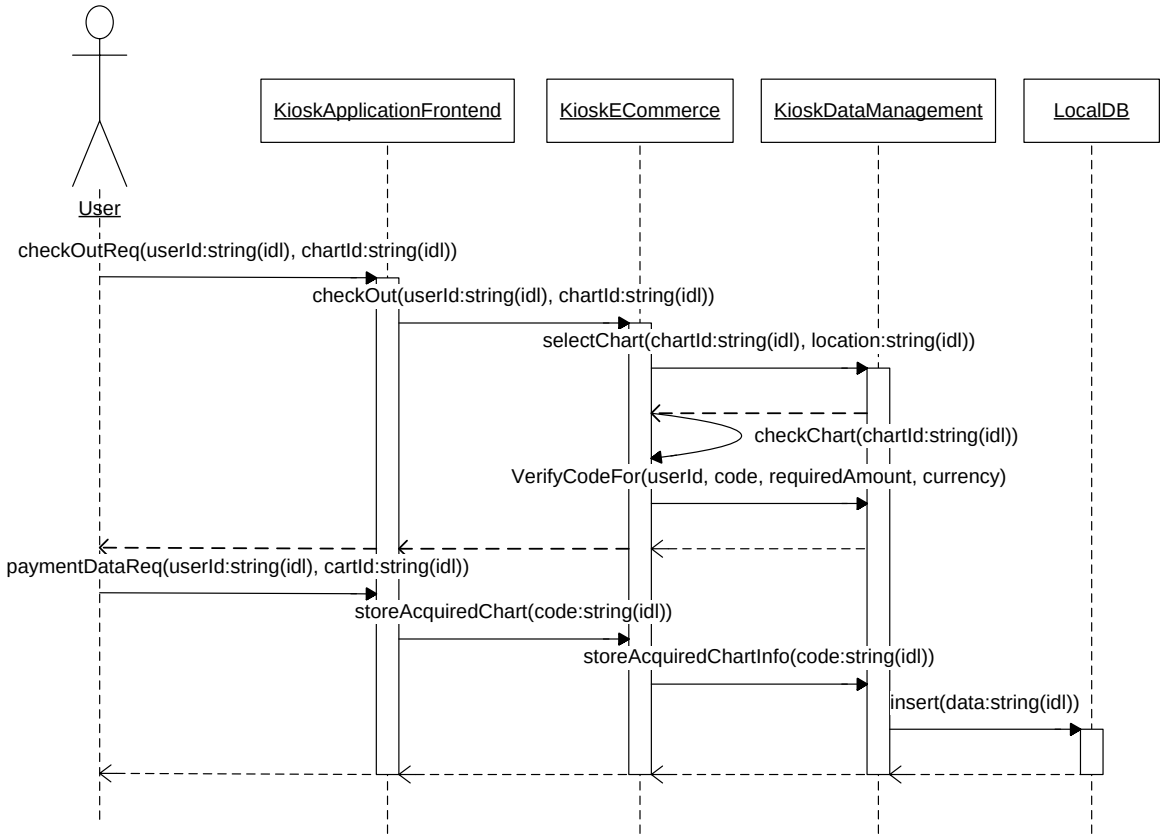


4.3.3.3 Content purchase

Once some contents have 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. First the user has to acquire a “code” (that could also be printed on a card) corresponding to a certain paid amount, using this code s/he is able to acquire grants over contents. Then the user will be able to initiate the check out procedure. The application front-end will pass the request to the e-commerce module that will retrieve the current chart. Once this is achieved 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) s/he will fill in the needed payment information, that is, the code he previously purchased. Once this data will be available and the user will confirm the acquisition will, the acquisition process will end and the checkout process will be initiating.

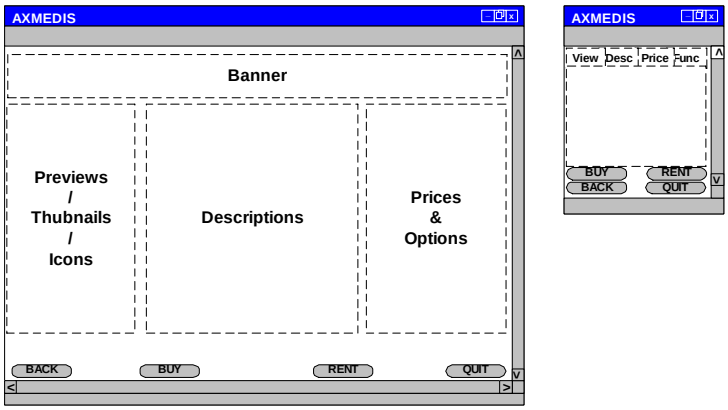


Code acquisition



Acquire

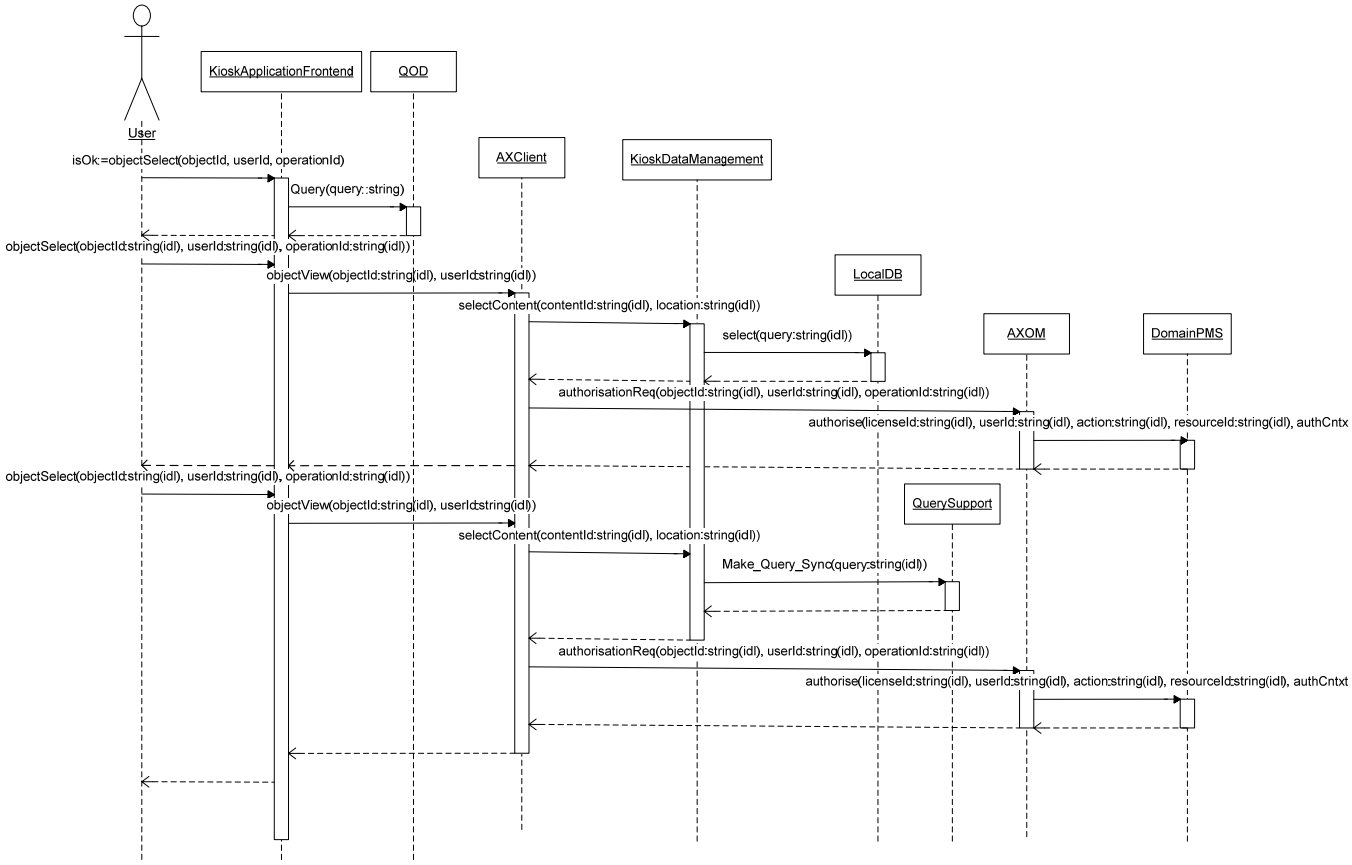
The front-end forwards all to the e-commerce module that proceeds to finalise the payment procedure with the third party managing financial issues. A possible aspect for the GUI related to this operation is reported in the following image where functional areas are placed in evidence.



4.3.3.4 Content delivery and fruition

The delivery module grants the user access to the contents for the grants for which he has paid. From the Kiosk application, the user can access a specific section allowing content fruition and download. In order to get contents formatted for specific device needs, the delivery module can adapt the content to the fruition device (if necessary) calling the Query on Demand module, which role is to manage the content retrieving and adaptation. The delivery module then retrieves device data (kind, storage, certificate...) and performs required authorization 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).. As the most complex case is the one of a delivery onto a user own device we will describe this instance.

At the Kiosk POPs, the content fruition is granted by the AXClient component, that is an AXMEDIS control (namely the AxActiveX), that has in charge the management of the contents according to the user's requests and grants, contacting the PMS. For the fruition on his proprietary device, the user has to install and setup the proper AXMEDIS Tool, which in turn will have to connect to AXCS and PMS components.

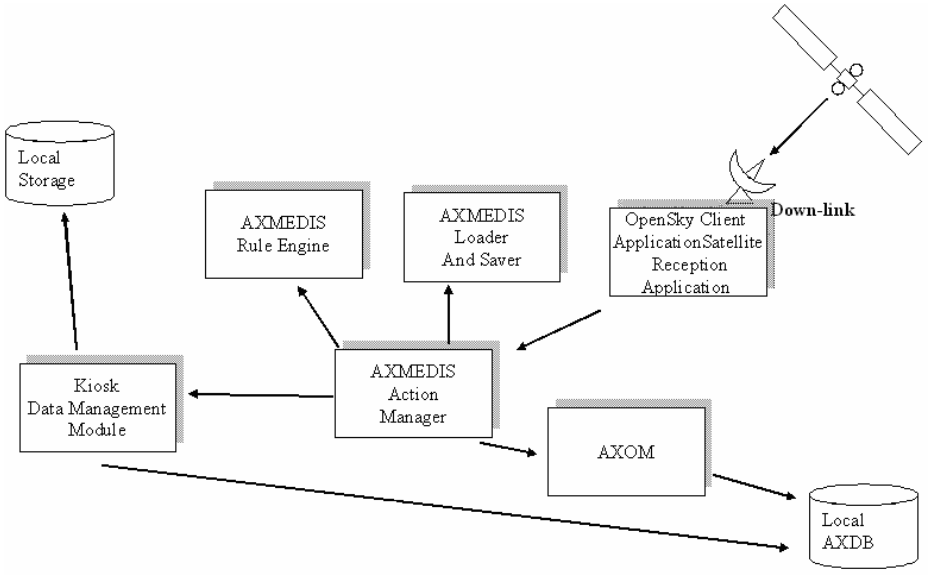


Fruition

4.3.4 System Maintenance Use Cases

Below System Maintenance Use Cases are presented: downloading by satellite, users and modules management, catalogue creation and uploading by satellite.

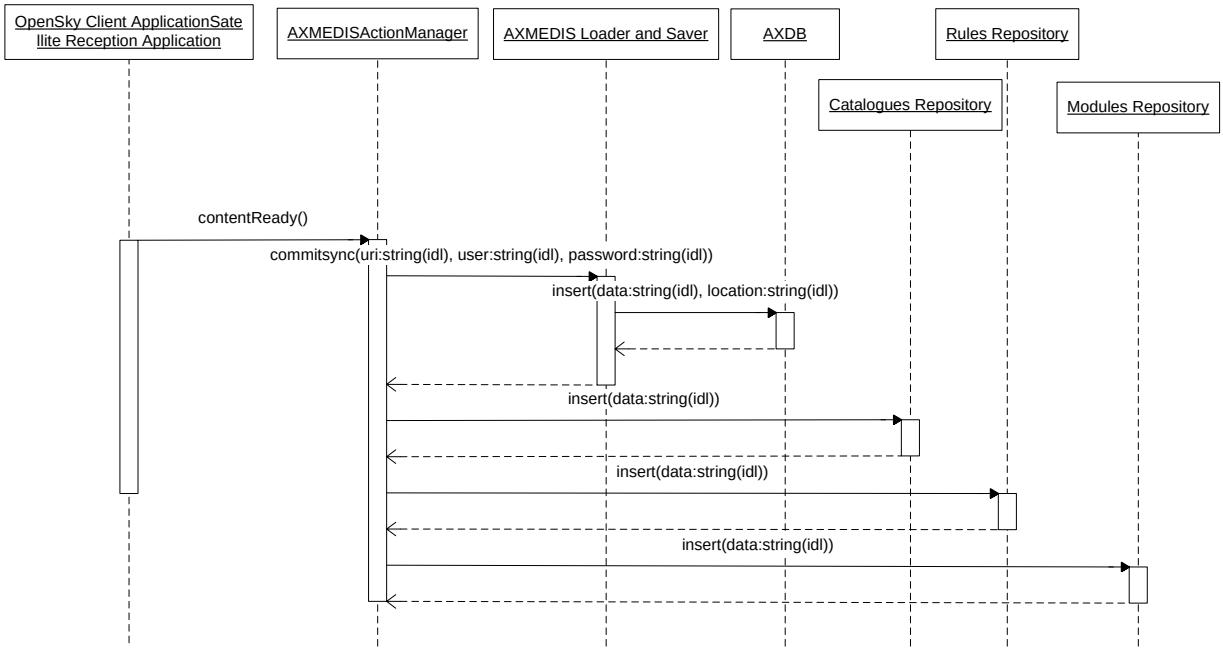
4.3.4.1 System maintenance



Content Download (via Satellite)

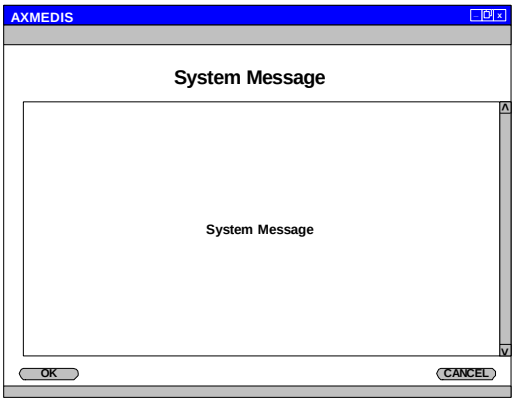
We assume that the starting point is when the checking time is over a Down-Link channel check has to be performed, the OpenSky Client Application Satellite Reception checks for data availability, we assume that data is available and therefore is downloaded and progressively cached locally till when all data to be received is stored locally (were it not a further check would be scheduled and the application would enter wait mode). All the data are AXMEDIS objects.
The OpenSky Client Application for Satellite Reception is the module in charge of receiving content/data from satellite. It covers all the functionalities required for the reception of data addressed to the specific receiving station: identification of the content addressed to the station, download of data, check of consistence on received data, and storage of objects into a local repository.
Once this step is over the AXMEDIS Action Manager retrieves the data from the local repository and checks to determine the most suitable action to perform on each object:

- if it is an AXMEDIS content, stores it on AXMEDIS DB using the Saver Web Service
- if it is a kiosk catalogue then sends it to the KioskFactory
- if it is a formatting rule, stores it into a rules repository
- if it is a module update, stores it into a modules repository

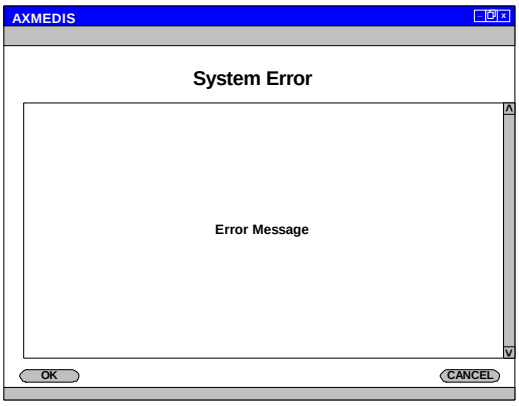


System Maintenance

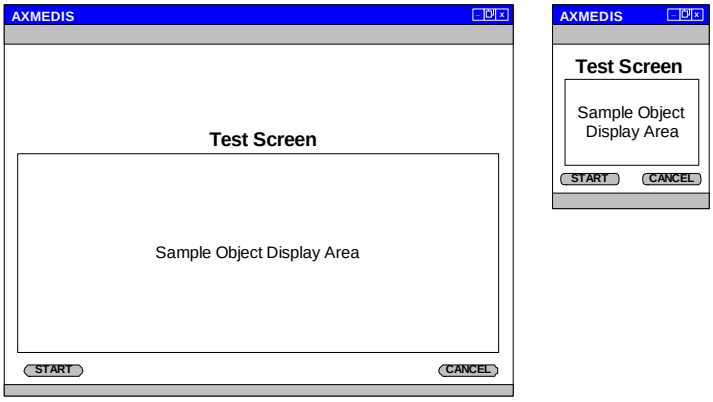
As previously mentioned there is also much side info that will need to be conveyed to the user during operation like messages, error notifications, test pages and so on. A possible aspect for the related GUI pages is reported in the following images.



Possible aspect of a system message page



Possible aspect of an error message page



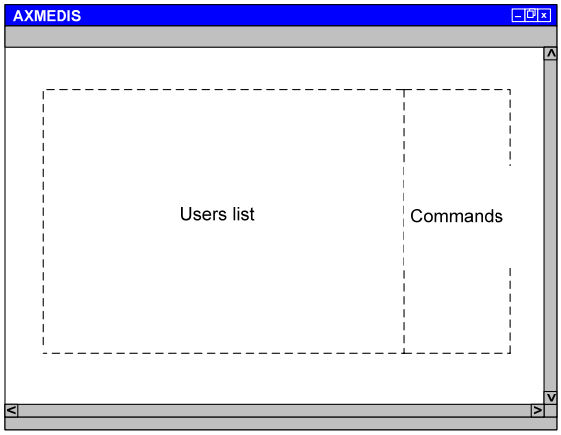
Possible aspect of a test page for both regular & PDA GUI

4.3.4.2 Load User

User Administrator can add a new user into the AXMEDIS system. Loading a new user corresponds to register him, so diagrams for this use case are not presented since they are very similar to the ones showed in Section 6.3.2.2, except that the actor is the User Admin and the user he is registering is not himself.

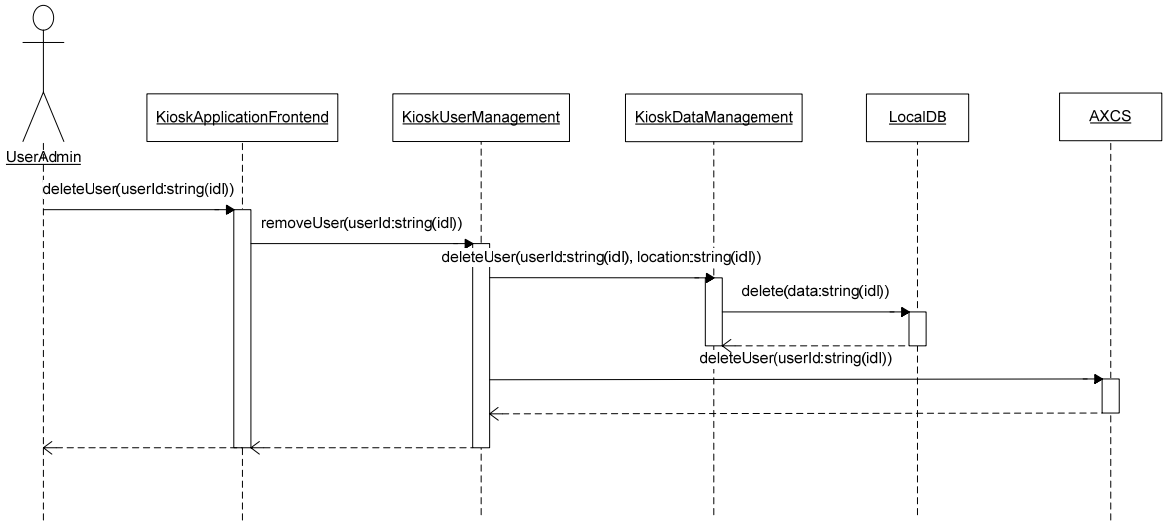
4.3.4.3 Modify User

When he wishes to update a user's profile, User Admin gets user data both from local and AXMEDIS databases, performs updates and then stores updated profile. Of course, User Admin has no direct access to database but interacts with front-end application; this, in turn, will interact with user management module, that will communicate to data management module in order to retrieve and store data into repositories. The temporal diagram for this use case is very close to the one reported in Section 4.3.2.4, so it is not reported again below.



4.3.4.4 Delete User

When he wants to deleter an user, User Admin interacts with front-end which will forward request to user management module. This module will communicate deleting request to data management that will delete user from Local DB and will request to AXCS to do the same on AXMEDIS database. Below related diagram is presented. Sketch of graphical user interface is not reported since it can be merged into update user interface presented above.



Delete user

4.3.4.5 Report User

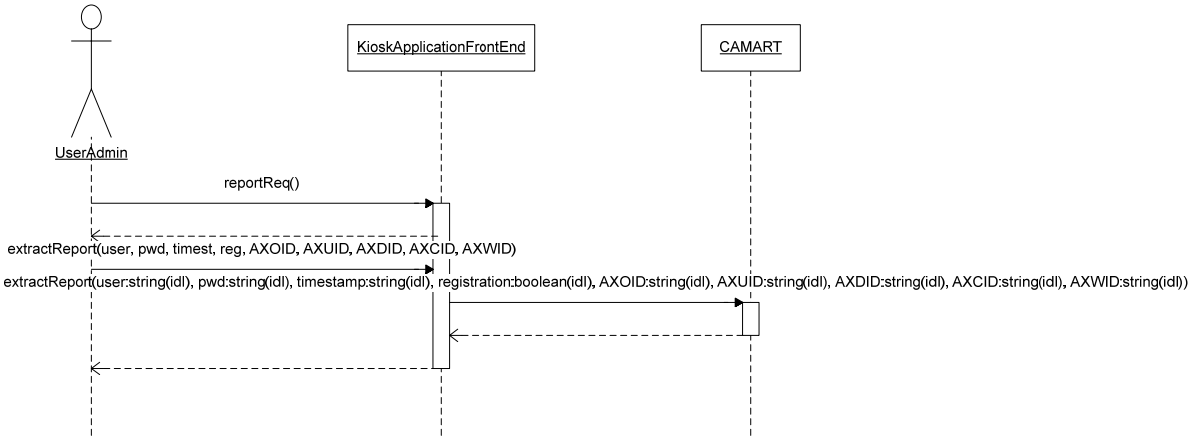
User Administrator can extract report information. In order to do so, he interacts with the Application front-end which in turn will interact with user management module to request report form. Once he has achieved it, User Admin can fill in fields depending on his search criteria and then send his query that will be propagated to CAMART, that is, Core Accounting Manager and reporting tool. CAMART can:

- get all the log
- get all the log after a certain registration/execution timestamp
- get all the log after a certain registration/execution timestamp that are related to:
    - an user
    - an object
    - a creator

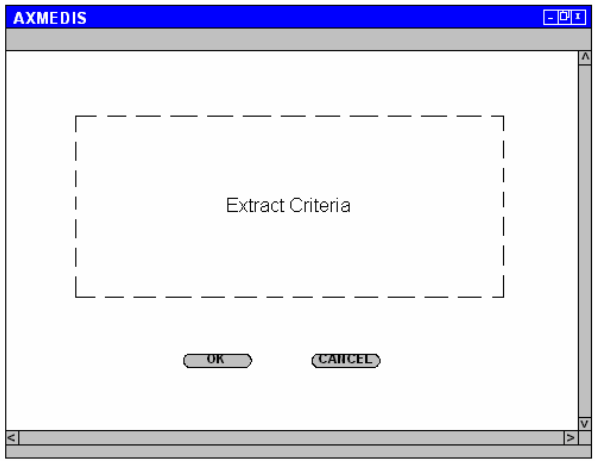
- a distributor
- a worked

- a combination in AND of the previous,

depending on user's selected search criteria. For further details about CAMART, please see Part H, Section 6 and DE9.1.4. Below Report User Temporal diagram and a sketch of user interface are reported.

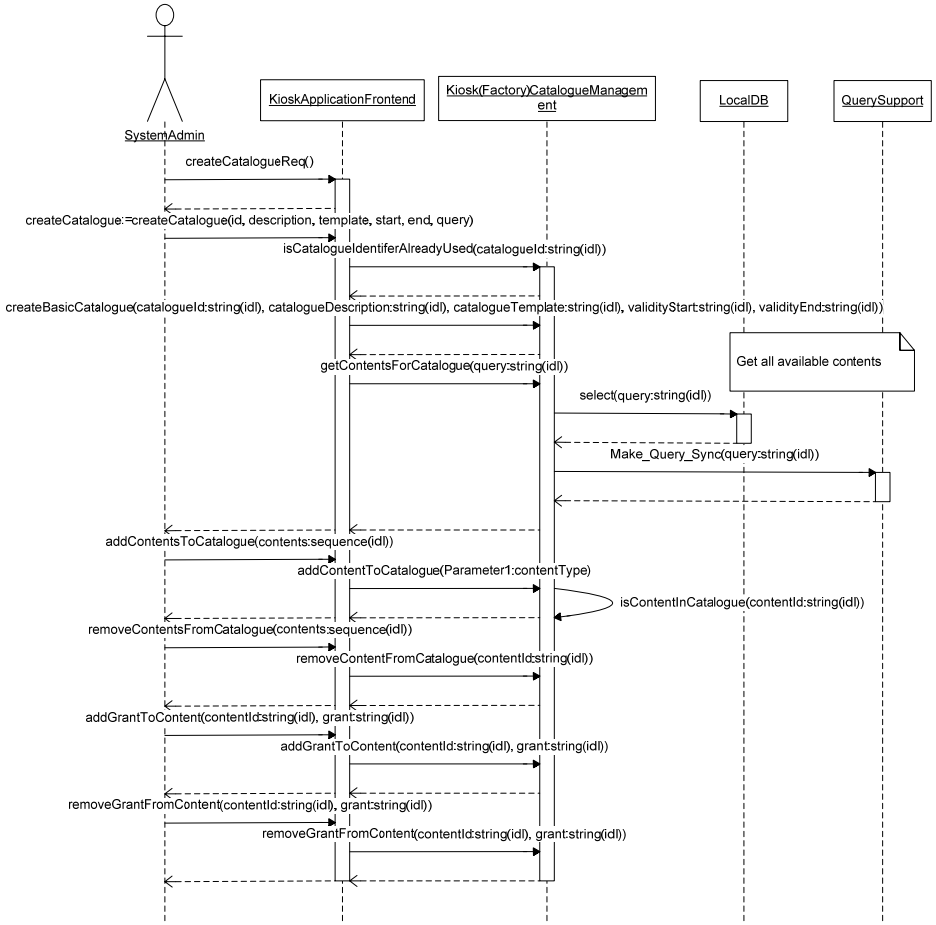


Report user

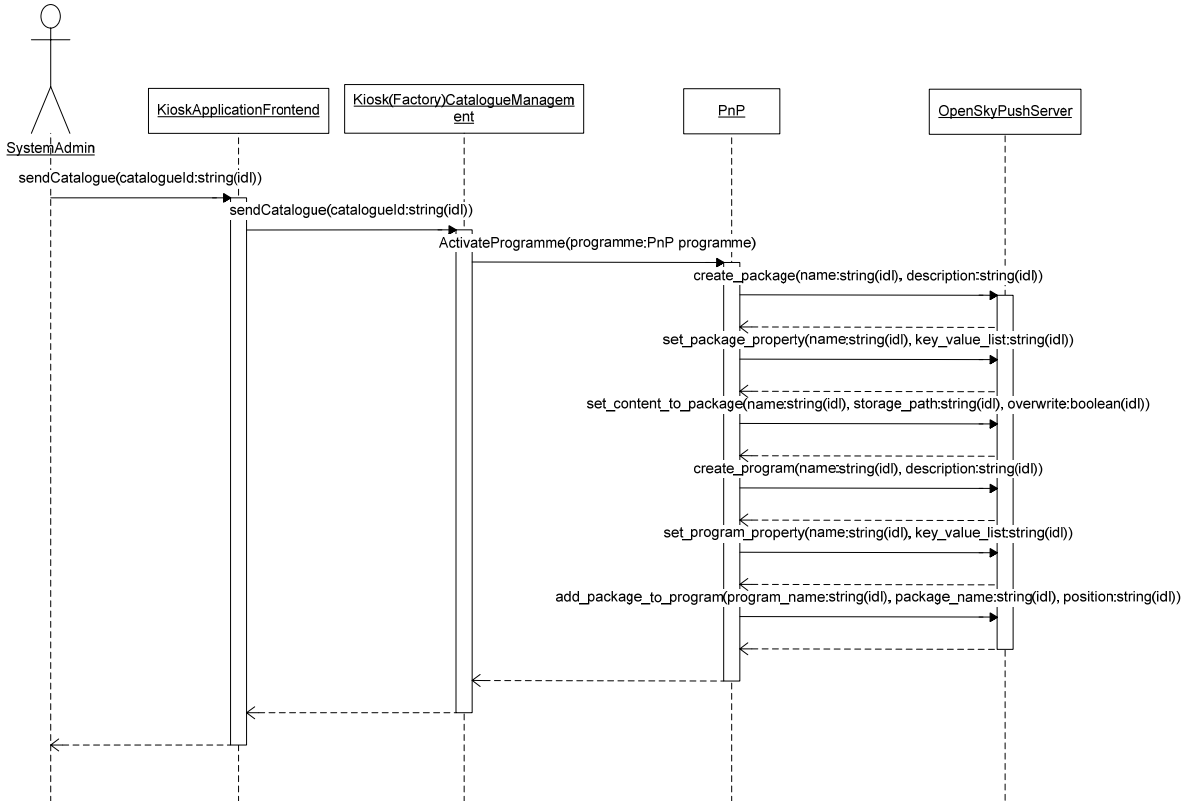


4.3.4.6 Create Catalogue

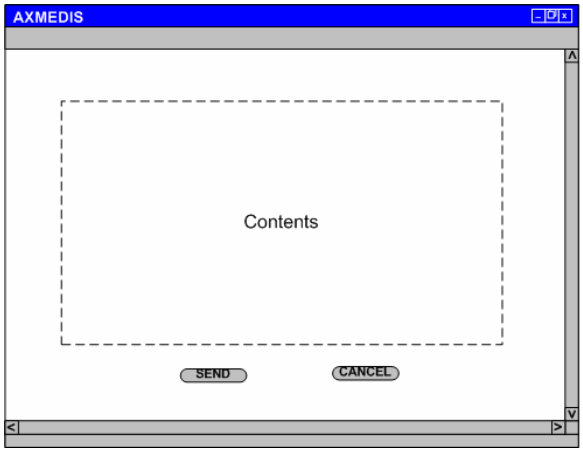
System Administrator interacts with Application front-end in order to ask to catalogue management module to create a new catalogue. So this module has to query both local and remote repositories to get all available contents and their metadata; once received these data, it puts them together in order to obtain a catalogue object. Catalogue object is a sequence of “content” objects: for each one, a list of information is presented, retrieved partially by AxInfo metadata, partially by Dublin Core metadata (e.g. title, description and format). If this operation succeedes, System Administrator can decide to send newly created catalogue to Satellite system. In this case, catalogue management module has to transform catalogue object into an OpenSky Package before sending it. In order to do so, it has to create a package, set package properties, add content to package, create a transmission program, set program properties and add package to program. All these details of the management of the package are hidden to the System Administrator since he uses the Programme and Publication AXMEDIs Tool with a proper programme that sends catalogue and contents to some selected Kiosks. Temporal diagram for catalogue creation and a sketch for user interface are shown below. PDA interface is not shown since catalogue creation is reserved to System Admin user so this task is foreseen for desktop only.



Catalogue creation

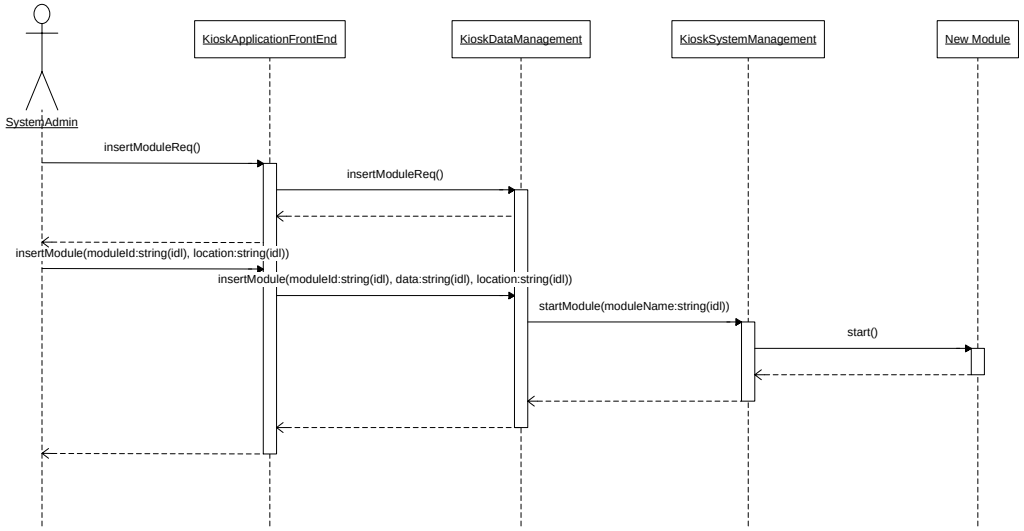


Catalogue sending

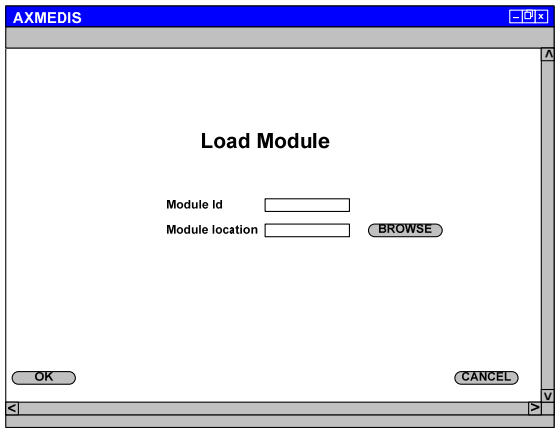


4.3.4.7 Load Module

System Administrator can load a new software module into the system. To do so, he uses front-end Application to tell which is the new module to data management module. This component will interact with system management in order to create and start wished module. Below temporal diagram for new module loading is presented. Then a sketch of a possible user interface is reported; PDA interface is not reported since Admin tasks are foreseen by desktop application only.

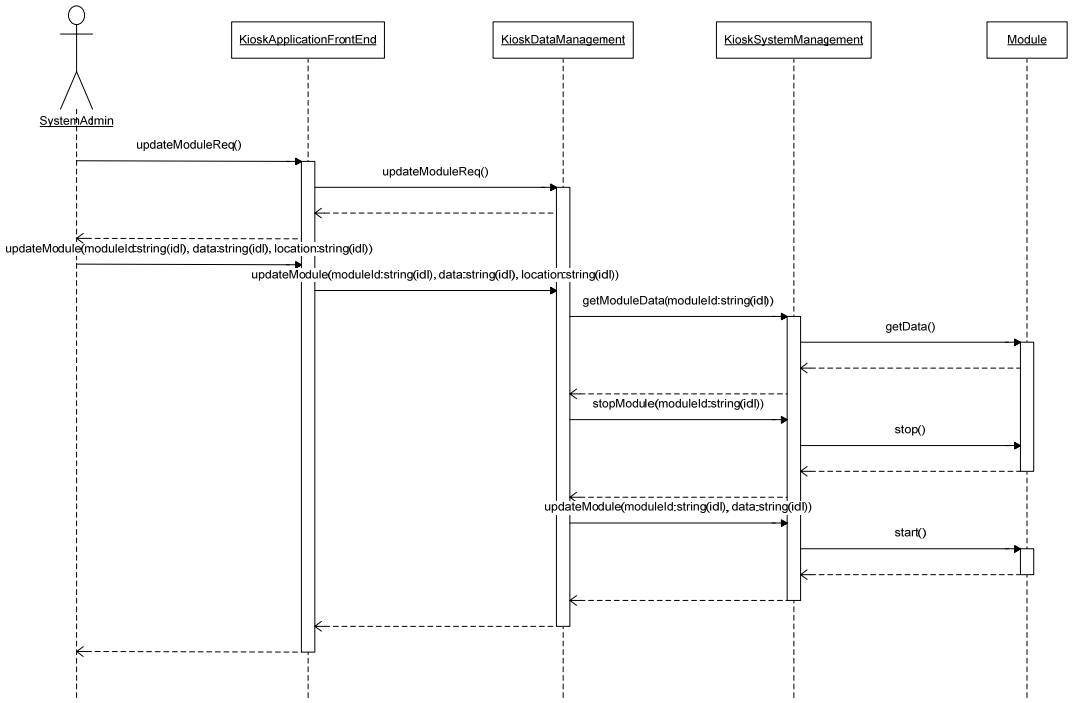


Load module



4.3.4.8 Update Module

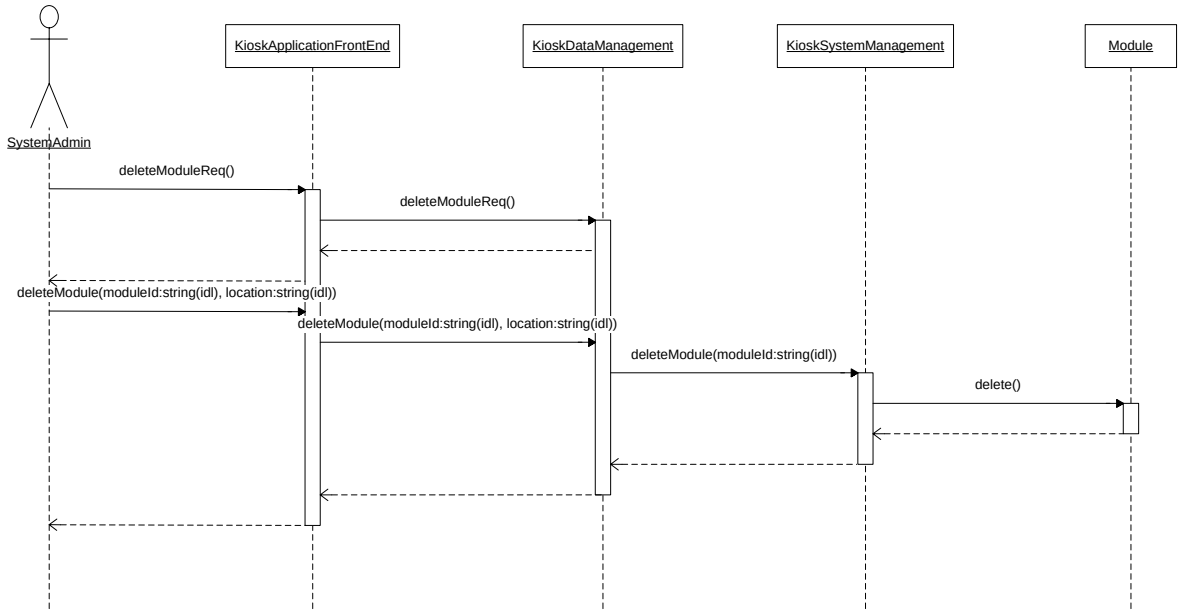
System Administrator can upload an existing software module. He uses front-end Application to tell new module data to data management module. This in turn will interact with system management to create a module backup, in order to be able to restore old module if there are problems activating new one. Then old module is stopped and substituted by new data. Below temporal diagram for module updating is presented. Possible user interface could be very similar to the one already presented for module loading, so it is not reported again.



Update module

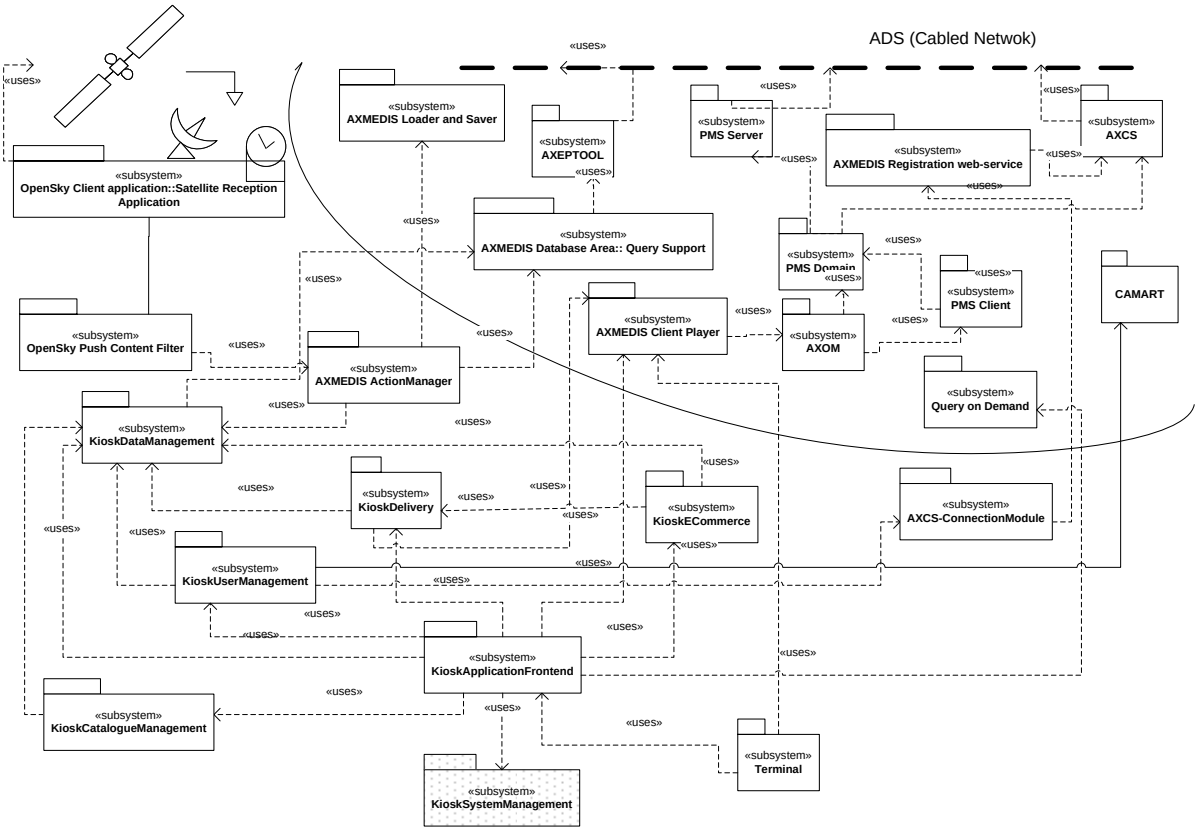
4.3.4.9 Delete Module

System Administrator can delete a software module from the system. To do so, he uses front-end to retrieve deleting form. He fills in module identifier and module location fields, then application front-end will forward data to data management which in turn will pass data to system management which is responsible of module deleting. Below temporal diagram for module deleting is presented. Possible user interface could be very similar to the one already presented for module loading, so it is not reported again.



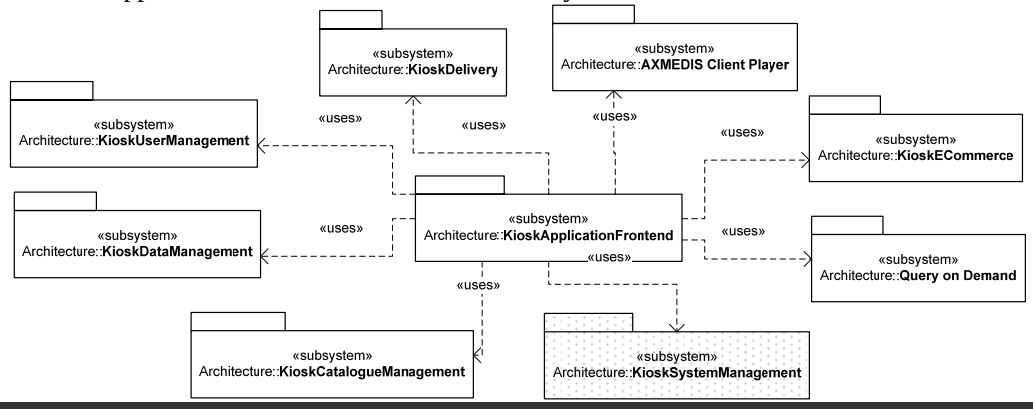
Delete module

4.4 Kiosk Server Architecture



4.4.1 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

Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDISActiveXControl	AXMEDISActiveXControl	ActiveX
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

4.4.1.1 Kiosk Application Front-end API

```
interface KioskApplicationFrontend
{
   boolean createDomain(in string domainId);

   string langSelect(in string langId);
   boolean langSetup(in string langId);

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

   string registrationReq();
   boolean registerUser(in string userData);
   string setUserProfile(in string userId, in string userProfile);
   string getUserProfile(in string userId);
   string updateUserProfile(in string userProfile);
   boolean deleteUser (in string userId);
   string reportReq();
   string extractReport(in string user, in string pwd, in string timest, in boolean reg, in string AXOID, in string AXUID,
          in string  AXDID, in string AXCID, in string AXWID);

   boolean checkData (in string data);

   string downloadClientInstr ();
   void downloadClient (in string deviceData);

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

   string askForCode(in string userId, in string amount, in string currency);
   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);

   string createCatalogue();
   boolean sendCatalogue(in string catalogueId);

   string insertModuleReq();
   boolean insertModule(in string moduleId, in string location);
   string updateModuleReq();
   boolean updateModule(in string moduleId, in string data, in string location);
```

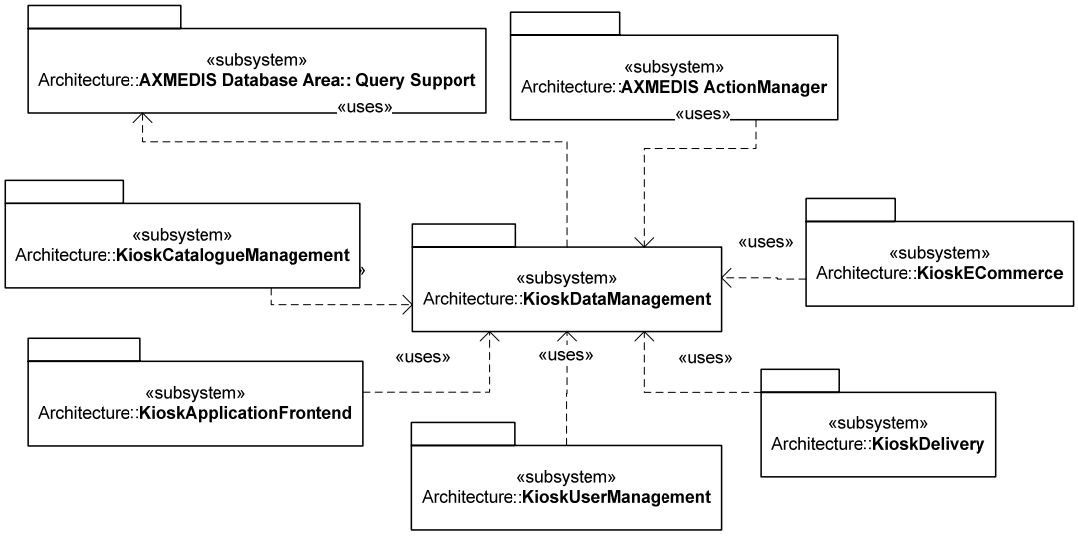
```
    string deleteModuleReq();
    boolean deleteModule(in string moduleId, in string location);
};
```

Function	Description	Method	Parameters	Reply
Domain creation	Allows the user to create and register a new domain	createDomain	string domainId	boolean createdDomain
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
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
User management	Update user profile data	updateUserProfile	string userProfile	string updatedProfile
User management	Delete user data	deleteUser	string userId	boolean deletedUser
User management	Initiate report user task	reportReq	-	string reportForm
User management	Extract user report information	extractReport	string user string pwd string timest boolean reg string AXOID string AXUID string AXDID string AXCID string AXWID	string reportData
Data check	Performs basic validity check on inserted data.	checkData	string data	boolean isOk
Download Client	Requires instructions to download AXMEDIS client application	downloadClientInstr	-	string downloadInstructions
Download Client	Download AXMEDIS client application	downloadClient	string deviceData	void
Load a catalogue	Retrieves a Catalogue object, in terms of its stored data.	loadCatalogue	string userId string catalogueId	boolean isOk
Browse catalogue Acquire objects	Allows the user to browse the catalogue pages according to a pre-defined template Gets a code after payment	browseCatalogue askForCode	string catalogueId string pageId string userId string amount string currency	string catalogPage string chartId
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 chartId
Asking for payment	Allows to initiate a checkout procedure	checkOutReq	string userId string chartId	string checkOutForm
Acquire objects Asking for payment data	Allows to input data needed for initiating a checkout Allows to require payment for objects currently in the chart	Checkout paymentDataReq	string checkOutForm string userId string chartId	string deliveryUrl string paymentDataForm
Download objects Create catalogue	Allows the user to start content delivery procedure Allows the system admin to create a new catalogue	contentDownload createCatalogue	string deliveryUrl -	object content string catalogueId

Function	Description	Method	Parameters	Reply
Create catalogue	Allows the system admin to send catalogue to satellite for broadcast	sendCatalogue	string catalogueId	boolean sentCatalogue
Modules management	Allows the system admin to initiate to load a new module	insertModuleReq	-	string loadForm
Modules management	Allows the system admin to load a new module	insertModule	string moduleId string location	boolean isOk
Modules management	Allows the system admin to initiate to update a module	updateModuleReq	-	string updateForm
Modules management	Allows the system admin to update a module	updateModule	string moduleId string data string location	boolean isOk
Modules management	Allows the system admin to initiate to delete a module	deleteModuleReq	-	string deleteForm
Modules management	Allows the system admin to delete a module	deleteModule	string moduleId string location	boolean isOk

4.4.2 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	ILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXActionManagerConnector	AXMEDIS Action Manager	API
Query Support Web Service Interface	AXMEDIS Query Support	Web Services
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

4.4.2.1 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);
   string insertModuleReq();
   string updateModuleReq();
   string deleteModuleReq();
   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 createCodeFor(in string userId, in string amount, in string currency);

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

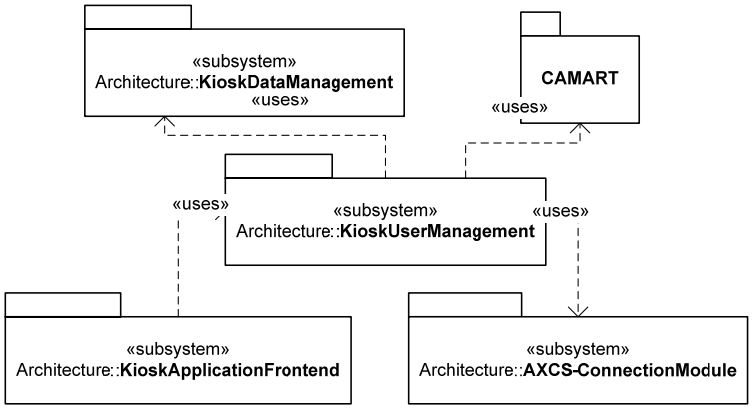
Function	**Description**	**Method**	**Parameters**	**Reply**
Select user data	Allows to select and retrieve user data from the specified location	selectUser	string userId string location	string data
Check user data	Allows to check user data in the specified location	checkUser	string userId string data 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	Allows to delete user data from the specified location	deleteUser	string userId string location	boolean isOk

Function	Description	Method	Parameters	Reply
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	Allows 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
Module management	Allows to get form for module loading	insertModuleReq	-	boolean loadForm
Module management	Allows to get form for module updating	updateModuleReq	-	boolean updateForm
Module management	Allows to get form for module deleting	deleteModuleReq	-	boolean deleteForm
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	Allows 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	Allows 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 charted 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	Allows to delete chart data from the specified location	deleteChart	string chartId string location	boolean isOk
Acquire code	Creates a new code after payment	createCodeFor	string userId string amount string currency	string code
Search by keyword	Initiates a search by specifying a list of keywords.	searchByKeyword	string keywordList string location	string data

4.4.3 Kiosk Server Architecture: User Management

In the following diagrams are reported the main operations related to the user management 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	ILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDIS Registration Web Service	AXMEDIS Registration	Web Services
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

4.4.3.1 User Management API

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

```
    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, in string userProfile);
    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
Create user - catalogue association	Establishes an association between a catalogue and a user (this is an extension)	createUserCatalogueAssociation	string userId string catalogueId	boolean isOk
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 retireves 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

4.4.3.2 User Management external connection

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

```
interface AxcsConnectionModule
{
   boolean connect();

   string Registration(in string nickName, in string password, in usrDatatype regdata);
};
```

Function	Description	Method	Parameters	Reply
Connect module	Try to connect to the module to check if it is on	connect	-	boolean isOk
Register User	Register the user on the AXMEDIS system	Registration	string nickName string password usrDatatype regdata	string result string definitive_UID

For more details on usrDatatype, please see Part H, Section 2.2.2, p.21ff. Here is a request sample message for invoking Registration AXCS method:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <m:Registration xmlns:m="http://new.webservice.namespace">
            <NickName>…distributor user…</NickName>
            <Password>…password…</Password>
            <Regdata>
                <AXUID> URN:AXMEDIS:00001:USR:00000000-0000-0000-0000-000000000001</AXUID>
                <AXDOM>0V2X4XV75BD1SS456XC5P98345BD4L05</AXDOM>
                <email>luxor@unknown.com</email>
                <NickName>luxor27</NickName>
                <Password>…password…</Password>
                <nationality>French</nationality>
                <PubKey>…public key…</PubKey>
                <RegDate>2005-02-05T14:03:10</RegDate>
                <RegDeadline>2007-02-05T14:03:10</RegDeadline>
                <Status>U</Status>
                <B2BUserID> </B2BUserID>
                <Website></Website>
                <RefName></RefName>
                <Phone></Phone>
                <Company></Company>
                <CmpAddress></CmpAddress>
                <CmpPhone1></CmpPhone1>
                <CmpPhone2></CmpPhone2>
                <CmpFax></CmpFax>
                <Email></Email>
                <NickName></NickName>
                <Password></Password>
                <Location></Location>
                <Nationality></Nationality>
                <PubKey></PubKey>
                <AXCID></AXCID>
                <AXDOM></AXDOM>
                <B2BUserID></B2BUserID>
                <RegDate></RegDate>
                <RegDeadline></RegDeadline>
                <Status></Status>
                <AXDID></AXDID>
                <AXDOM></AXDOM>
                <B2BUserID></B2BUserID>
                <RegDate></RegDate>
                <RegDeadline></RegDeadline>
                <Status></Status>
                <AXAID></AXAID>
                <AXDOM></AXDOM>
                <B2BUserID></B2BUserID>
                <PubKey></PubKey>
                <RegDate></RegDate>
                <RegDeadline></RegDeadline>
                <Status></Status>
                <AXSID></AXSID>
```

```
                <AXDOM></AXDOM>
                <B2BUserID></B2BUserID>
                <NationDomain></NationDomain>
                <RegDate></RegDate>
                <RegDeadline></RegDeadline>
                <Status></Status>
                <AXPID></AXPID>
                <AXDOM></AXDOM>
                <B2BUserID></B2BUserID>
                <RegDate></RegDate>
                <RegDeadline></RegDeadline>
                <Status></Status>
            </Regdata>
        </m:Registration>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

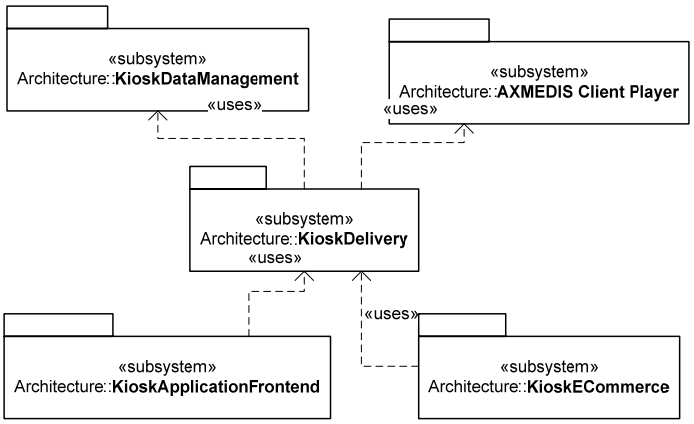
Here is a response sample message received after invoking Registration AXCS method:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <m:Registration xmlns:m="http://new.webservice.namespace">
            <Result>0</Result>
            <definitive_UID> URN:AXMEDIS:00001:USR:00000000-0000-0000-0000-000000000001</definitive_UID>
        </m:Registration>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

For more details on AXCS Registration, please see Part H, Section 2.2.2.

4.4.4 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) Single Thread or Multithread	Static Library Multithread
Language of Development	.NET framework (c#)
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.)
AXMEDISClientPlayer	AxActiveX	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

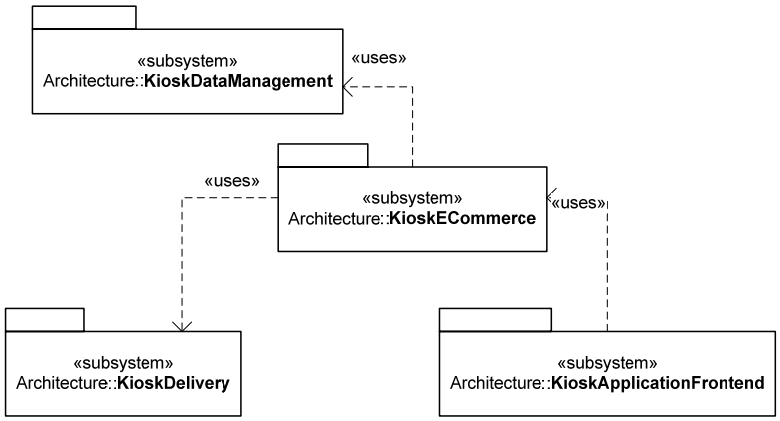
4.4.4.1 Delivery Module API

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

};
```

Function	Description	Method	Parameters	Reply
Delivery	Allows the system to start the delivery process	initiateDelivery	string userId string deliveryUrl	boolean isOk

4.4.5 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	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.)
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

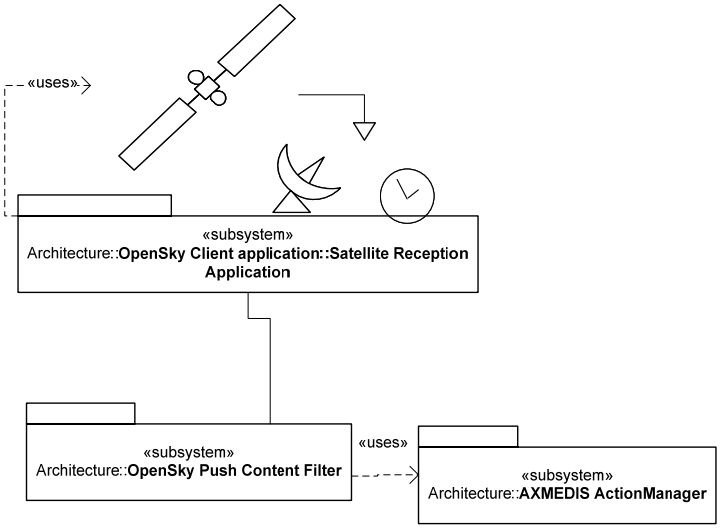
4.4.5.1 e-Commerce Module API

```
interface KioskECommerce
{
   string addToChart(in string userId, in string objectId, in string operationId);
   boolean clearChart(in string chartId);
   boolean removeFromChart(in string chartId, in string objectId, in string operationId);

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

   string paymentDataReq(in string userId, in string chartId);
   boolean paymentConfirm(in string userId, in string chartId);
   boolean storeAcquiredChart(in string chartId);
};
```

Function	Description	Method	Parameters	Reply
Add object to chart	Add an AXMEDIS object to the user chart	addToChart	string userId string objectId string operationId	string chartId
Clear chart	Clear the user chart	clearChart	string chartId	boolean isOk
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
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
Acquire	Store data	storeAcquiredChart	string chartId	boolean isOk

4.4.6 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	ILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
Programme and Publication Module		
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

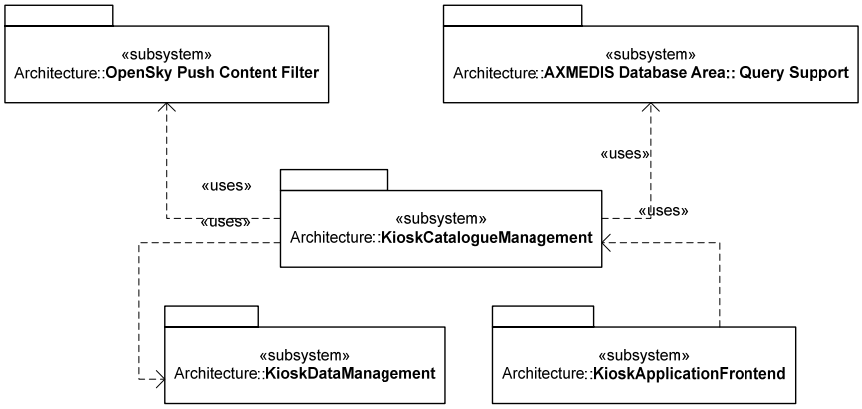
4.4.6.1 Satellite Reception API

```
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

4.4.7 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	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.)
Query Support Web Service Interface	AXMEDIS Query Support	Web Services
OpenSky Push Server	OpenSky Push Server	API through HTTPS protocol
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

4.4.7.1 Catalogue Management interfaces description

```
interface KioskCatalogueManagement
{
   string listCatalogue ();
      catalogueType createBasicCatalogue(in string catalogueId, in string catalogueDescr,  in string catalogueTempl, in string
   catalogueStart, in string catalogueEnd) ;
   boolean updateCatalogue(in string catalogueId, in string catalogueTemplate, in string catalogueData);
   boolean deleteCatalogue(in string catalogueId);
   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);

   boolean sendCatalogue(in string catalogueId);

   catalogueType addContentToCatalogue( in Content contentToAdd) ;
   bool isContentInCatalogue(in string contentid ) ;
  catalogueType removeContentFromCatalogue( in string contentId ) ;
   catalogueType addGrantToContent( in string contentId, in Grant grantToAdd);
   catalogueType removeGrantFromContent(in string contentId, in Grant grantToRemove);
   bool isCatalogueIdentiferAlreadyUsed(in string catalogueId);
   array getContentsForCatalogue(in string queryForQS);
   string buildQueryFromAxoid(in string axoid);
   content getContentFromAxoid(in string axoid);

};
```

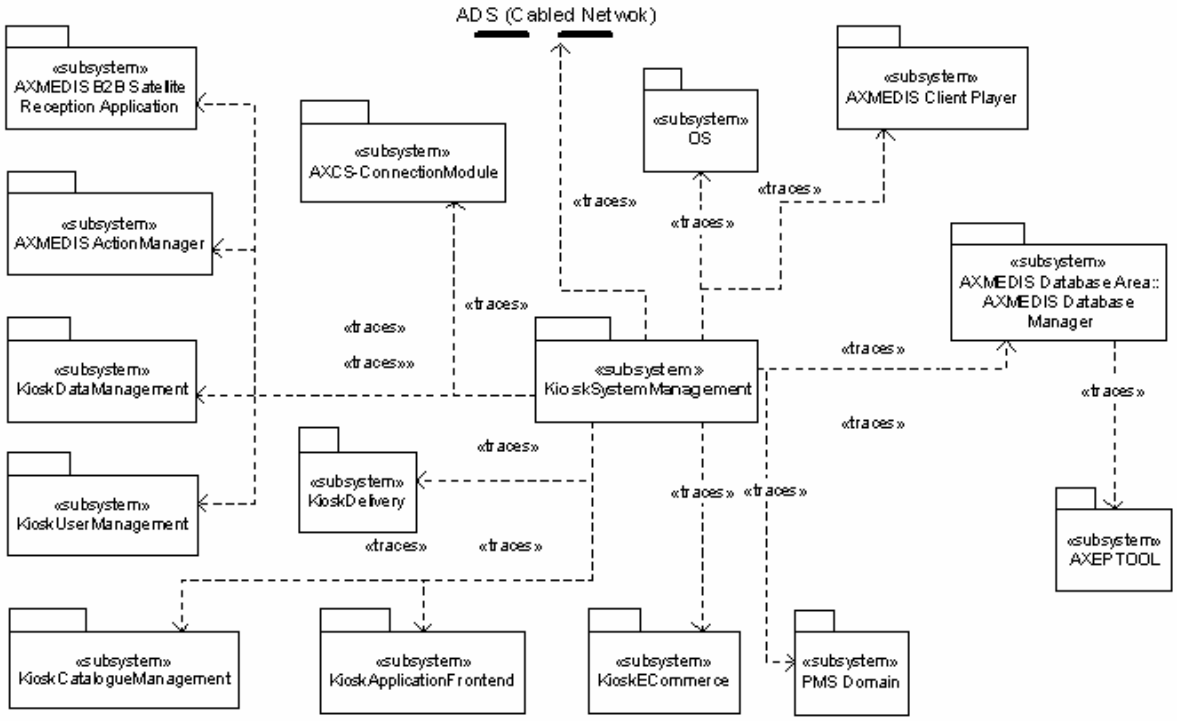
4.4.7.2 Catalogue Management API

Function	Description	Method	Parameters	Reply
List catalogue	Allows to retrieve the list of available catalogues	listCatalogue	-	string catalogueList
Catalogue creation	Create an empty catalogue with some information	createBasicCatalogue	string catalogueId string catalogueDescription string catalogueTemplate string validitystart string validityEnd	catalogueType catalogue
Catalogue update	Allows to create a catalogue starting from available objects and catalogue template	updateCatalogue	string catalogueId string catalogueTemplate string catalogueData	boolean isOk
Delete catalogue	Delete a catalogue object associated with a given ID.	deleteCatalogue	string catalogueId	boolean isOk
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 catalogePage
Catalogue loading	Allows to lad a specific catalogue page into memory for subsequent management	loadCataloguePage	string catalogueId string pageId	string catalogePage
Catalogue display	Allows to display a selected catalogue page	displayCataloguePage	string catalogueId string pageId	boolean isOk
Catalogue creation	Allows to send a catalogue to satellite server	sendCatalogue	string catalogueId	boolean sentCatalogue
Catalogue creation	Allows to add a content to the catalogue	addContentToCatalogue	content contentToAdd	catalogueType catalogue
Catalogue creation	Tells if a content has been already added to current catalogue	isContentInCatalogue	string contentid	boolean isContentInCat
Catalogue creation	Removes a content from the catalogue	removeContentFromCatalogue	string contentId	catalogueType catalogue

Function	Description	Method	Parameters	Reply
Catalogue creation	Adds a grant to a content into the catalogue	addGrantToContent	string contentId Grant grantToAdd	catalogueType catalogue
Catalogue creation	Remove a grant from a content into the catalogue	removeGrantFromContent	string contented Grant grantToRemove	catalogueType catalogue
Catalogue creation	Tells if a catalogue identifier is already in use	isCatalogueIdentiferAlreadyUsed	string catalogueId	boolean isUsed
Catalogue creation	Retrieves contents using given query	getContentsForCatalogue	string queryForQS	array contents
Catalogue creation	Build a query to send to Query Support to retrieve a content with given axoid	buildQueryFromAxoid	string axoid	string query
Catalogue creation	Returns a content having specified axoid	getContentFromAxoid	string axoid	content acontent

4.4.8 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	ILABS	
Status (proposed/approved)	Proposed	
Platforms supported	Microsoft Windows	
Interfaces with other tools:	Name of the communicating tools	Communication model and format (protected or not, etc.)
AXMEDISDomainPMSConnector	AXMEDIS Domain PMS	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
AXDBConnector	AXMEDIS DataBase Manager	Web Services or API
PMS Domain Home	PMS Domain Home	Web Services
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

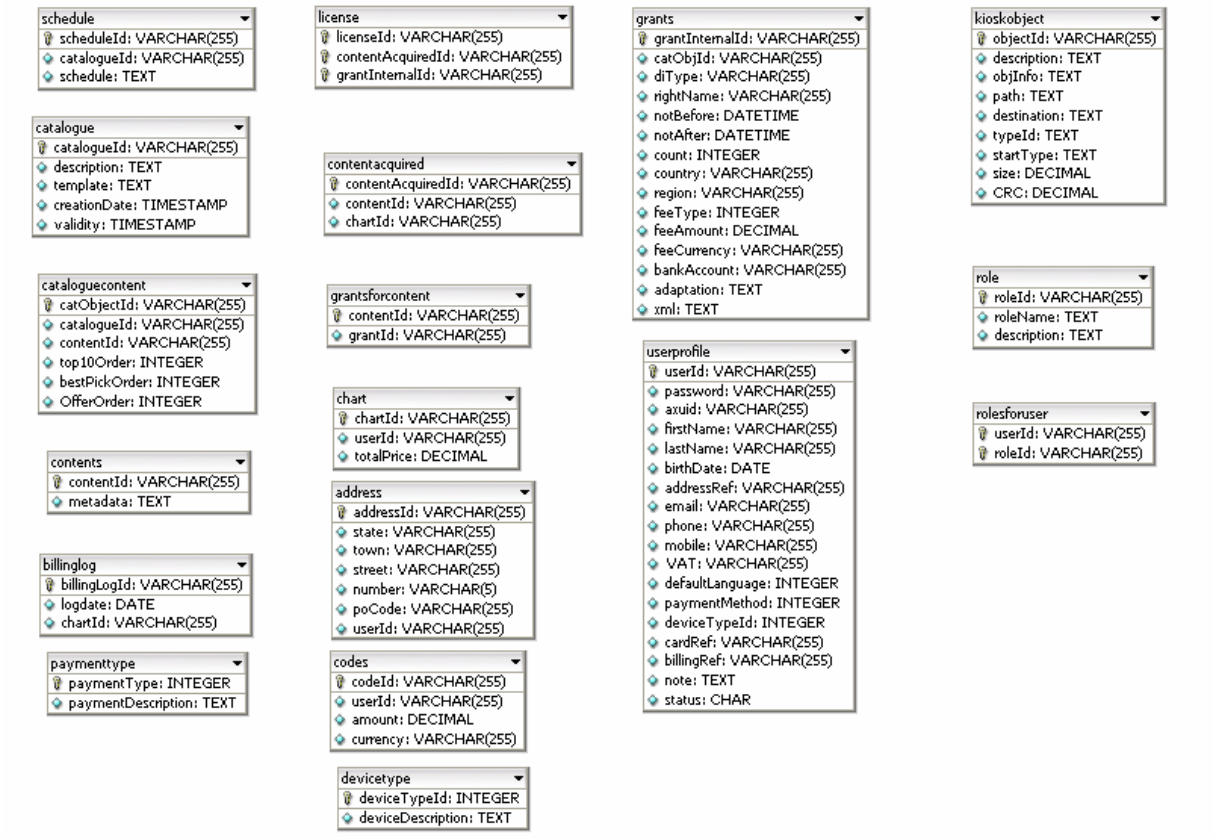
4.4.8.1 System Management API

```
interface KioskSystemManagement
{
   string startModule(in string moduleName);
   string checkModule(in string moduleId);
   boolean stopModule(in string moduleId);
   string getModuleData(in string moduleId);
   boolean updateModule(in string moduleId, in string data);
   boolean deleteModule(in string moduleId);

   boolean createDomain (in string domainId);
   boolean checkDomainId (in string domainId);
   Domain prepareDomain (in string AXOM, in string AXID, in string typeOfID);

   string downloadClientInstr ();
   void downloadClient (in string deviceData);
};
```

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
Get a module data	Allows the system manager to retrieve a module data	getModuleData	string moduleId	string moduleData
Update a module	Allows the system manager to update a module	updateModule	string moduleId string data	boolean isOk
Delete a module	Allows the system manager to delete a module	deleteModule	string moduleId	boolean isOk

Function	Description	Method	Parameters	Reply
Domain creation	Allows the user to create and register a new domain	createDomain	string domainId	boolean createdDomain
Domain creation	Checks if domain identifier complies with naming convention	checkDomainId	string domainId	boolean domainIdOK
Domain creation	Creates a new domain	prepareDomain	string AXOM string AXID string typeOfID	Domain domain
Download Client	Requires instructions to download AXMEDIS client application	downloadClientInstr	-	string downloadInstructions
Download Client	Download AXMEDIS client application	downloadClient	string deviceData	void

4.5 Database Entity Relationship



4.6 Data Types

What follows is the description of the main data structures used at the kiosk.

4.6.1 User object

The following table describes the user related data structure as devised within the kiosk application. Most of the personal data are considered optional: if the user would not like to store this info during the registration phase, yet he will be able to provide them later or never, and also to delete the information previously stored.

Attribute	Type	Description
firstName	String	User first name
lastName	String	User last name
userId	String	Username
password	String	User password
roles	String	{role, role…} Set of allowed roles for the specified user encompassing: Administrator, End-user…
defaultLanguage	Int	The default language for the GUI and content (the default value will be English)
birthDate	Date	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

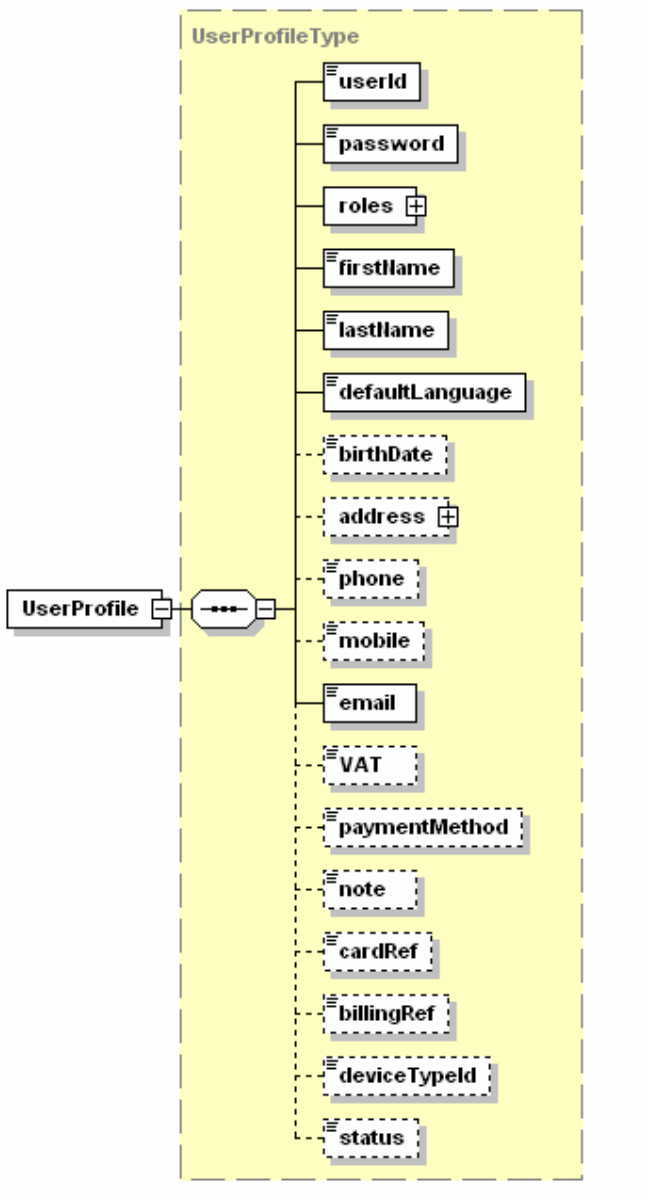
Attribute	Type	Description
email	String	User e-mail address
VAT paymentMethod	String Int	User VAT code number Preferred payment method: 0 = Pre-paid-cards / 1 = Credit card / 2 = Other
cardRef	String	Reference to a temporary structure holding user credit
billingRef	String	Reference to a temporary structure holding user chosen billing info
deviceTypeId note status	Int String String	0 = PDA / 1 = Smartphone / 2 = Other Note about user User status: B = Blocked / U = Unblocked

4.6.1.1 XML Representation of a User Profile

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="UserProfile" type="UserProfileType"/>
    <xs:complexType name="UserProfileType">
        <xs:sequence>
            <xs:element name="userId" type="xs:string"/>
            <xs:element name="password" type="xs:string"/>
            <xs:element name="roles" type="rolesType"/>
            <xs:element name="firstName" type="xs:string"/>
            <xs:element name="lastName" type="xs:string"/>
            <xs:element name="defaultLanguage" type="xs:int"/>
            <xs:element name="birthDate" type="xs:string" minOccurs="0"/>
            <xs:element name="address" type="addressType" minOccurs="0"/>
            <xs:element name="phone" type="xs:string" minOccurs="0"/>
            <xs:element name="mobile" type="xs:string" minOccurs="0"/>
            <xs:element name="email" type="xs:string"/>
            <xs:element name="VAT" type="xs:string" minOccurs="0"/>
            <xs:element name="paymentMethod" minOccurs="0">
                <xs:simpleType>
                    <xs:restriction base="xs:int">
                        <xs:enumeration value="0"/>
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="note" type="xs:string" nillable="true" minOccurs="0"/>
            <xs:element name="cardRef" type="xs:string" minOccurs="0"/>
            <xs:element name="billingRef" type="xs:string" minOccurs="0"/>
            <xs:element name="deviceTypeId" minOccurs="0">
                <xs:simpleType>
                    <xs:restriction base="xs:int">
                        <xs:enumeration value="0"/>
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="status" minOccurs="0">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="B"/>
                        <xs:enumeration value="U"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="rolesType">
        <xs:sequence>
            <xs:element name="roleId" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="addressType">
        <xs:sequence>
            <xs:element name="addressState" type="xs:string" minOccurs="0"/>
            <xs:element name="addressTown" type="xs:string" minOccurs="0"/>
            <xs:element name="addressStreet" type="xs:string" minOccurs="0"/>
            <xs:element name="addressNumber" type="xs:string" minOccurs="0"/>
```

```
            <xs:element name="addressPoCode" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

diagram



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

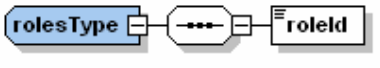
type **pin:UserProfileType**

children **userId password roles firstName lastName defaultLanguage birthDate address phone mobile email VAT paymentMethod note cardRef deviceTypeId status**

description This element is the root element containing information of the user profile. It contains all necessary information to manage users personal data. The fields are:

- **userId** username
- **password** user password
- **roles** user roles within the system
- **firstName** user first name
- **lastName** user last name
- **defaultLanguage** user default language for GUI and content (default value will be “English”)
- **birthDate** user date of birth
- **address** user address
- **phone** user telephone number
- **mobile** user mobile telephone number
- **email** user e-mail address
- **VAT** user VAT code
- **paymentMethod** user payment preferred method: 0 = Pre-paid-cards / 1 = Credit card / 2 = Other
- **cardRef** reference to a temporary structure holding user chosen credit card(s)
- **billingRef** reference to a temporary structure holding user chosen billing info
- **deviceTypeId** 0 = PDA / 1 = Smartphone / 2 = Other
- **status** user status within the system: B = Blocked / U = Unblocked

diagram



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

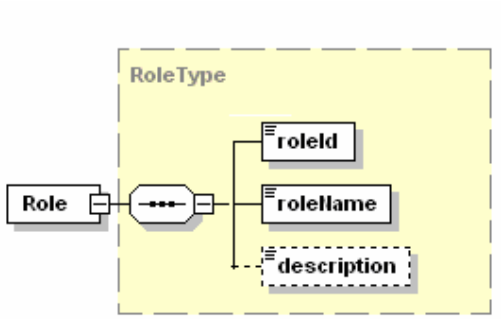
type **pin:rolesType**

children **roleId**

description This element contains all information about the roles of an user. The fields are one or more:

- **roleId** reference to user’s role

diagram



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

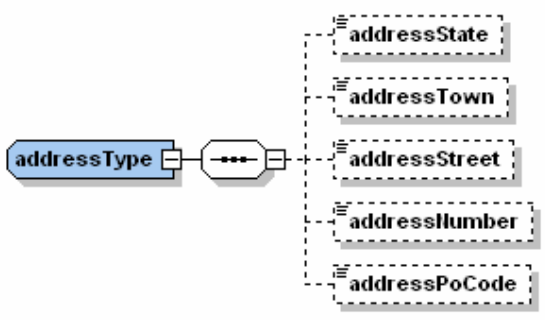
type **pin:RoleType**

children **roleId roleName description**

description This element contains all information about a system role. The fields are one or more:

- **roleId** role’s unique identifier in AXMEDIS system
- **roleName** role’s name
- **description** role’s verbouse description

diagram



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

type **pin:addressType**

children **addressState addressTown addressStreet addressNumber addressPoCode**

description This element contains all information about the mail address of an user. The fields are:

- **addressState** user's preferred mail address (state)
- **addressTown** user's preferred mail address (town)
- **addressStreet** user's preferred mail address (street)
- **addressNumber** user's preferred mail address (number)
- **addressPoCode** user's preferred mail address (postal area code)

4.6.2 Code object

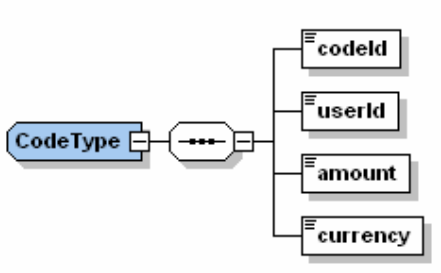
The following table describes the temporary data structure used to hold user credit card info during the billing and check-out operations.

Attribute	Type	Description
codeId	String	User card number
userId	String	Reference to owner user
amount	String	Money anount
currency	String	Currency

4.6.2.1 XML Representation of a User Code

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="Code" type="CodeType"/>
    <xs:complexType name="CodeType">
        <xs:sequence>
            <xs:element name="codeId" type="xs:string"/>
            <xs:element name="userId" type="xs:string"/>
            <xs:element name="amount" type="xs:string"/>
            <xs:element name="currency" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

diagram



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

type **pin:CodeType**

children **codeId userId amount currency**

description This element contains all information about the card of an user. The fields are:

- **codeId** user's card identifier
- **userId** user's reference
- **amount** money amount
- **currency** money currency

4.6.3 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.

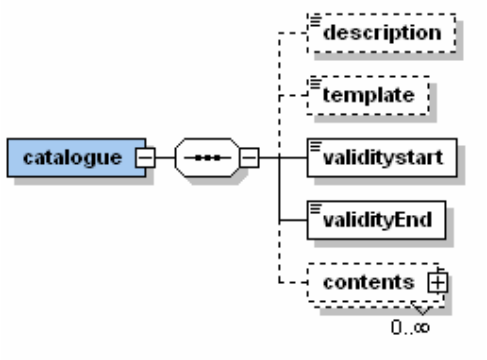
Attribute	Type	Description
catalogueId	String	Unique ID of the Catalogue
description	String	Textual description of the Catalogue
template	String	Template to apply to catalogue structure
validityStart	String	Date of start validity of the catalogue
validityEnd	string	Date of end validity of the catalogue

4.6.3.1 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="contents" type="contentsType" minOccurs="0"/>
            <xs:element name="description" type="xs:string"/>
            <xs:element name=" template " type="xs:string"/>
            <xs:element name="validityStart" type="xs:string"/>
            <xs:element name="validityEnd" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="contentsType">
        <xs:sequence>
            <xs:element name="content" type="contentType"/>
        </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="grants" type=" xs:string " maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Just a note about metadata: it is foreseen to use AxInfo and Dublin Core metadata. At the moment, the most relevant information to be displayed to user when catalogue is requested seem to be: title, description and format from Dublin Core metadata; PARs from AxInfo; other information (if content "owns" to AXMEDIS or to the Kiosk exclusively; if content is one of "top10" downloaded contents, a new entry, and so on) retrieved at runtime when catalogue is created.

diagram



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

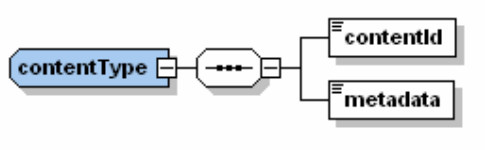
type **pin:CatalogueType**

children **catalogueId contents description template validityStart validityEnd**

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:

- **catalogueId** catalogue unique identifier
- **contents** references to AXMEDIS objects
- **description** catalogue description
- **template** template to apply to catalogue structure **validityStart** date of start validity of the catalogue
- **validityEnd** date of end validity of the catalogue

diagram



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

type **pin:contentType**

children **contentId metadata**

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

4.6.4 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.

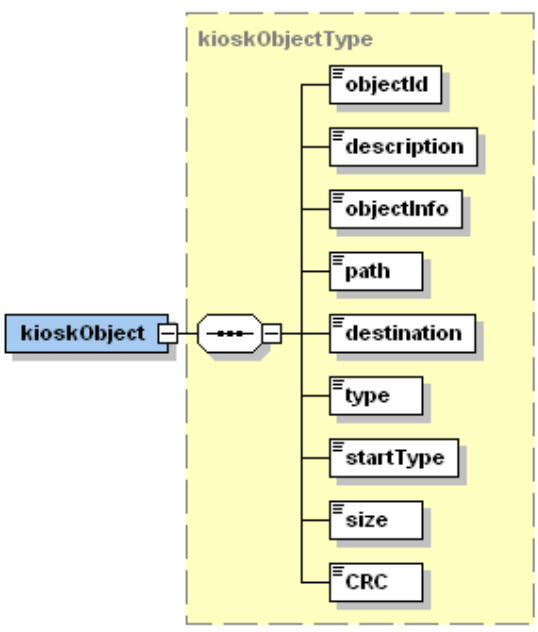
Attribute	Type	Description
objectId	String	Unique ID of the Kiosk Object
description	String	Textual description of the object
objInfo	String	XML field describing the object metadata
path	String	The URL, inside the kiosk environment, of the object
destination	String	The URL of the physical location of 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

4.6.4.1 XML Representation of Kiosk object

```
<?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="typeId" 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 identifier 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

4.6.5 Loading schedule object

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

Attribute	Type	Description
scheduleId catalogueId	String String	The ID of the schedule. 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

4.6.5.1 XML Representation of Loading schedule object

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="loadingScheduleObject" type="LoadingScheduleType"/>
    <xs:complexType name="LoadingScheduleType">
        <xs:sequence>
            <xs:element name="scheduleId" type="xs:string"/>
            <xs:element name="catalogueId" type="xs:string"/>
            <xs:element name="schedule" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

We omit diagram representation because it is very simple and very close to textual representation.

4.6.6 BillingLog 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.
logDate	String	Teh date of the billing
chartId	String	The Id of the chart

4.6.6.1 XML Representation of Billing object

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="billingObject" type="BillingObjectType"/>
    <xs:complexType name="BillingObjectType">
        <xs:sequence>
            <xs:element name="billingId" type="xs:string"/>
            <xs:element name="logDate" type="xs:string"/>
            <xs:element name="chartId" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

We omit diagram representation because it is very simple and very close to textual representation.

4.6.7 Chart object

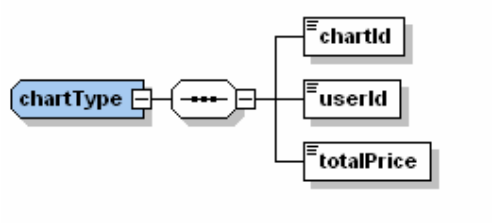
The following table describes the chart related data structure as devised within the kiosk

Attribute	Type	Description
chartId	String	The ID of the chart.
userId	String	The ID of the user
totalPrice	Decimal	The total amount

4.6.7.1 XML Representation of Chart object

```
<?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="chartId" type="xs:string"/>
            <xs:element name="userId" type="xs:string"/>
            <xs:element name="totalPrice" type="xs:float"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

diagram



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

type **pin:chartType**

children **chartId userId totalPrice**

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:

- **chartId** unique chart identifier
- **userId** unique id of the user in the kiosk system
- **totalPrice** the total amount of the chart

5 Used content samples description

The present section has to be considered as a simple reminder to be used to quickly locate places where the content to be used for the demonstrator is described and referenced. The reference sources for content identification and description may be found in DE3.1.3, DE8.1.1, DE8.4.1, DE8.2.1 and DE2.2.1 where content related to kiosks is introduced and presented. The starting point in relation to content to be used for the demonstrator can be summarised as follows:

- Content is focused on art
- Content is designed primarily for a K12 environment
- Basic content samples to be used will have no special look and feel
- Used content will be in Italian and English
- Content is available in form of ZIP files (holding a IMS-Manifest and its resources)
- To adapt content to project purposes it will be necessary to pre-process it (from a ZIP file to an AXMEDIS object, following a set of rules aimed at grant easy composition and adaptation of transformed objects)
- Rules for description, composition, formatting and adaptation will have to be defined consistently with the described process.

Apart from this we have divided info about content in the other deliverables as follows:

- In DE2.2.1 can be found a brief description of content needed for testing purposes,
- In DE8.1.1 can be found the available content and related mapping onto test cases,
- In DE8.4.1 can be found a guideline on editorial format definition,
- In DE3.1.3 can be found a guideline on content aspect and production (with examples and references),
- In DE8.2.1 can be found a guideline on content selection and usage (with examples and references).

6 References

In this first part of the section are reported the relevant project deliverable that are to be used as a reference for the present document:

- DE2.1.1 User Requirements and use cases
- DE2.2.1 Test cases and content description

- DE3.1.1 Guidelines and Specification of research enabling technologies
- DE3.1.2 Framework and tools Specification
- DE3.1.3 AXMEDIS Content Aspects Specification
- DE8.1.1 Content for Test Cases and Validation
- DE8.2.1 Content Selection Guidelines
- DE8.4.1 AXMEDIS Editorial Format Guidelines and basic examples
- DE10.4.1 Assessment and Evaluation Manual
- DE9.3.4 Integrated prototype of content production and distribution in push and on-demand for I-TV
- DE3.1.2.2.11 Specification of AXMEDIS Programme and Publication Tools

In this second part of the section are reported reference text, papers, articles, site and other relevant information sources that have been used to prepare the document and are other than project deliverables.

[1] The Publishers Association – THE GLOSSARY of BOOK TRADE TERMINOLOGY from GLOSSARY OF PUBLISHING TERMINOLOGY 1997 - www.osi.hu/cpd/resources/paglossary.htm

[2] Henry Budgett, J K Johnstone – Glossary of terms associated with the typesetting and printing industries – Based on a series of articles in a newsletter called "Desktop Publisher" published between 1986 and 1989 - The material contained in this glossary is originally the copyright of The Desktop Publishing Company Ltd and must be acknowledged as such if the material is re-used in any other form. However, permission for re-use is freely granted - http://members.aol.com/richardw51/typeglossary.htm

[3] Harold Underdown – The Complete Idiot's Guide to Publishing Children's Books - Appendix A: Glossary of publishing terms and jargon 2004 - http://www.underdown.org/cigglossary.htm

[4] Book Zone Pro - http://www.bookzonepro.com/glossary.html based on the Go Ahead Self-Publish! Glossary, compiled and edited by Eileen Birin - © 2004 Wheatmark, Inc

[5] Mantex - DeskTop Publishing, a glossary of terms 2000 - www.mantex.co.uk/samples/dtp.htm

[6] The Rainwater Press Publishing Primer - A glossary of terms for the electronic publishing, graphic arts, and printing industries 2000 - http://www.rainwater.com/glossary.html

[7] Brandon Hall – Glossary 2004 - http://www.brandonhall.com/public/glossary/

[8] Movie Glossary http://www.factmonster.com/ipka/A0775970.html

[9] Movie Terminology Glossary http://www.imdb.com/Glossary/

[10] The FilmLand Web Glossary by TriJet Productions © 1996 http://www.filmland.com/glossary/Dictionary.html

[11] PBS – Public Broadcasting Service - Digital TV Glossary http://www.pbs.org/digitaltv/glossary.html

[12] Digital Audio Guide – Glossary, D&F Services http://www.digitalaudioguide.com/glossary.htm

[13] Digital TV Glossary - WOSU - http://www.wosu.org/digital/glossary.php

[14] Satellite TV Glossary – Dish Network http://www.dish-network-directv-specials.com/satellite-tv-glossary.aspx

[15] VCR Glossary – Dish Network http://www.dish-network-directv-specials.com/satellite-tv-glossary.aspx

[16] Common Industry Format for Usability Test Reports Glossary http://zing.ncsl.nist.gov/iusr/documents/glossary.htm

[17] Christian Salvaneschi, LOBSTER WEB SERVICE V2.1, GIUNTI Interactive Labs Srl, 2004, Sestri Levante

7 Glossary of acronyms & terms

The present section provides a reference to glossaries referenced / provided in other project documents.

7.1 Glossary of terms in the publishing environment

(Sources: please see References & Sources items [1]-[6])

7.2 Glossary of terms in the e-learning environment

(Sources: please see References & Sources items [7])

7.3 Glossary of terms in the movie, TV, Media… environment

(Sources: please see References & Sources items [8]-[15])

7.4 Other glossaries

(Sources: please see References & Sources items [16])

8 APPENDIX A: Lobster WSDL file

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Lobster" targetNamespace="http://www.giuntilabs.com/Lobster/wsdl/"
xmlns:wsdlns="http://www.giuntilabs.com/Lobster/wsdl/" xmlns:typens="http://www.giuntilabs.com/Lobster/type/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:stk="http://schemas.microsoft.com/soap-toolkit/wsdl-extension"
xmlns:dime="http://schemas.xmlsoap.org/ws/2002/04/dime/wsdl/"
xmlns:ref="http://schemas.xmlsoap.org/ws/2002/04/reference/" xmlns:content="http://schemas.xmlsoap.org/ws/2002/04/content-
type/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
        <schema targetNamespace="http://www.giuntilabs.com/Lobster/type/" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" element-
FormDefault="qualified">
            <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
            <import namespace="http://schemas.xmlsoap.org/wsdl/"/>
            <import namespace="http://schemas.xmlsoap.org/ws/2002/04/reference/"/>
            <import namespace="http://schemas.xmlsoap.org/ws/2002/04/content-type/"/>
        </schema>
    </types>
    <message name="LexSOAPConnector.Login">
        <part name="Domain" type="xsd:string"/>
        <part name="User" type="xsd:string"/>
        <part name="Pass" type="xsd:string"/>
    </message>
    <message name="LexSOAPConnector.LoginResponse">
        <part name="Result" type="xsd:string"/>
    </message>
    <message name="LexSOAPConnector.GetVersion">
    </message>
    <message name="LexSOAPConnector.GetVersionResponse">
        <part name="Result" type="xsd:string"/>
    </message>
    <message name="LexSOAPConnector.GetDomainInfo">
        <part name="SID" type="xsd:string"/>
    </message>
    <message name="LexSOAPConnector.GetDomainInfoResponse">
        <part name="Result" type="xsd:string"/>
    </message>
    <message name="LexSOAPConnector.FindPackage">
        <part name="SID" type="xsd:string"/>
        <part name="Query" type="xsd:string"/>
        <part name="Start" type="xsd:unsignedInt"/>
        <part name="Size" type="xsd:unsignedInt"/>
    </message>
    <message name="LexSOAPConnector.FindPackageResponse">
        <part name="Result" type="xsd:string"/>
```

```
</message>
<message name="LexSOAPConnector.DeletePackage">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.DeletePackageResponse">
</message>
<message name="LexSOAPConnector.InsertPackageRequest">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
    <part name="ObjType" type="xsd:unsignedInt"/>
</message>
<message name="LexSOAPConnector.InsertPackageRequestResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.UpdatePackageRequest">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.UpdatePackageRequestResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.NotifyUploadDone">
    <part name="SID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.NotifyUploadDoneResponse">
</message>
<message name="LexSOAPConnector.GetPackageFolder">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetPackageFolderResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.Shutdown">
    <part name="SID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.ShutdownResponse">
</message>
<message name="LexSOAPConnector.KeepSessionAlive">
    <part name="SID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.KeepSessionAliveResponse">
</message>
<message name="LexSOAPConnector.GetLicense">
    <part name="SID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetLicenseResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetWAList">
    <part name="SID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetWAListResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.SetInsertOptions">
    <part name="SID" type="xsd:string"/>
    <part name="TargetWAID" type="xsd:string"/>
    <part name="AllowPublicAccess" type="xsd:int"/>
</message>
<message name="LexSOAPConnector.SetInsertOptionsResponse">
</message>
<message name="LexSOAPConnector.SetStatus">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
    <part name="Status" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.SetStatusResponse">
</message>
```

```
<message name="LexSOAPConnector.GetPackageInfo">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetPackageInfoResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.FindPackageInfo">
    <part name="SID" type="xsd:string"/>
    <part name="Query" type="xsd:string"/>
    <part name="Start" type="xsd:unsignedInt"/>
    <part name="Size" type="xsd:unsignedInt"/>
</message>
<message name="LexSOAPConnector.FindPackageInfoResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetPackageHistory">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetPackageHistoryResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.NotifyUploadDone2">
    <part name="SID" type="xsd:string"/>
    <part name="VersionLabel" type="xsd:string"/>
    <part name="VersionComment" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.NotifyUploadDone2Response">
    <part name="Result" type="xsd:unsignedInt"/>
</message>
<message name="LexSOAPConnector.SetVersionStatus">
    <part name="SID" type="xsd:string"/>
    <part name="PackID" type="xsd:string"/>
    <part name="VersionNumber" type="xsd:unsignedInt"/>
    <part name="VersionStatus" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.SetVersionStatusResponse">
</message>
<message name="LexSOAPConnector.GetPackageTypes">
    <part name="SID" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.GetPackageTypesResponse">
    <part name="Result" type="xsd:string"/>
</message>
<message name="LexSOAPConnector.FindPackageInfo2">
    <part name="SID" type="xsd:string"/>
    <part name="Query" type="xsd:string"/>
    <part name="PackTypes" type="xsd:string"/>
    <part name="Start" type="xsd:unsignedInt"/>
    <part name="Size" type="xsd:unsignedInt"/>
</message>
<message name="LexSOAPConnector.FindPackageInfo2Response">
    <part name="Result" type="xsd:string"/>
</message>
<portType name="LobsterSoapPort">
    <operation name="Login" parameterOrder="Domain User Pass">
        <input message="wsdlns:LexSOAPConnector.Login"/>
        <output message="wsdlns:LexSOAPConnector.LoginResponse"/>
    </operation>
    <operation name="GetVersion" parameterOrder="">
        <input message="wsdlns:LexSOAPConnector.GetVersion"/>
        <output message="wsdlns:LexSOAPConnector.GetVersionResponse"/>
    </operation>
    <operation name="GetDomainInfo" parameterOrder="SID">
        <input message="wsdlns:LexSOAPConnector.GetDomainInfo"/>
        <output message="wsdlns:LexSOAPConnector.GetDomainInfoResponse"/>
    </operation>
    <operation name="FindPackage" parameterOrder="SID Query Start Size">
        <input message="wsdlns:LexSOAPConnector.FindPackage"/>
```

```
        <output message="wsdlns:LexSOAPConnector.FindPackageResponse"/>
    </operation>
    <operation name="DeletePackage" parameterOrder="SID PackID">
        <input message="wsdlns:LexSOAPConnector.DeletePackage"/>
        <output message="wsdlns:LexSOAPConnector.DeletePackageResponse"/>
    </operation>
    <operation name="InsertPackageRequest" parameterOrder="SID PackID ObjType">
        <input message="wsdlns:LexSOAPConnector.InsertPackageRequest"/>
        <output message="wsdlns:LexSOAPConnector.InsertPackageRequestResponse"/>
    </operation>
    <operation name="UpdatePackageRequest" parameterOrder="SID PackID">
        <input message="wsdlns:LexSOAPConnector.UpdatePackageRequest"/>
        <output message="wsdlns:LexSOAPConnector.UpdatePackageRequestResponse"/>
    </operation>
    <operation name="NotifyUploadDone" parameterOrder="SID">
        <input message="wsdlns:LexSOAPConnector.NotifyUploadDone"/>
        <output message="wsdlns:LexSOAPConnector.NotifyUploadDoneResponse"/>
    </operation>
    <operation name="GetPackageFolder" parameterOrder="SID PackID">
        <input message="wsdlns:LexSOAPConnector.GetPackageFolder"/>
        <output message="wsdlns:LexSOAPConnector.GetPackageFolderResponse"/>
    </operation>
    <operation name="Shutdown" parameterOrder="SID">
        <input message="wsdlns:LexSOAPConnector.Shutdown"/>
        <output message="wsdlns:LexSOAPConnector.ShutdownResponse"/>
    </operation>
    <operation name="KeepSessionAlive" parameterOrder="SID">
        <input message="wsdlns:LexSOAPConnector.KeepSessionAlive"/>
        <output message="wsdlns:LexSOAPConnector.KeepSessionAliveResponse"/>
    </operation>
    <operation name="GetLicense" parameterOrder="SID">
        <input message="wsdlns:LexSOAPConnector.GetLicense"/>
        <output message="wsdlns:LexSOAPConnector.GetLicenseResponse"/>
    </operation>
    <operation name="GetWAList" parameterOrder="SID">
        <input message="wsdlns:LexSOAPConnector.GetWAList"/>
        <output message="wsdlns:LexSOAPConnector.GetWAListResponse"/>
    </operation>
    <operation name="SetInsertOptions" parameterOrder="SID TargetWAID AllowPublicAccess">
        <input message="wsdlns:LexSOAPConnector.SetInsertOptions"/>
        <output message="wsdlns:LexSOAPConnector.SetInsertOptionsResponse"/>
    </operation>
    <operation name="SetStatus" parameterOrder="SID PackID Status">
        <input message="wsdlns:LexSOAPConnector.SetStatus"/>
        <output message="wsdlns:LexSOAPConnector.SetStatusResponse"/>
    </operation>
    <operation name="GetPackageInfo" parameterOrder="SID PackID">
        <input message="wsdlns:LexSOAPConnector.GetPackageInfo"/>
        <output message="wsdlns:LexSOAPConnector.GetPackageInfoResponse"/>
    </operation>
    <operation name="FindPackageInfo" parameterOrder="SID Query Start Size">
        <input message="wsdlns:LexSOAPConnector.FindPackageInfo"/>
        <output message="wsdlns:LexSOAPConnector.FindPackageInfoResponse"/>
    </operation>
    <operation name="GetPackageHistory" parameterOrder="SID PackID">
        <input message="wsdlns:LexSOAPConnector.GetPackageHistory"/>
        <output message="wsdlns:LexSOAPConnector.GetPackageHistoryResponse"/>
    </operation>
    <operation name="NotifyUploadDone2" parameterOrder="SID VersionLabel VersionComment">
        <input message="wsdlns:LexSOAPConnector.NotifyUploadDone2"/>
        <output message="wsdlns:LexSOAPConnector.NotifyUploadDone2Response"/>
    </operation>
    <operation name="SetVersionStatus" parameterOrder="SID PackID VersionNumber VersionStatus">
        <input message="wsdlns:LexSOAPConnector.SetVersionStatus"/>
        <output message="wsdlns:LexSOAPConnector.SetVersionStatusResponse"/>
    </operation>
    <operation name="GetPackageTypes" parameterOrder="SID">
        <input message="wsdlns:LexSOAPConnector.GetPackageTypes"/>
        <output message="wsdlns:LexSOAPConnector.GetPackageTypesResponse"/>
    </operation>
```

```
        <operation name="FindPackageInfo2" parameterOrder="SID Query PackTypes Start Size">
            <input message="wsdlns:LexSOAPConnector.FindPackageInfo2"/>
            <output message="wsdlns:LexSOAPConnector.FindPackageInfo2Response"/>
        </operation>
    </portType>
    <binding name="LexSOAPConnectorSoapBinding" type="wsdlns:LobsterSoapPort">
        <stk:binding preferredEncoding="UTF-8"/>
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="Login">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.Login"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Domain User Pass"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="GetVersion">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetVersion"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="GetDomainInfo">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetDomainInfo"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="FindPackage">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.FindPackage"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID Query Start Size"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="DeletePackage">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.DeletePackage"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="InsertPackageRequest">
            <soap:operation soapAc-
tion="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.InsertPackageRequest"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID ObjType"/>
            </input>
```

```
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="UpdatePackageRequest">
            <soap:operation soapAc-
tion="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.UpdatePackageRequest"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="NotifyUploadDone">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.NotifyUploadDone"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="GetPackageFolder">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetPackageFolder"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="Shutdown">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.Shutdown"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="KeepSessionAlive">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.KeepSessionAlive"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="GetLicense">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetLicense"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
```

```
            </output>
        </operation>
        <operation name="GetWAList">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetWAList"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="SetInsertOptions">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.SetInsertOptions"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID TargetWAID AllowPublicAccess"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="SetStatus">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.SetStatus"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID Status"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="GetPackageInfo">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetPackageInfo"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="FindPackageInfo">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.FindPackageInfo"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID Query Start Size"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="GetPackageHistory">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetPackageHistory"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="NotifyUploadDone2">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.NotifyUploadDone2"/>
```

```
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID VersionLabel VersionComment"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="SetVersionStatus">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.SetVersionStatus"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID PackID VersionNumber VersionStatus"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="GetPackageTypes">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.GetPackageTypes"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
        <operation name="FindPackageInfo2">
            <soap:operation soapAction="http://www.giuntilabs.com/Lobster/action/LexSOAPConnector.FindPackageInfo2"/>
            <input>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="SID Query PackTypes Start Size"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://www.giuntilabs.com/Lobster/message/" encoding-
Style="http://schemas.xmlsoap.org/soap/encoding/" parts="Result"/>
            </output>
        </operation>
    </binding>
    <service name="Lobster">
        <port name="LobsterSoapPort" binding="wsdlns:LexSOAPConnectorSoapBinding">
            <soap:address location="http://myserver/LobsterWebService/Lobster.wsdl"/>
        </port>
    </service>
</definitions>
```